UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

DETECCIÓN DE COMUNIDADES EN LOS SISTEMAS DE RECOMENDACIÓN: USO DE UNA CACHÉ EN RBOX 2.0.

JUAN ALBERTO DANIEL OCHOA JOHN

PROFESOR GUÍA: DR. EDMUNDO LEIVA LOBOS

TESIS DE GRADO PRESENTADA
EN CONFORMIDAD A LOS REQUISITOS PARA
OBTENER EL GRADO DE MAGÍSTER EN
INGENIERÍA INFORMÁTICA

SANTIAGO – CHILE
2014

AGRADECIMIENTOS

Gracias a todos.

A mi incondicional familia y a mi fantástica mujer, esto no habría sido posible sin ustedes...

RESUMEN

El objetivo de esta tesis es verificar si precomputar comunidades mejora la eficiencia de los sistemas de recomendación (SR). Los SR entregan recomendaciones como salida, con el fin de presentar a un usuario activo un conjunto de opciones de su interés, basado en distintos aspectos, uno de ellos son las comunidades a las que el usuario pertenece. La tesis se ha apoyado en un *framework* de eventos colaborativos llamado 3-Ontology, implementado en una API denominado Rbox 2.0 donde la comunidad es una de sus dimensiones esenciales.

Las interacciones entre usuarios y aplicaciones en la Web 2.0 hacen emerger comunidades. Los seguidores de *Twitter* de un usuario forman una comunidad, los que han comprado un mismo libro en *Amazon* también. Los SR hacen uso de comunidades, y típicamente es preciso usar estrategias para detectarlas. No obstante, estas estrategias no están apoyadas por un mecanismo que soporte el proceso de recomendación íntegramente y sus resultados deben ser reprocesados cada vez al momento de computar una recomendación. El contar con un *framework* que estandariza la representación de la información de las distintas interacciones que ocurren en plataformas basadas en la Web 2.0, permite que la detección de comunidades pueda ser realizada de manera estándar. Como añadidura a RBox, se ha construído un mecanismo de abstracción, representación, detección y persistencia - *community cache* - que permite contar con un respaldo de las comunidades ya detectadas en recomendaciones anteriores.

Para validar la eficacia del modelo propuesto, se ha desarrollado un sistema de recomendación, que utiliza el modelo, y recomienda en base a distintas estrategias de detección de comunidades existentes en la literatura. Los escenarios de prueba definidos, muestran que el tiempo promedio necesario para recomendar utilizando la información de las comunidades detectadas con el apoyo del modelo construido es hasta un 7 % menor que cuando se realiza el mismo ejercicio sin él. Como aporte al área de SR, se provee una capa de abstracción al *framework* RBox, que permite detectar, persistir y utilizar en la recomendación, información respecto a las comunidades emergentes de manera eficiente.

Palabras Claves: *community detection*, Sistemas de Recomendación, Web 2.0, 3-Ontology, *community cache*, *social networking*

ABSTRACT

The main objective of this thesis is verify if the efficiency of recommendation systems (RS) improves when the communities are pre calculated. The output of RS are recommendations presented to an active user and intends to guide his navigation on contents of their interest. Recommendations are based on different aspects, one of those aspects are the communities to the user belongs. This thesis has been supported on a framework of collaborative events called 3-Ontology, implementing an API called RBox 2.0, where the communities are one of the essentials dimensions.

The result of the interactions between users and Web 2.0 applications is the apparition of many communities. The Twitter followers of an user forms a community as well as the buyers of a same book on Amazon. The RS uses communities and typically different detection strategies are needed to identify them. However, those strategies are not backed on a mechanism that entirely supports the recommendation process, and the detection results must be reprocessed in every recommendation. The fact of counting with a framework that standardizes the representation of every interaction produced in a social media environment permits that the community detection may be realized in a standard way. As a plugin to RBox, we present a mechanism of community abstraction, representation, detection and persistence (community cache), that permits handling a backup of the communities that have been already detected in past recommendations.

In order to validate the efficacy of the proposed model, we have developed a recommendation system, that uses the model, and performs recommendations based on different community detection strategies, based on literature. The testing scenarios show that the average recommendation time is seven percent lower when the recommendations are performed with the proposed model. As a contribution to the RS area, we present an abstraction layer for the Rbox framework, that permits to detect, persist and use efficiently in the recommendation, the information related to different emerging communities.

Keywords: community detection, recommendation systems, Web 2.0, 3-Ontology, community cache, social networking

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

El objetivo de este capítulo es presentar el problema y la propuesta de la solución de la detección de comunidades emergentes desde las interacciones sociales en la Web 2.0 usando el *framework* de eventos de la 3-Ontology. La propuesta es incorporar la dimensión de la comunidad en el *framework* para crear sistemas de recomendación, resultando esto en un *framework* que apoya, soporta y abstrae la generación de ellos, llamado RBox.

1.1 ANTECEDENTES Y MOTIVACIÓN

Vivimos en un mundo donde la cotidianidad ha sido invadida por distintos aparatos tecnológicos inteligentes, que han permitido a las personas acceder a la Web en cualquier momento y en cualquier lugar. Esto, sumado a la potente aparición de la Web 2.0, implica que los usuarios de Internet ya no solo se dedican a consumir información, sino que, muy por el contrario, ahora asumen roles como actores determinantes en la generación de contenidos (Andersen, 2007). Ciertamente, las barreras que antes existían entre ser productor y consumidor de información han caído, empoderando a los usuarios a ser un agente activo en el flujo de la información. La Web 2.0, en definitiva, ha propiciado una forma diferente de usar la Web como canal de comunicación (Padula, Reggiori, Capetti, 2009).

El uso masivo de aplicaciones móviles, Web y todo tipo de aplicaciones sociales en general ha provocado que Internet almacene una enorme cantidad de información. Esto ha generado un problema para los millones de usuarios de la Web, ya que es complejo acceder a información que sea acorde con los gustos e intereses particulares para cada uno de ellos. En ese sentido, el asistir a los usuarios de Internet a filtrar y discriminar información relevante, ha sido una de las principales motivaciones de investigadores de todo el orbe, principalmente enfocándose en dos grandes áreas:

los motores de búsqueda y los sistemas de recomendación. Estos últimos filtran la información de acuerdo a distintas dimensiones (Adomavicius Tuzhilin, 2011), como la confiabilidad y el prestigio de los usuarios (Victor, De Cock, Cornelis, 2011), geolocalización (Symeonidis, Papadimitriou, Manolopoulos, Senkul, Toroslu, 2011) y el contexto de interacción (Adomavicius Tuzhilin, 2011) y a una serie de directrices, que van desde el comportamiento de otros usuarios, preferencias explicitadas en el pasado propias, ajenas y aquellas tipificadas por las comunidades en las que un usuario participa (Arab Afsharchi, 2014).

Comunidades online como *Twitter*, *Facebook* y *YouTube* han tenido un crecimiento notable durante los últimos años, e incluso son posicionados entre los sitios más visitados de Internet. En estas plataformas, los usuarios pueden crear, explorar y compartir de diversas formas (Brambilla, Fraternali, Vaca, 2011) con distintos grupos de manera explícita, como en *Facebook* o Google+ o implícita, como en *Twitter*. Una manera de explicitar las comunidades existentes en un entorno colaborativo social es mediante la detección de comunidades (en inglés, community detection). Una comunidad típicamente se define como un conjunto de individuos que tienen propiedades en común (Du, Wu, Pei, Wang, Xu, 2007) y para detectarlas existen diversas técnicas y algoritmos (Plantié Crampes, 2013).

Las aplicaciones de la detección de comunidades son diversas. Entre ellas se puede encontrar el análisis temporal de las comunidades online, como los bloggers o los sistemas de recomendación de medios digitales (Lin, Sundaram, Chi, Tatemura, Tseng, 2007; Schlitter Falkowski, 2009), en donde interesa conocer las correlaciones temporales entre los usuarios y sus relaciones. Además, es posible conocer los tipos de transformaciones (Palla, Barabási, Vicsek, Hungary, 2007) que tiene una comunidad en el tiempo: compresiones, contracciones y divisiones. Es posible, también, analizar la detección de tópicos en sistemas de etiquetado colaborativos, para detectar los temas más relevantes que son tocados al interior de una comunidad, en base a repeticiones o folksonomías (Simpson, 2008; Papadopoulos, Kompatsiaris, Vakali, 2009). Por lo que es relevante contar con la información de las comunidades al momento de realizar una recomendación.

1.2 DESCRIPCIÓN DEL PROBLEMA

En un contexto de interacción social se forman diversas comunidades de usuarios en base a distintos intereses o temas. Esas comunidades no siempre están visibles ni disponibles para su uso en las aplicaciones de la Web 2.0. Además, estas comunidades pueden estar de manera latente y en respuesta a contenido recientemente popular y son información que los sistemas de recomendación podrían utilizar para mejorar su desempeño.

En concreto, al momento de generar una recomendación, es necesario procesar y detectar cada vez esas comunidades latentes, con la implicancia de que en cada ejecución del algoritmo de recomendación se destinen recursos computacionales a detectar comunidades que ya han sido identificadas en iteraciones anteriores, aumentando innecesariamente el tiempo de respuesta necesario para otorgar una recomendación. Luego, la pregunta de investigación precisa de esta tesis es la siguiente: ¿es posible mejorar el desempeño de los sistemas de recomendación pre computando de comunidades implicadas en la recomendación?.

1.3 SOLUCIÓN PROPUESTA

Este proyecto de investigación será del tipo Investigación Aplicada I+D (Investigación y Desarrollo):

1. **Investigación:** El contar con un mecanismo de *'cache'* para manejar las comunidades detectadas en ejecuciones previas de un sistema de recomendación, permite mejorar el tiempo de respuesta necesario para otorgar una recomendación.

2. **Desarrollo:** Se diseñará y construirá una arquitectura orientada a servicios, que faculte a RBox 2.0 (Vasquez, 2014) a manejar la detección de comunidades y la construcción de sistemas de recomendación que consideren a esta dimensión de la 3-Ontology.

1.3.1 Características de la Solución

Las características de la solución propuesta son las siguientes:

- Un componente de *software* que hace uso del API que provee RBox, basado en patrones de diseño acordes con la implementación ya realizada.
- Un sistema orientado a servicios que posibilite la detección de comunidades aplicando diferentes enfoques.
- La disponibilización de un dataset inédito, que será utilizado como base para su mapeo al esquema 3-Ontology.
- Prueba de tiempo de ejecución aplicando distintos métodos de detección de comunidades, en un sistema de recomendación generado con RBox.

1.3.2 Propósito de la Solución

- Proveer un componente de *software* extensible, cuyo foco sea el análisis de comunidades y, en particular, la detección de comunidades.
- Añadir a la arquitectura de RBox un componente de *software* que sea capaz de consumir este servicio, permitiendo así la implementación de procedimientos que hagan uso de la dimensión comunidad.
- Extender la herramienta RBox, otorgando al área de los SR una herramienta que se empodera cada vez más en el uso del darse-cuenta colaborativo en el esquema basado en eventos propuesto por Leiva-Lobos y Covarrubias (2002).

1.4 OBJETIVOS Y ALCANCE DEL PROYECTO

A continuación se presenta el objetivo general del proyecto, el cual se lleva a puerto a través del cumplimiento de los diversos objetivos específicos. Además, se declaran los alcances del

trabajo de tesis.

1.4.1 Objetivo general

Detectar comunidades implícitas que se forman a través de las interacciones de los usuarios en la Web social para conseguir sistemas de recomendación más eficientes utilizando el *framework* que provee Rbox 2.0.

1.4.2 Objetivos específicos

1. Seleccionar técnicas y/o algoritmos que permitan la detección de comunidades.
2. Definir la arquitectura del servicio que se añadirá a RBox 2.0, junto con un proceso de encapsulamiento de los datos del *framework* para estandarizar el consumo del servicio.
3. Implementar y añadir a RBox 2.0 un servicio que detecte y persista las comunidades (*community cache*) a partir de los datos encapsulados.
4. Implementar un algoritmo de recomendación utilizando la información de las comunidades como antecedente, para registrar el tiempo de respuesta requerido para entregar una recomendación.
5. Desarrollar conclusiones en relación a los resultados obtenidos.
6. Publicar los resultados en una revista de la especialidad.

1.4.3 Alcances

El trabajo de tesis propuesto tiene los siguientes alcances:

- Contempla solamente la dimensión de la comunidad de la 3-Ontology y en particular, la detección de comunidades.

- Se basa en la definición de SR y alcances establecidos en la tesis de Magíster de Vásquez (2014).
- Solamente se validará con el dataset generado, y que está basado en la red social *Twitter*, no obstante todos los modelos realizados serán genéricos.
- El producto de *software* será construído en base a las definiciones y estándares ya definidos en RBox.
- La implementación del servicio de análisis de comunidades será de tipo RESTful y será independiente de RBox.
- La implementación de componentes para RBox hereda las limitantes y alcances ya definidos respecto al entorno de ejecución.

1.5 METODOLOGÍA Y HERRAMIENTAS DE DESARROLLO

1.5.1 Metodología

La metodología usada para este trabajo, como se explicó anteriormente, tiene que ver en el contexto de un proyecto de investigación aplicado I+D.

En el ámbito de la investigación se realizará una prueba empírica respecto del tiempo de ejecución de un sistema de recomendación, bajo una estrategia de comienzo en frío (Lam, Vu, Le, Duong, 2008), y que considere la dimensión de la comunidad. El proceso y el detalle involucrado en la recomendación y en la detección de comunidades propiamente tal será detallado en el Capítulo ??.

En el ámbito del desarrollo, la metodología de desarrollo está basada en la filosofía ágil de SCRUM. Sin roles explícitamente definidos al tratarse de un trabajo personal. La necesidad de la agilidad tiene que ver con la volatilidad de los requerimientos y las variaciones sobre la marcha en la arquitectura, luego y de esta forma, se tiene la flexibilidad suficiente para abordar cambios a lo largo del desarrollo. La documentación que se genera es la mínima indispensable:

- Documentación del código mediante Javadocs.

- Diseño de arquitectura.

1.5.2 Herramientas de desarrollo

Las herramientas que fueron usadas para la realización de este trabajo de tesis son las siguientes:

- **Linux/Elementary OS Luna y Mac OSX 10.9.3**, como soportes del ambiente de desarrollo.
- **Java SE Development Kit 7**, para el desarrollo del *software*. Su uso hereda de la decisión tomada en el desarrollo de RBox, basándose en la extensibilidad definida en la herramienta.
- **Sublime Text 2/3 y Netbeans IDE**, como apoyo al desarrollo de las aplicaciones.
- **Bitbucket**, como repositorio de código GIT.
- **Python 2.7 y Flask**, como entornos de desarrollo del servicio RESTful.
- **igraph**, como librería especializada en el análisis de grafos.
- **TeXstudio**, como herramienta para la construcción de documentos en LaTeX.
- **Notebook MacBook Pro Mid 2012**, Intel Core i5 2.5 GHz, 4 GB 1600 MHz DDR3.
- **Notebook Lenovo Ideapad Y500**, Intel Core i7 3630QM (2400 MHz - 3400 MHz), 8 GB 1600 MHz DDR3.

1.6 RESULTADOS OBTENIDOS

Se ha definido un *framework* compuesto de una arquitectura orientada a servicios y un componente de *software* añadido a RBox. Estas piezas de *software*, permiten realizar una detección de comunidades a partir de un grafo de interacciones sociales para luego retroalimentar el esquema de 3-Ontology con las comunidades detectadas a partir de recomendaciones realizadas. Esto permite

que, al volver a generar una recomendación similar, el tiempo de ejecución necesario para que esta recomendación se complete se reduzca, mediante el uso de una estrategia de persistencia (*community cache*).

1.7 ORGANIZACIÓN DEL DOCUMENTO

El documento se organiza como sigue. En el Capítulo 2 se presenta un marco teórico en donde se exponen los fundamentos conceptuales del trabajo, como detección de comunidades, la 3-Ontology y la descripción de los trabajos realizados anteriormente en el equipo de desarrollo de la universidad. Como antecedente del desarrollo de la solución presentada en esta tesis, en el Capítulo 3 se analizan los distintos componentes que dan forma a la plataforma tecnológica que apoya la experimentación requerida para alcanzar el objetivo general. Luego, en el Capítulo 4, se presenta el primer componente de *software* desarrollado como producto de este trabajo: un servicio para la detección de comunidades. Posteriormente, en el Capítulo 5, se describe arquitecturalmente un componente de *software* que integra a RBox el servicio de detección de comunidades y permite la persistencia y retroalimentación de las comunidades detectadas a partir de una recomendación. En el Capítulo 6 se describe y muestran los resultados del experimento definido para evidenciar el cumplimiento del objetivo general que este trabajo de tesis plantea. Finalmente, en el Capítulo 7 se presentan las conclusiones obtenidas a partir de este trabajo.

CAPÍTULO 2. MARCO TEÓRICO

En este capítulo se exponen los fundamentos teóricos del trabajo de tesis. Primero, se define el concepto de red social y sus formas de representación. Luego, se explora el concepto de análisis de redes sociales orientado principalmente a la detección de comunidades. Posteriormente, se define el concepto de sistema de recomendación, analizando aquellos conceptos que son considerados en este trabajo de tesis. Finalmente, se describe la lógica de la 3-Ontology, junto con un repaso de lo realizado anteriormente en relación a este concepto.

2.1 REDES SOCIALES

Las relaciones interpersonales son básicas para todo ser humano y están presentes en nuestra vida cotidiana, interactuamos con personas a cada momento y en cada lugar. Sociológicamente hablando, es posible definir el concepto de relación interpersonal como el total de interacciones que generan las personas en un contexto determinado (Dalton, 2007).

Estas interacciones pueden presentarse en distintos ámbitos, como por ejemplo, personales o laborales. Si se abstrae este concepto de relación interpersonal, es posible compararlo al contexto actual en lo que se conoce como una red social. Una red social es una estructura compuesta por nodos (individuos u organizaciones) y aristas, que conectan a los nodos formando una relación. Dichas relaciones pueden deberse a distintos tipos de interacciones como amistad, interés, entre otras.

Existen métodos para la representación de redes sociales, no obstante típicamente son utilizados dos. Uno es gráficamente mediante el renderizado de un grafo , otro es a través de una matriz de adyacencia (Wasserman Faust, 1994). Las aristas en las redes sociales pueden contener distintos atributos, como peso, signo y dirección. Cuando las aristas contienen peso, es que a ellas se

ha asociado un valor numérico que mide la relación en cierto aspecto (calidad, cercanía, similitud, entre otros), el signo puede determinar si la relación entre dos nodos es positiva o negativa, finalmente la dirección indica el sentido en el que la relación está definida.

2.2 ANÁLISIS DE REDES SOCIALES

Las redes sociales en un contexto de *social media* son generalmente muy grandes, con millones de usuarios y conexiones entre ellos. Haciendo uso de las capacidades únicas de masividad que provee la *social media*, estas redes están presentando desafíos con respecto a su análisis (Tang Liu, 2010):

I **Escalabilidad:** Las redes sociales pueden crecer a tamaños astronómicos, conteniendo cientos de miles de conexiones. En *Facebook*, por ejemplo, es tan evidente como que la comunidad de fans del fútbol tiene un orden de magnitud de 500 millones de usuarios (Smith, 2014). El análisis de redes sociales clásico tradicionalmente maneja un orden de magnitud de cientos o menos.

II **Heterogeneidad:** Entre dos usuarios puede existir múltiples tipos de relaciones. En la red social Google+, se aborda este concepto a través del uso de círculos para denotar el contexto de relación entre un usuario y sus contactos. Más aún, puede existir una variedad de interacciones entre un mismo conjunto de usuarios, como por ejemplo un mensaje privado, un *like*, un etiquetado en una fotografía en común. Este contenido en su conjunto es relevante para, por ejemplo, recomendar un contenido.

III **Evolución:** Una de las características fundamentales de *social media* es su orden temporal. Por ejemplo, en *Twitter*, que es una red de microblogging, el contenido relevante y de interés para la mayor parte de los usuarios (a.k.a trending topics) cambia rápida y constantemente en cortos períodos de tiempo (Kerr, 2012). Nuevos usuarios aparecen, usuarios "veteranos" dejan de participar, nuevas conexiones se generan, en resumen todo *dataset* que pertenezca a *social media* es dinámico, y se debe identificar constantemente aquellos focos relevantes para el análisis, como por ejemplo nuevas comunidades emergentes.

IV **Inteligencia Colectiva:** En *social media*, las personas tienden a compartir sus conexiones y

parecer respecto a distintos objetos de interés. El conocimiento generado por las masas a través de *tags*, comentarios, reviews y ratings son antecedentes recabados por muchos portales Web como *Amazon* o *Spotify*. Conceptos como la confiabilidad y reputación de los usuarios toman relevancia y trascendencia (Victor, De Cock, Cornelis, 2011) con el objetivo de recomendar a otros individuos conceptos (items en general) que sean de su interés.

V **Evaluación:** Este es un desafío ocasionado íntegramente por las barreras existentes en la obtención de información para entrenar modelos y algoritmos de análisis. Muchos sitios de *social media* implementan políticas de privacidad que limitan la información a la que un agente externo puede acceder. Como experimentar con un escenario real es poco probable, es necesario aproximarse a él (Leskovec Faloutsos, 2006; Hübler, Kriegel, Borgwardt, Ghahramani, 2008), o bien simular condiciones similares (Chakrabarti Faloutsos, 2006; Maiya Berger-Wolf, 2010).

2.3 DETECCIÓN DE COMUNIDADES

En relación con la inteligencia colectiva y la evolución de las interacciones de un conjunto de usuarios en un contexto de redes sociales, las comunidades permiten caracterizar a aquellos usuarios similares. La detección de comunidades es una de las tareas fundamentales en el análisis de redes sociales, ya que permiten describir fenómenos sociales analizando el comportamiento de las masas (Hechter, 1988).

Una comunidad, en el contexto de una red social, es aquella en la que un conjunto de nodos que interactúan entre ellos más que con los que están fuera del grupo. La detección de comunidades puede ser aplicada en distintos contextos del "mundo real", como por ejemplo, en una aplicación de contenido musical, agrupar usuarios con intereses similares y en base a esto recomienda música similar a otros usuarios. En otros casos, las comunidades pueden ser utilizadas para comprimir redes sociales muy grandes o proveer un mecanismo de exploración y navegación para grafos sociales. Es posible clasificar las diferentes líneas de investigación con respecto de la detección de comunidades (Tang Liu, 2010).

La primera surge por la carencia de los métodos clásicos de detección utilizados en las ciencias sociales, que son incapaces de manejar el volumen de datos de las redes sociales en el contexto

de *social media*, y se enfoca en escalar los métodos de detección de comunidades para que puedan manejar redes de gran tamaño (Andersen, 2007; Dourisboure, Geraci, Pellegrini, 2007; Gibson, Kumar, Tomkins, 2005).

La segunda línea nace para abordar el desafío de heterogeneidad, y tiene que ver con la detección de comunidades en redes con entidades e interacciones de distinta naturaleza (Java, Joshi, Finin, 2008; Tang, Liu, Zhang, Nazeri, 2008; Tang, Wang, Liu, 2009), como *Facebook* o *YouTube*.

La tercera línea de investigación nace para abordar el desafío de evolución y tiene que ver con los cambios en el tiempo que tienen todas las redes sociales en un contexto de *social media* (Asur, Parthasarathy, Ucar, 2009; Backstrom, Dwork, Kleinberg, 2007; Palla, Barabási, Vicsek, 2007; Tang, Liu, Zhang, 2012). Por ejemplo, el número de usuarios activos en *Facebook* ha crecido de 1 millón de usuarios activos en 2004 a 1.11 billones de usuarios en 2013 (AP, 2013), lógicamente la cantidad y estructura de las comunidades ha cambiado debido a la inclusión de nuevos usuarios y las interacciones ocasionadas por ellos.

Las aplicaciones de la detección de comunidades son diversas. Entre ellas se puede encontrar el análisis temporal de las comunidades online, como los bloggers o los sistemas de recomendación de medios digitales (Lin, Sundaram, Chi, Tatemura, Tseng, 2007; Schlitter Falkowski, 2009), en donde interesa conocer las correlaciones temporales entre los usuarios y sus relaciones. Es posible conocer los tipos de transformaciones (Palla, Barabási, Vicsek, Hungary, 2007) que tiene una comunidad en el tiempo: compresiones, contracciones y divisiones. Es posible, también, analizar la detección de tópicos en sistemas de etiquetado colaborativos, para detectar los temas más relevantes que son tocados al interior de una comunidad, en base a repeticiones o folksonomías (Simpson, 2008; Papadopoulos, Kompatsiaris, Vakali, 2009), detectando a la comunidad que estaba relacionada a un tag semilla.

2.4 SISTEMAS DE RECOMENDACIÓN

La recomendación es una tarea común en las aplicaciones construidas bajo el concepto de *social media*. Mientras más grande sea la población de usuarios de un sitio de *social media*, más exitoso se vuelve (Tang Liu, 2010). El recomendar contenido a los usuarios tomando como antecedente sus

interacciones y comunidades a las que pertenece, mejora la experiencia que este usuario tiene con respecto a la aplicación, aumentando su fidelidad y permanencia en el sitio. Entonces, es crítico para cualquier aplicación de *social media* ofrecer recomendaciones que insten a los usuarios que consumen sus servicios a generar más y más interacciones.

En ese sentido, los sistemas de recomendación (SR) tienen relevancia, ya que son sistemas que producen recomendaciones o guías personalizadas al usuario a entidades interesantes o útiles entre muchas disponibles (Burke, 2002). En primera instancia, los sistemas de recomendación fueron aplicados a dominios en los que el contexto era comercial, no obstante nada impide aplicarlos en otros dominios, como por ejemplo utilizando la inteligencia colectiva sobre cómo un grupo de periodistas etiquetan noticias (González, 2012).

Una definición formal de los SR considera dos conjuntos, C que contiene a todos los usuarios de un dominio y S que contiene todos los posibles ítems (referido de forma genérica) a recomendar. Luego, se define una función u como una cuantificación respecto de la utilidad que un ítem s tiene para el usuario c. Finalmente, para cada usuario c, que pertenece a C, se debe elegir el (o los, dependiendo del criterio) ítem s' que maximice la utilidad para el usuario (Adomavicius Tuzhilin, 2005). En relación al desafío de escalabilidad de los sistemas basados en *social media*, los conjuntos S y C pueden crecer indefinidamente. Luego, el problema de recomendación se basa en que el espacio S x C no se conoce completamente, sino que sólo una parte de él. Es muy improbable que un usuario haya tenido alguna interacción con todos los ítems de un sistema. La información referente a los perfiles de usuario, como la edad, género, estado civil, lugar geográfico, comunidades, entre otras y la meta-información asociada a los ítems, son el antecedente que los SR utilizan para personalizar recomendaciones.

En un principio, los SR utilizaban esencialmente tres técnicas de filtrado para recomendar: basado en características definidas por el usuario, basado en productos sin personalización y basado en datos generales de los usuarios. No obstante, esto ha evolucionado con el fin de aprovechar el auge de los sistemas colaborativos y se comienza a utilizar la información referente a dos tipos de interacción del usuario: implícita, que hace referencia a la interacción del usuario con el sistema, como leer una noticia y explícita, que hace referencia a la valoración de un ítem expresando su preferencia, como *liking* o *sharing*. Este último tipo de información se asume más valiosa (Zanker Jessenitschnig, 2009), por cuanto describe la colaboración de los usuarios del sistema y determinan la

relevancia, calidad e interés de un ítem, en desmedro de otro. Existen diversas clasificaciones para los SR, considerando en el aspecto en el que se fundamenta su proceso de recomendación(Adomavicius Tuzhilin, 2005):

- **Basado en el contenido:** Se recomienda un conjunto de ítems basándose en aquellos que son similares a los que un usuario activo ha elegido en el pasado. Usado principalmente en aplicaciones basadas en texto, como la recomendación de sitios Web.
- **Colaborativos:** Se recomienda basado en las valoraciones de otros usuarios. En este tipo de SR captura la inteligencia colectiva que los usuarios tienen sobre un dominio en particular y no depende del contenido del ítem. Las técnicas más usadas (Bobadilla, Ortega, Hernando, Gutiérrez, 2013) para realizar un filtrado en este tipo de SR son *User-User*, que calcula los usuarios más parecidos al usuario activo y se realiza una predicción de preferencia en base a la preferencia de los vecinos más cercanos e *Item-Item*, que calcula para cada ítem aquellos que son similares y en base a esto se realiza una recomendación de aquellos k similares según la preferencia del usuario.
- **Híbridos:** Se recomienda combinando los enfoques colaborativos y basado en contenidos, con el objetivo de minimizar las desventajas de cada uno (Burke, 2002). La hibridación puede realizarse implementando y combinando los resultados de ambos enfoques de manera separada, incorporar alguna propiedad de uno en el otro o bien, construyendo un modelo unificado con características de ambos enfoques.

2.4.1 *Context Aware*

Los enfoques antes mencionados están basados primordialmente en una triada entre el ítem, el usuario y sus interacciones. No obstante, se ha planteado que también es debido considerar la información contextual que es inherente a la interacción (Adomavicius Tuzhilin, 2005), como el tiempo, lugar, compañía, estado de ánimo, entre otras. En términos simples, se quiere marcar la diferencia entre recomendar música para un día soleado o lluvioso, o un restaurante para ir de noche con amigos o la familia un domingo por la tarde. Con el auge de la Web 2.0, cada vez existen más aplicaciones que requieren de información contextual para recomendar.

El concepto de contexto es variado y estudiado por varias disciplinas de investigación, que lo define de acuerdo a una idiosincrasia en particular. No obstante, el contexto desde el punto de vista de los SR, refiere al que es inherente a la interacción del usuario, como por ejemplo la ubicación del usuario, los objetos que están en su entorno, la fecha, estación, clima, temperatura, entre otras, esto aumenta la dimensionalidad del problema con respecto al tradicional enfoque de representar el contexto mediante el ítem y un usuario. El incluir y usar información contextual en SR basados en ella, en enfoques multidimensionales, puede mejorar las recomendaciones (Adomavicius, Sankaranarayanan, Sen, Tuzhilin, 2005).

Bajo esta premisa, se han desarrollado los Context-Aware Recommender Systems (en adelante, CARS) para enriquecer el proceso de recomendación basándose en información útil proveniente del contexto, la cual se puede obtener explícitamente, en donde se consulta directamente a los usuarios mediante formularios Web para obtener la información, implícitamente, a partir de cambios en la ubicación de los usuarios detectados mediante *smartphones* o dispositivos similares o bien, de manera inferida, en donde se obtiene información del contexto a través de métodos estadísticos o minería de datos (Adomavicius Tuzhilin, 2011).

Luego, los CARS sitúan los eventos dentro de un espacio de dimensiones acorde para los SR, cuya dimensionalidad depende explícitamente de la validación realizada en el contexto de la aplicación para la que se recomienda, seleccionando solamente las que aporten de forma positiva al proceso de recomendación (Yujie Licai, 2010).

2.5 LA LÓGICA DE LA 3-ONTOLOGY

La 3-Ontology (Leiva-Lobos Covarrubias, 2002), es un *framework* conceptual que modela el "darse-cuenta colaborativo", está sustentada bajo el área de *Computer-supported cooperative work* (CSCW), que modela el "contexto" de cualquier interacción que tenga un valor colaborativo. El mecanismo de modelamiento es a través de tres conceptos que construyen sentido para los usuarios: comunidades, eventos y lugares. En términos simples, todo contexto de colaboración está "situado" en un espacio, tiempo y cultura.

En la lógica de la 3-Ontology, una comunidad corresponde a un conjunto de usuarios, que

son protagonistas de los eventos y que habita un lugar, un lugar (físico o virtual) es donde habitan los objetos, que son protagonistas de los eventos, un evento está situado en un tiempo y un lugar y siempre son de carácter colaborativo.

El darse-cuenta es una relación dependiente y co-definida entre estos tres conceptos, por lo que no existe una jerarquía entre ellos. Una representación basada en la 3-Ontology permite explicitar el "valor" colaborativo de cada interacción. Luego, una interacción sucede en un lugar, es efectuada por un usuario que pertenece a una comunidad, en un tiempo determinado, etc. La Figura 2-1 resume lo antes mencionado.

Existe una relación de constitución entre los contenedores de sentido, tal que cada uno de ellos posee un servicio asociado que permite darse-cuenta desde distintas perspectivas:

I **De la historia:** Es posible denotar la relación existente entre los eventos efectuados a través de una traza, que es un orden parcial de los eventos efectuados por un usuario o un conjunto de ellos. Las trazas permiten agregar continuidad en el desarrollo del flujo histórico y notificar al usuario del contexto dadas las trazas de los eventos o bien, establecer continuidad de las actividades que realice un usuario en particular.

II **De la presencia:** Es posible constituir los lugares donde suceden los eventos a partir de mapas. Permitiendo a los usuarios ver donde ocurren los eventos, los objetos y personas que están involucrados, entre otros. Los mapas permiten denotar conectividad, respecto a usuarios cercanos en un mismo espacio físico o virtual y también disponibilidad al caracterizar lo que está sucediendo en un lugar determinado, o bien que objetos se encuentran en el.

III **De los usuarios:** Es posible representar comunidades y conjuntos de estas a partir de retratos, que determinan la identidad de los usuarios, afinidad entre ellos, pertenencia implícita como explícita. Los retratos permiten obtener identidad, denotando, para un usuario, quiénes son similares a el y a que comunidades pertenecen y expansibilidad, que permite a una comunidad variar, en el tiempo, la cantidad de usuarios que la componen.

La 3-Ontology es un *framework* de tipo conceptual y no se adscribe a ninguna tipología u ontología de índole informática. No obstante, es posible considerarla una meta-ontología para definir y modelar ontologías para dominios colaborativos en específico.

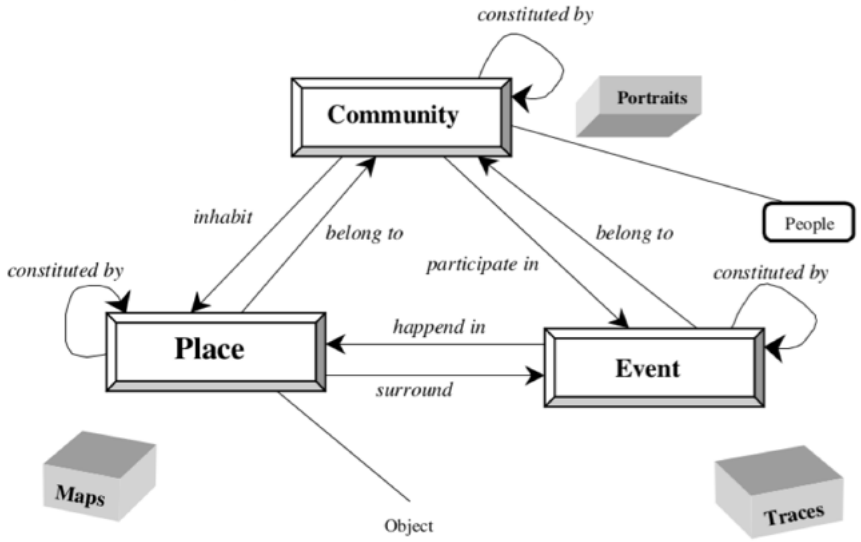

Figura 2-1: *Lógica de la 3-Ontology.*

En ese sentido, Palomino (2012) presenta un modelo conceptual (MC) que extiende el modelamiento de Synergy (Tareen, Lee, Lee, 2010), basándose en la meta-ontología que provee la 3-Ontology. Este modelo permite describir las interacciones que suceden dentro de una red social, no obstante no cuenta con una implementación concreta ni modela las tres formas de darse-cuenta colaborativo ni las relaciones existentes entre los tres contenedores de sentido.

Como consecuencia y basándose en el MC antes mencionado, Vázquez (2013) presenta RBox, un *framework* para la creación de sistemas de recomendación. La Figura 2-2 muestra el modelo entidad-relación propuesto por RBox, el cual es una implementación concreta de la 3-Ontology. La implementación que provee RBox completa el *gap* existente con respecto a MC. No obstante, no considera la exploración detallada de todos los contenedores de sentido, en concreto lugares y comunidades. En este trabajo de tesis se comienza la exploración de la dimensión comunidad, añadiendo a RBox una extensión que permite detectar comunidades y realizar recomendaciones considerando a estas como un antecedente.

2.6 RESUMEN

En este capítulo se realizó una revisión de los fundamentos teóricos del trabajo de tesis. Para comenzar, se ha definido el concepto de red social. Luego, se presentó el concepto de análisis

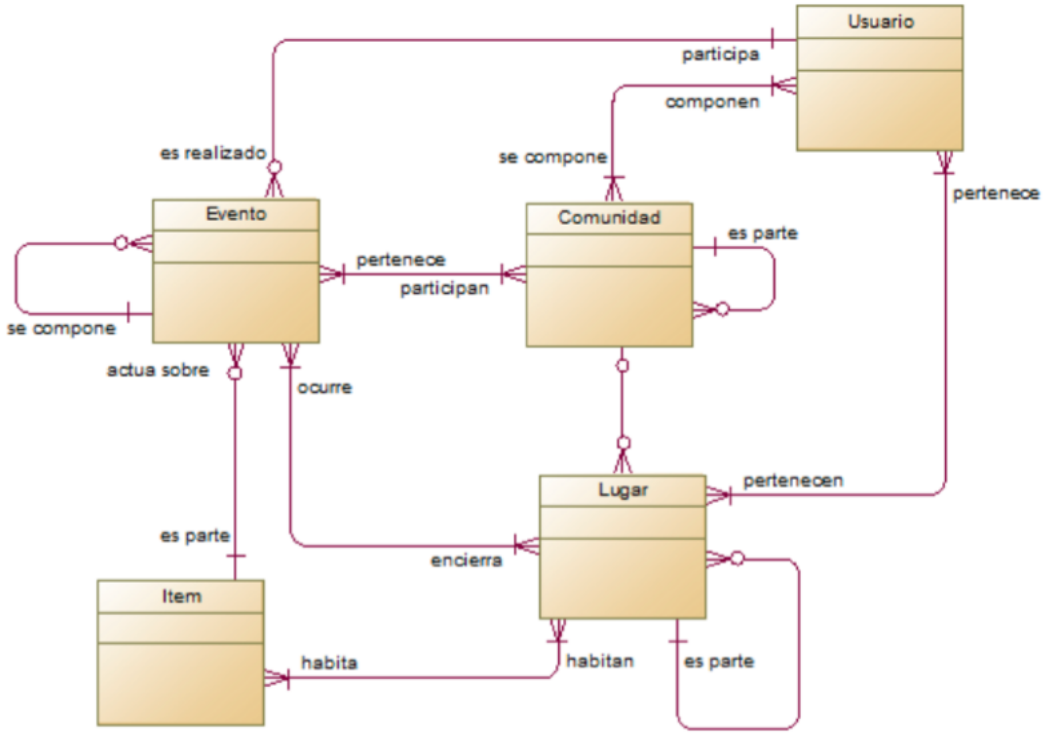

Figura 2-2: *Modelo conceptual de RBox.*

de redes sociales y los desafíos que supone para el área la inclusión de *social media*. Posteriormente, se repasa el concepto de detección de comunidades y se explicitan distintas líneas de investigación. A continuación se trata el concepto de sistemas de recomendación de forma genérica y como han evolucionado a aquellos que consideran el contexto al momento de recomendar. Finalmente, se expone el concepto de la 3-Ontology, junto con los contenedores de sentido y se aborda una contribución de este trabajo de tesis a una implementación basada en este *framework*.

CAPÍTULO 3. VISIÓN GENERAL DE LA SOLUCIÓN

En este capítulo se describe conceptualmente la solución desarrollada en este trabajo de tesis. En primer lugar, se expone el modelamiento que fundamenta el desarrollo de los distintos componentes de *software* que este trabajo presenta. Posteriormente, se describe el mecanismo de *community cache* detonado al momento de recomendar. Luego, se explica cómo esos conceptos son transformados en un modelamiento de *software*. Finalmente, se muestra la integración entre los distintos componentes con apoyo de un diagrama general de la solución.

3.1 MODELAMIENTO

En esta sección se detalla el modelamiento de los distintos aspectos involucrados en el desarrollo de este trabajo de tesis. El modelamiento sienta las bases sobre las que la solución es construída. En otras palabras, el resultado de este ejercicio conforma los requisitos funcionales que la solución debe contemplar.

3.1.1 Modelamiento de interacciones

Como ya se ha mencionado anteriormente, en un contexto de social media pueden ocurrir diversas interacciones, como un mensaje, un 'me gusta', un follow, entre otras. En una red social hay diversas interacciones y depende del contexto y del dominio la relevancia de cada una de ellas en relación a su consideración en una recomendación hacia un usuario activo.

Son estas interacciones las que denotan a una comunidad. Por ejemplo, si en un contexto de una tienda virtual, un usuario realiza una valoración (*rating*) mediante el método que fuere, como por ejemplo una valoración de un artículo en una escala del uno al cinco, dicha valoración

será considerada, de una u otra manera, al momento de recomendar aquel artículo a otro usuario que pertenezca a la misma comunidad.

No obstante, diferentes interacciones pueden denotar muchos segmentos de interés para un usuario. Hipotéticamente hablando, si se considera un contexto de social media en el que sólo existen dos posibles interacciones: amistad y *following*, es posible que un usuario establezca una interacción de amistad con personas que conozca físicamente o virtualmente. No obstante, es posible que genere eventos de *following* con personas relacionadas con temas en boga, como una noticia importante, algún evento deportivo, entre otras. Ante esto pueden ocurrir dos situaciones principales, que los segmentos o comunidades de interés denotadas por ambas interacciones para un usuario sean similares, o que ambas interacciones determinen segmentos de interés divergentes entre sí.

En RBox, las interacciones son modeladas a través de eventos, siguiendo las directrices establecidas por el meta modelo de la 3-Ontology. Pragmáticamente, existe una interfaz llamada Event (ver Figura 3-1), que define este meta modelo. Por lo que todo evento debe tener un valor y contextualmente debe ser realizado por un usuario, enfocado en un objeto en particular (que puede ser otro usuario), situado en una comunidad, realizado en un lugar en particular y debe contar con una marca temporal.

Entonces, uno de los problemas que deben ser resueltos en la implementación de la solución es el de realizar una correspondencia entre el modelo de datos del dominio situado en un ambiente de social media y una definición de evento tipada por RBox.

3.1.2 Modelamiento de la detección de comunidades

En ese sentido y considerando lo anterior, es que la detección de comunidades que la solución provea debe considerar a aquellas que están latentes en el dominio de los distintos eventos. Por ende, es necesario abstraer el concepto de evento de manera tal que la solución sea capaz de detectar comunidades en base a todas aquellas interacciones que sean definidas como relevantes para un cierto dominio de análisis.

En el mundo de la tecnología (y de los repositorios de código) existen distintas librerías que implementan mecanismos y técnicas para detectar comunidades que existen en la literatura:

```java
package cl.usach.diinf.rbox.data;

/**
 * Interfaz que representa el comportamiento basico de un evento basado en el framework
 * conceptual 3-Ontology
 *
 */
public interface Event {
        /**
         * Obtiene el identificador del usuario
         * @return id del usuario
         */
        long getUserId();
        /**
         * Obtiene el identificador del item
         * @return id del item
         */
        long getItemId();
        /**
         * Obtiene una marca temporal del evento
         * @return timestamp
         */
        long getTimestamp();
        /**
         * Obtiene el lugar donde se efectuo el evento
         * @return placeId
         */
        long getPlaceId();
        /**
         * Obtiene la comunidad donde se efectuo el evento
         * @return communityId
         */
        long getCommunityId();
        /**
         * Obtiene la valoracion realizada por el usuario al item. Este puede ser de cualquier clase.
         * @return valor del evento
         */
        <T> T getValue();
}
```

Figura 3-1: *Interfaz Event del código fuente de RBox.*

- Gephi (gephi.github.io/)
- igraph (igraph.org/)
- graph-tool (graph-tool.skewed.de/)
- JGraphT (jgrapht.org/)
- GraphStream (graphstream-project.org/)
- entre otras ...

Su uso, en desmedro de implementar los algoritmos desde su publicación, tiene la tiene la gran ventaja de contar con una comunidad que respalda su uso, funcionamiento, resultados y que además hace el código mantenible.

No obstante, es claro notar que estas librerías, o APIs, están enfocadas en un modelamiento en base a grafos, en donde los nodos representan a los usuarios de una red social y las aristas representan una conexión (a partir de interacciones, según la red social) entre dos usuarios. Por ende, otro de los problemas que deben ser resueltos por la solución presentada por este trabajo, es el de encapsular la información contenida por las distintas instancias de los objetos que pertenezcan una clase que implemente la interfaz Event, con el objetivo de generar un grafo de interacciones.

Al contar con un grafo de interacciones, es posible utilizar las APIs disponibles para análisis de grafos (la detección de comunidades incluida) con el objetivo de detectar las comunidades latentes, que existen en aquel grafo.

3.1.3 La 3-Ontology y las comunidades

Las comunidades son una dimensión considerada en el meta modelo de la 3-Ontology y está definida en el *framework* de RBox como muestra el código de la Figura 3-2. Una comunidad, entonces, está formada principalmente por un conjunto de usuarios. Cuando se hace uso de la detección de comunidades, se debe contar con un mecanismo o componente que sea capaz de integrar los dos mundos: el de la teoría de grafos con el correspondiente a la 3-Ontology.

```java
package cl.usach.diinf.rbox.data;

import java.util.List;


import it.unimi.dsi.fastutil.longs.LongSet;

/**
 * Interfaz que representa el comportamiento básico de una comunidad basado en el framework
 * conceptual 3-Ontology
 * @author Rodrigo
 */
public interface Community extends List<User>{
    /**
     * Obtiene el identificador de la comunidad
     * @return id comunidad
     */
    long getId();

    /**
     * Obtiene el nombre de la comunidad
     * @return id comunidad
     */
    String getName();

    /**
     * Obtiene un conjunto de identificadores de {@link User} de la comunidad
     * @return Conjunto de identificadores
     */
    LongSet getUserIds();
}
```

Figura 3-2: *Interfaz Community del código fuente de RBox.*

Como se ha mencionado en la sección anterior, el componente que sea responsable de manejar la detección de comunidades, estará trabajando bajo un dominio de la teoría de grafos y no necesariamente debe inmiscuirse con el meta modelo planteado por la 3-Ontology. Esto fundamentado en el principio de única responsabilidad (single responsibility principle, en inglés) en la ingeniería de *software* (Martin, 2003), el cual plantea que cada componente de una aplicación debe responder a un propósito atómico e independiente. En ese sentido, la detección de comunidades puede ser considerado un propósito independiente del proceso o *framework* que haga uso de sus beneficios.

Lo anterior genera una arista adicional a considerar en el diseño de la solución que es, precisamente, la integración entre los dos mundos. Debe existir un componente, herramienta o librería que sea capaz de realizar una transformación desde la meta información que contiene un grafo con comunidades detectadas, hasta un esquema definido en base a las directrices que plantea la 3-Ontology, es decir, pasar de un conjunto de nodos y aristas pertenecientes a un *'cluster'*, hacia una comunidad, que contiene un conjunto de usuarios.

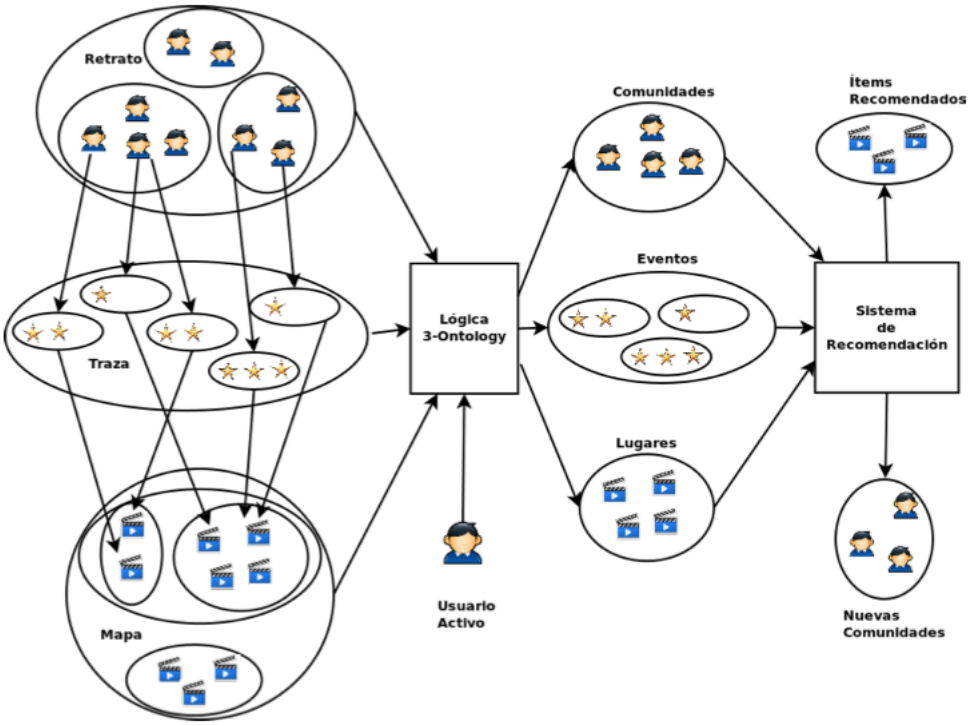

Figura 3-3: *Flujo propuesto por RBox para llevar a cabo una recomendación.*

3.2 COMMUNITY CACHE

El objetivo principal de este trabajo es verificar si contar con un mecanismo de *'cache'* para manejar las comunidades en ejecuciones previas de un sistema de recomendación, permite mejorar el tiempo de respuesta que es requerido para otorgar una recomendación a un usuario activo. En ese contexto, es necesario definir el mecanismo de *cache* que será construído y como actúa en el flujo actual de la recomendación, planteado por el *framework* RBox.

Este flujo considera (de izquierda a derecha en la Figura 3-3) , precisamente, un *mapping* desde una red social hacia la lógica de la 3-Ontology. Posteriormente, para un usuario activo se tiene definición de los contenedores de sentido: comunidades, eventos y lugares. Luego, y al momento de realizar una recomendación, se toma en cuenta aquellos contenedores de sentido tipificados por distintos factores, como el contexto temporal en el que fueron realizados, como por ejemplo, lugares visitados entre una fecha y otra.

Lo anterior da como resultado un conjunto de ítems recomendados, así como también nuevas comunidades. Son estas comunidades las que deben ser persistidas y, de esta manera, estar disponibles para su uso como retroalimentación hacia recomendaciones posteriores. Una *cache* es

necesaria debido a que cabe la posibilidad de que las comunidades detectadas en distintas iteraciones del algoritmo de recomendación ya hayan sido detectadas en iteraciones precedentes, por lo que su manejo debe ser tratado con mayor ligereza que aquellas que son detectadas por primera vez. El concepto de ligereza en el manejo de las comunidades pasa por incorporar al mecanismo de *cache* dos niveles de manejo: uno que faculte la activación o no de la detección de comunidades en una recomendación y otro que maneje, de cierta forma, las distintas apariciones de las comunidades detectadas.

3.2.1 Primer nivel de *cache*

El primer nivel de *cache* considera un mecanismo para enfrentar el fenómeno de partida en frío de los sistemas de recomendación, en particular el de las comunidades de un usuario. Si el contexto de social media en el que un recomendador esté funcionando no considera de manera explícita el concepto de comunidad (por ejemplo, Google+ si lo hace), entonces en el esquema de la 3-Ontology, los usuarios no tendrían comunidades asociadas en primera instancia.

No obstante, lo anterior no quiere decir que estas no existan, por lo que se define un mecanismo que active la detección de comunidades si un usuario activo no pertenece a ninguna comunidad. En el caso de redes sociales de *microblogging*, como Twitter, las comunidades no son explícitas y la gran cantidad de tópicos que son tratados (y a la velocidad que son tratados), implica que un usuario puede estar inmerso en distintas comunidades, de manera implícita.

Esto puede tener un impacto en el tiempo requerido para realizar una recomendación. Ciertamente, el proceso de detección de comunidades durante la generación de una recomendación, representa un costo de tiempo adicional. A esto hay que agregar el tiempo requerido para operaciones de persistencia, como lectura y escritura de una base de datos, por ejemplo.

Luego, el primer nivel de *cache* permite establecer una estrategia de detección de comunidades, en la que, por ejemplo, se puede utilizar un proceso que, luego de cierto tiempo planificado, desactive el primer nivel de *cache*, con la finalidad de actualizar las comunidades a las que pertenece un usuario. Esto en desmedro de realizar la detección de comunidades de manera recurrente y cada vez que una aplicación lo requiera.

3.2.2 Segundo nivel de *cache*

El segundo nivel de *cache* es un mecanismo de manejo de las comunidades ya detectadas. Lo que se busca es mitigar, de cierta forma, el aumento en el tiempo necesario para realizar una recomendación considerando detección de comunidades. Si se asume que una comunidad puede ser detectada más de una vez, en más de una iteración del algoritmo de recomendación, entonces lo que se debe hacer es identificar que esa comunidad ya existe.

Cuando una comunidad ya ha sido detectada en iteraciones anteriores del algoritmo, no deben ocurrir las operaciones de persistencia. Se debe evaluar si una comunidad detectada (C') existe en el medio persistente (C) comparando al conjunto de usuarios (u) que éstas contienen. Si ambos conjuntos, digamos Cu y C'u son iguales, entonces las comunidades son iguales (*hit*) y la persistencia de la comunidad detectada debe omitirse. En caso contrario (*miss*), las operaciones de persistencia para la comunidad detectada deben ejecutarse.

Lo que se pretende es provocar un efecto de reducción de tiempo al mediano plazo. Lógicamente, en las primeras iteraciones del algoritmo, y debido al fenómeno de partida en frío, la cantidad de *hits* será menor a la cantidad de *miss* al momento de evaluar la persistencia de una comunidad detectada. No obstante, a medida que las recomendaciones ocurran, es posible que el proceso de detección de comunidades arroje resultados que ya han ocurrido en iteraciones anteriores, por lo que el mecanismo de persistencia para esa comunidad no será efectuado. Luego, se ahorra el tiempo requerido para ello y la recomendación ocurrirá más rápido.

3.3 COMPONENTES DE *SOFTWARE*

En las secciones anteriores se realizó un análisis de aquellas dificultades y consideraciones que debe tener la solución. En esta sección se analizarán las características y responsabilidades que deben tener los componentes de *software*, que serán desarrollados con la finalidad de superar aquellas dificultades y manejar aquellas consideraciones.

3.3.1 Manejo de las interacciones

En primer lugar, deben manejarse adecuadamente las interacciones que existen en un ambiente de social media, con el fin de abordar el problema de correspondencia entre el modelo de datos del dominio y la definición de evento tipada por RBox. En ese sentido, debe existir un componente de *software* que permite realizar un mapeo entre estos dos dominios y exportar hacia un medio persistente la información de las interacciones realizadas por los usuarios.

Este componente debe conectarse al ambiente de social media a través del método más "limpio" posible. Es decir, si el ambiente cuenta con una API disponible para acceder a su información, este método debe ser preferido por sobre otros, como por ejemplo adoptar una estrategia de scrapping.

La estrategia de extracción debe considerar la mayor cantidad de información que las políticas de privacidad de los ambientes permitan y debe tender a completar lo más posible el modelamiento de datos utilizado por RBox, y que está basado en el marco otorgado por la 3-Ontology.

Por otro lado, se deben definir las interacciones del dominio de extracción en RBox. Esto pasa por identificar, tipificar y crear una jerarquía de clases de manera tal que cada una de las interacciones que ocurren en un dominio determinado, deben corresponder, en RBox, a una clase que herede de Evento.

3.3.2 Detección de comunidades

En segundo lugar, debe existir un componente de *software* que otorgue funcionalidad de detección de comunidades. Este componente debe ser construído de manera tal que su integración con RBox no sea invasiva, permitiendo su reemplazo por otros componentes similares, según se requiera.

Como parte de la integración, debe ser abordado el problema de encapsulación de los distintos eventos definidos en RBox, con el objetivo de generar un grafo de interacciones. Este grafo de interacciones debe ser genérico y no debe importar el evento que propicie su construcción. Luego, el componente de *software* debe ser capaz de interpretar este grafo de interacciones y debe proveer diversos mecanismos para realizar detección de comunidades. La respuesta de este componente debe

ser un grafo con sus comunidades explicitadas.

3.3.3 Persistencia de comunidades

En tercer lugar, debe existir un componente de *software* que maneje la persistencia de las comunidades detectadas. Entre las responsabilidades de este componente está la de conectarse con el componente de detección de comunidades, incorporando a RBox un servicio que faculte a todo sistema de recomendación desarrollado con el *framework*, el uso de la detección de comunidades.

Uno de los principales desafíos que aborda este componente es el de implementar el mecanismo de *community cache* descrito en la sección anterior. Este mecanismo se debe detonar cuando comiencen las operaciones que tengan relación con la persistencia de las comunidades.

La persistencia ocurre al momento de interpretar y transformar desde el grafo con comunidades explicitadas, que es salida del componente de detección de comunidades, hacia el modelo de comunidades que maneja RBox. Es en este momento cuando deben añadirse operaciones al DAO de RBox para que sea capaz de detectar comunidades con un conjunto de usuarios idénticos.

Además, este componente debe integrarse a RBox implementando todos aquellos servicios de DAO que tengan relación con el manejo de las comunidades y que en su primera versión se han dejado sin implementar. Esto con el objetivo de completar el conjunto de herramientas que el API de RBox provee a los desarrolladores de sistemas de recomendación.

3.4 INTEGRACIÓN DE COMPONENTES

Los componentes descritos anteriormente deben integrarse de diversas maneras. La Figura 3-4 muestra un diagrama que representa conceptualmente esta integración, entre paréntesis están descritos los nombres de referencia que aparecen en la Figura. En primer lugar está el componente (*mapping component*), que resuelve la extracción de interacciones y sus transformaciones a eventos.

Este componente extrae información desde el ambiente de social media (*social media environment*) a través del mecanismo más limpio como ya se mencionó anteriormente y persiste estas interacciones en un medio persistente, cuyo esquema está en regla con el definido por la 3-

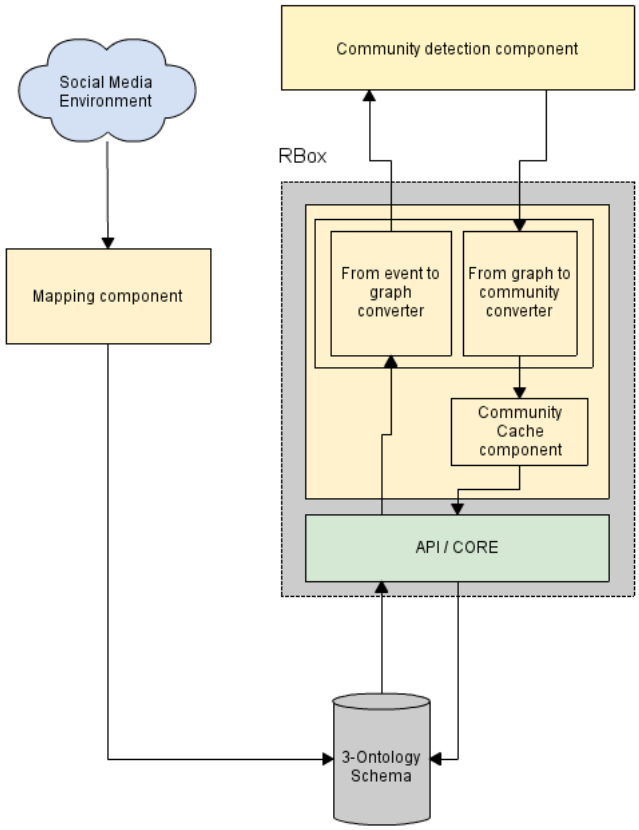

Figura 3-4: *Diagrama conceptual de integración de los componentes de software.*

Ontology (3-Ontology *Schema*).

Con el medio persistente ya con eventos, y suponiendo que se está en proceso de recomendación, el componente que convierte los eventos a grafos (*From event to graph converter*), a través de las operaciones de DAO definidas en el API y el CORE de RBox, extrae aquellos eventos que sean de interés y construye un grafo de interacciones.

Posteriormente, este grafo de interacciones es enviado al componente de detección de comunidades (*Community detection component*), que interpreta el grafo, lo procesa y detecta las comunidades mediante un método que le es definido.

Este último componente retorna la información del grafo con comunidades hacia un componente de transformación de grafos a comunidades (*From graph to community converter*) que interpreta este grafo en el esquema definido por la 3-Ontology para las comunidades.

Luego, se procede a las operaciones de persistencia de las comunidades detectadas, por lo que se da inicio a la intervención del componente que maneja la *community cache*, quién filtra las comunidades ya persistidas. Esto permite que, luego y nuevamente por operaciones del DAO, se

persistan aquellas comunidades detectadas en la ejecución actual del algoritmo de recomendación. Finalmente, se prosigue con la ejecución del algoritmo con normalidad.

3.5 RESUMEN

En este capítulo se han abordado los desafíos y problemáticas derivadas del cumplimiento del objetivo general de este trabajo de tesis. Se han descrito aquellos componentes que deben ser incorporados a RBox con la finalidad de proveer una plataforma tecnológica que permite probar la hipótesis propuesta. En los capítulos siguientes se describe con mayor detalle a los componentes de *software* diseñados en este capítulo, con sus implementaciones concretas, arquitecturas y paradigmas utilizados.

CAPÍTULO 4. SERVICIO PARA LA DETECCIÓN DE COMUNIDADES

En este capítulo se describe el servicio de detección de comunidades que se desarrolló en este trabajo de tesis. En primer lugar, se caracteriza y describe a la arquitectura y sus componentes. Posteriormente, se declaran los algoritmos y endpoints utilizados por el servicio. Finalmente, se muestra un ejemplo de funcionamiento con un problema clásico de la literatura.

4.1 CARACTERIZACIÓN DEL SERVICIO

En la presente sección, se realiza una caracterización del servicio que se ha construido, desde un punto de vista tanto tecnológico e ingenieril.

4.1.1 Descripción general

Una de las principales dificultades de este trabajo de tesis ha sido la de añadir a RBox la capacidad de detectar comunidades de una manera limpia. Esto implica que dicha añadidura no debe suponer, por una parte, una mayor reingeniería de los componentes que ya se han definido en la herramienta y por otra, una estricta dependencia de RBox con respecto al componente de detección de comunidades. En resumen, la nueva pieza de *software* que se ha incorporado debe cumplir con las siguientes reglas:

I No debe implicar modificaciones en el código de los componentes ya definidos.

II La pieza debe ser una alternativa válida para detectar comunidades.

III La pieza no debe impedir su reemplazo por otra con mayores prestaciones o mucho más específica.

Siguiendo estos principios definidos para la aplicación es que se ha tomado como alternativa una arquitectura orientada a servicios, ya que permite flexibilidad al integrarse con sistemas "legados" (Bianco, Kotermanski, Merson, 2007). Continuando con la misma línea de pensamiento, al momento de definir que tipo de servicio conviene disponibilizar, si un servicio Web SOAP[1] convencional o un servicio de tipo REST[2], se ha probado y evidenciado que los servicios de tipo REST son mayormente adecuados cuando se requieren integraciones estratégicas con otras aplicaciones, cuando el performance importa (Guinard, Ion, Mayer, 2012). Luego, el servicio construido es de tipo REST.

Ahora bien, de las alternativas posibles para proveer un servicio que sea eficaz detectando comunidades: construir una librería con distintos algoritmos o seleccionar y adaptar una existente, se ha optado por la segunda alternativa, ya que existe una librería llamada igraph (Csárdi Nepusz, 2006), construída bajo C, que provee diferentes estrategias y algoritmos típicamente utilizados para el análisis de grafos y en específico, la detección de comunidades. Esta librería cuenta con un encapsulamiento de la implementación en C hacia Python, lo que permite comunicar sus prestaciones con un *framework* que soporte la construcción de servicios Web de tipo REST.

4.1.2 Arquitectura del Servicio

Como primer antecedente, y para contextualizar, es necesario mencionar la arquitectura que se ha definido para la construcción del servicio. La Figura 4-1 muestra un diagrama con los componentes definidos y sus interacciones. La descripción de cada uno de ellos es la siguiente:

- ***Client*:** es una aplicación, servicio o componente de *software* que requiere un servicio de análisis de comunidades y, en concreto, un servicio de detección de comunidades. Este cliente debe preocuparse de encapsular la información de manera conveniente a la que el servicio espera

[1]SOAP es el acrónimo de *Simple Object Access Protocol*, que define un *stack* de componentes para el intercambio de información entre aplicaciones.

[2]REST es el acrónimo de *Representaton State Transfer*, que engloba y define una manera de acceder a recursos a través de la web únicamente utilizando el protocolo HTTP.

al momento de realizar la petición de servicio, así como también de traducir la respuesta del recién mencionado. En el diagrama, esta situación se refleja como Wrapped Data.

- ***Network Environment*:** representa al ambiente de red que separa al cliente y al servicio. Es muy relevante considerarlo debido a que, dependiendo del tamaño del problema, es posible que el ambiente del cliente no requiera uno de alta disponibilidad de recursos (a.k.a *cloud servers*). Sin embargo, el servicio puede abastecer a muchísimos clientes, por lo que puede requerir un ambiente escalable horizontalmente para su deployment, como por ejemplo la nube de *Amazon*.

- **Endpoints:** son las interfaces que expone el servicio para que los clientes se comunican con el. En el caso del servicio de detección de comunidades, cada endpoint implica un algoritmo diferente para realizar la detección de comunidades.

- **Servicio:** el servicio es el resultado de la unión de distintas tecnologías. En primer lugar, y para permitir la exposición de un servicio Web de tipo rest, se utiliza flask-restful[3]Sitio web oficial en http://flask-restful.readthedocs.org/en/latest.. Luego, para permitir la comunicación con igraph, se utiliza un componente que encapsula la comunicación con la mencionada librería, llamada python-igraph. Finalmente, para todo el procesamiento y disponibilización de algoritmos, se utiliza la librería igraph. La especificación tanto de python-igraph como de igraph está disponible en el sitio Web oficial [4].

4.2 ALGORITMOS UTILIZADOS

En esta sección se enuncian los algoritmos que se han publicado mediante el servicio de detección de comunidades y que son aquellos que la librería igraph ha implementado. La Tabla 4.1 muestra a cada uno de ellos y el endpoint en particular al que están sujetos.

[3]

[4]Sitio web oficial de igraph para python en http://igraph.org/python/.

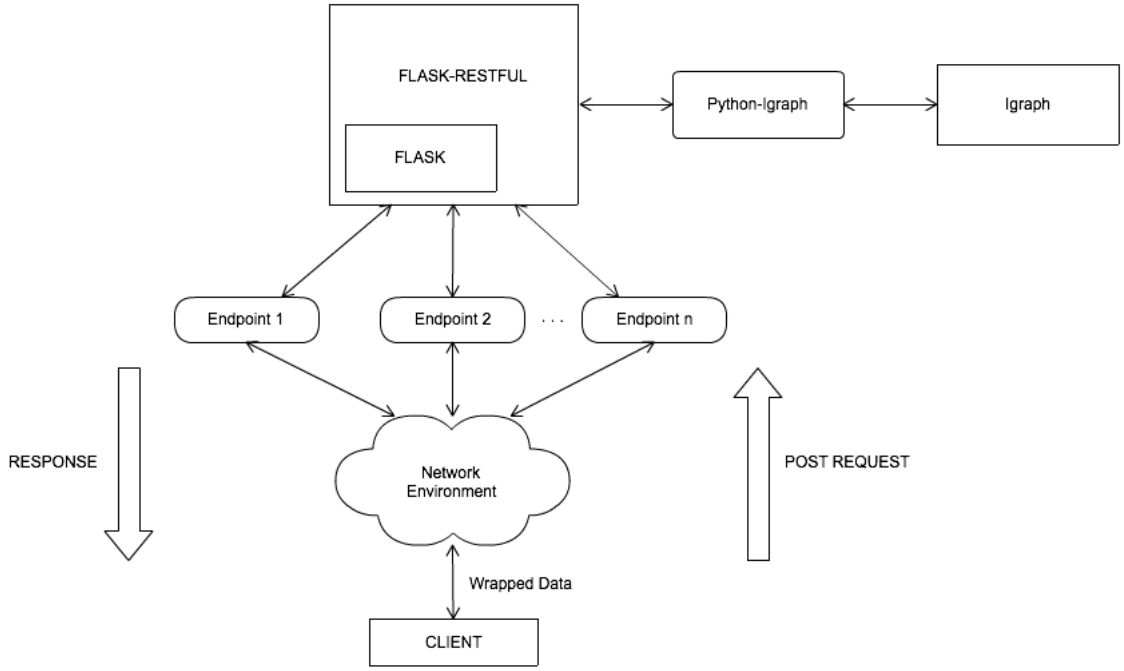

Figura 4-1: *Arquitectura del servicio de detección de comunidades.*

Tabla 4.1: Estrategias utilizadas para la detección de comunidades.

Algoritmo	**Endpoint (POST)**
Multilevel (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008)	/CommunityDetection/*Multilevel*
Label Propagation (Raghavan, Albert, & Kumara, 2007)	/CommunityDetection/LabelPropagation
Leading Eigenvector (Newman, 2006)	/CommunityDetection/LeadingEigenvector
Infomap (Rosvall & Bergstrom, 2008)	/CommunityDetection/Infomap
Spinglass (Traag & Bruggeman, 2009)	/CommunityDetection/Spinglass

4.3 EJEMPLO DE FUNCIONAMIENTO

En esta sección se muestra el funcionamiento del servicio de detección de comunidades a partir de un ejemplo clásico en la literatura, que es detectar las comunidades en el club de karate

de Zachary (Zachary, 1977), que tiene 34 nodos, 78 aristas y el máximo componente conectado es de 34 nodos. Para esto, se han destinado endpoints (ver Tabla 4.2) del servicio de manera tal que el ejemplo sea parte de la aplicación:

Tabla 4.2: Servicios que implementan el manejo de la detección de comunidades en el Club de Karate de Zachary.

Descripción	**Endpoint (GET)**
JSON que representa el grafo del problema del club de karate de Zachary.	/KarateClub
JSON que representa a las comunidades detectadas en el *dataset* del club de karate de Zachary utilizando el mismo algoritmo que el endpoint /CommunityDetection/*Multilevel*.	/KarateClub/Communities/*Multilevel*
JSON que representa a las comunidades detectadas en el *dataset* del club de karate de Zachary utilizando el mismo algoritmo que el endpoint /CommunityDetection/LabelPropagation	/KarateClub/Communities/LabelPropagation
JSON que representa a las comunidades detectadas en el *dataset* del club de karate de Zachary utilizando el mismo algoritmo que el endpoint /CommunityDetection/LeadingEigenvector	/KarateClub/Communities/LeadingEigenvector
JSON que representa a las comunidades detectadas en el *dataset* del club de karate de Zachary utilizando el mismo algoritmo que el endpoint /CommunityDetection/Infomap	/KarateClub/Communities/Infomap
JSON que representa a las comunidades detectadas en el *dataset* del club de karate de Zachary utilizando el mismo algoritmo que el endpoint /CommunityDetection/Spinglass	/KarateClub/Communities/Spinglass

La Figura 4-2 muestra el grafo que se genera al realizar el parsing correspondiente luego

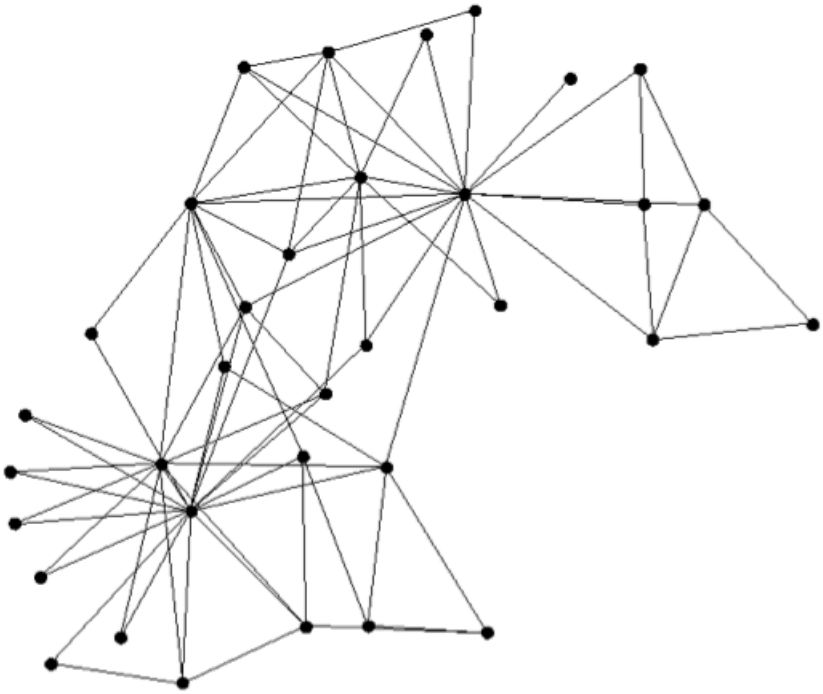

Figura 4-2: *Grafo que representa al Club de Karate de Zachary.*

de hacer el request a /KarateClub. Cuando se realiza el request a /KarateClub/Communities para detectar las comunidades que existen en el *dataset* de Zachary el resultado, para cada algoritmo, se muestra desde la Figura 4-3 hasta la Figura 4-7. La Tablas 4.3 a 4.7 describen las comunidades detectadas por extensión, para cada algoritmo:

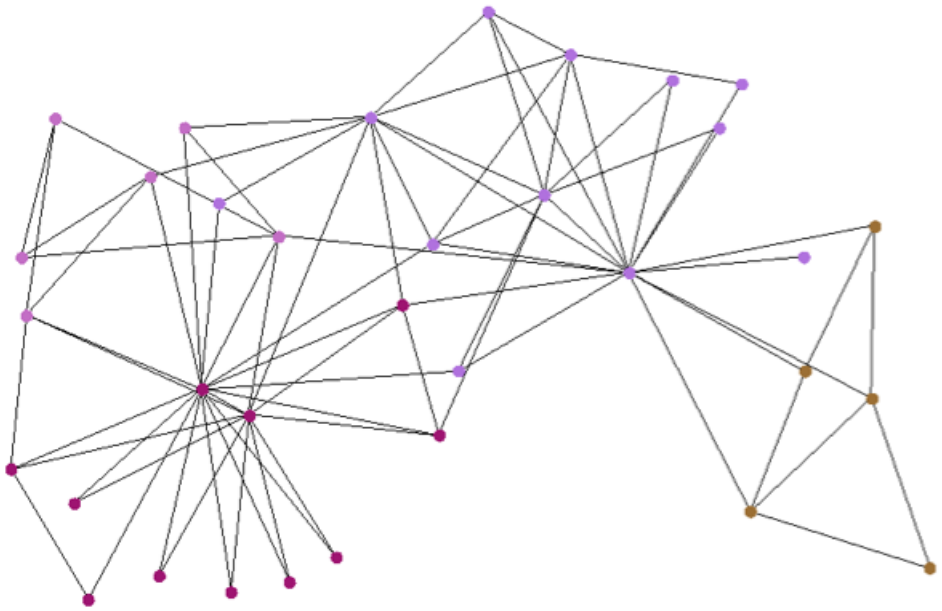

Figura 4-3: *Comunidades detectadas en el Club de Karate mediante el método Multilevel.*

Tabla 4.3: Definición por extensión de las comunidades detectadas en el Club de Karate mediante el método *Multilevel*.

Nº Comunidad	**Nodos**
1	5, 6, 7, 11, 17
2	1, 2, 3, 4, 8, 10, 12, 13, 14, 18, 20, 22
3	24, 25, 26, 28, 29, 32
4	9, 15, 16, 19, 21, 23, 27, 30, 31, 33, 34

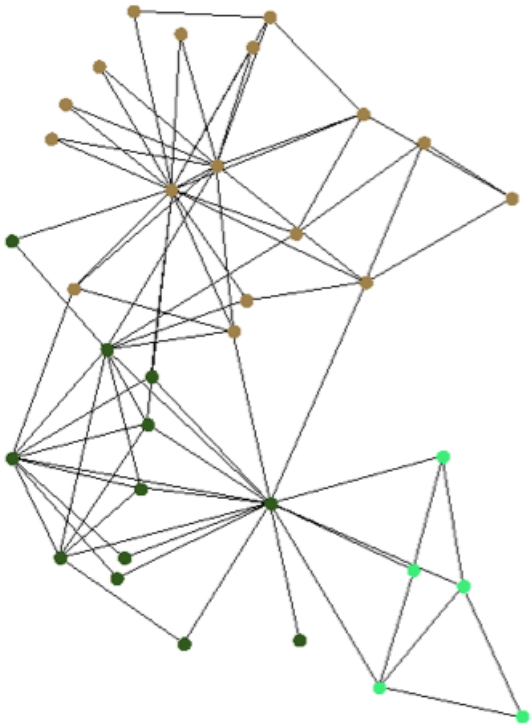

Figura 4-4: *Comunidades detectadas en el Club de Karate mediante una ejecución del método Label Propagation.*

Tabla 4.4: Definición por extensión de las comunidades detectadas en el Club de Karate mediante una ejecución del método *Label Propagation*.

Nº Comunidad	**Nodos**
1	1, 2, 3, 4, 8, 10, 12, 13, 14, 18, 20, 22
2	5, 6, 7, 11, 17
3	9, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

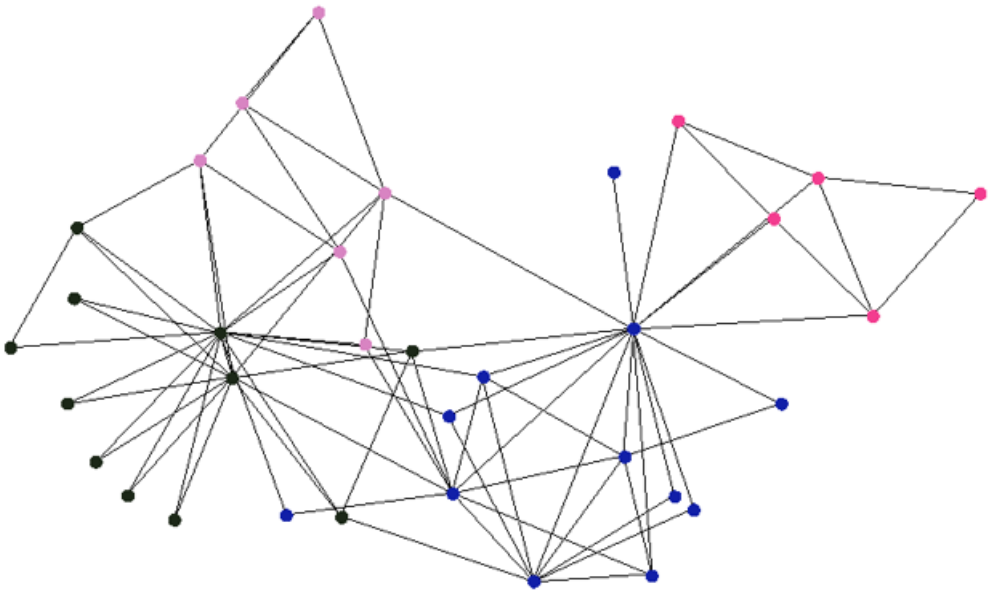

Figura 4-5: *Comunidades detectadas en el Club de Karate mediante el método Spinglass.*

Tabla 4.5: Definición por extensión de las comunidades detectadas en el Club de Karate mediante el método *Spinglass*.

Nº Comunidad	**Nodos**
1	24, 25, 26, 28, 29, 32
2	5, 6, 7, 11, 17
3	1, 2, 3, 4, 8, 10, 12, 13, 14, 18, 20, 22
4	9, 15, 16, 19, 21, 23, 27, 30, 31, 33, 34

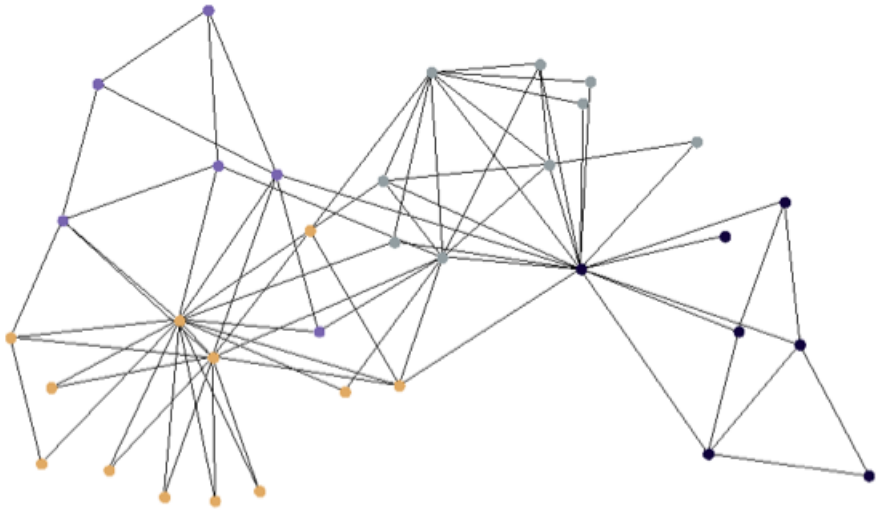

Figura 4-6: *Comunidades detectadas en el Club de Karate mediante el método LeadingEigenvector.*

Tabla 4.6: Definición por extensión de las comunidades detectadas en el Club de Karate mediante el método *LeadingEigenvector*.

Nº Comunidad	**Nodos**
1	1, 5, 6, 7, 11, 12, 17
2	9, 10, 15, 16, 19, 21, 23, 27, 30, 31, 33, 34
3	2, 3, 4, 8, 13, 14, 18, 20, 22
4	24, 25, 26, 28, 29, 32

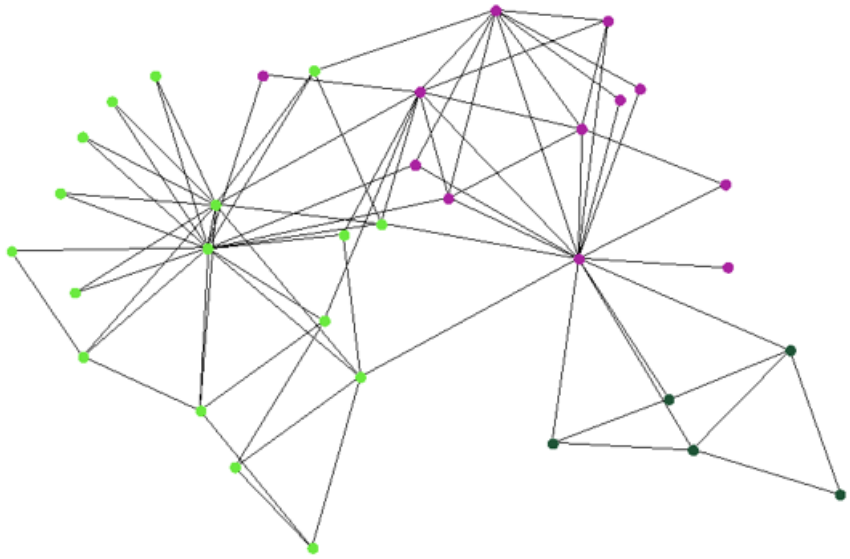

Figura 4-7: *Comunidades detectadas en el Club de Karate mediante el método Infomap.*

Tabla 4.7: Definición por extensión de las comunidades detectadas en el Club de Karate mediante el método *Infomap*.

Nº Comunidad	**Nodos**
1	1, 2, 3, 4, 8, 10, 12, 13, 14, 18, 20, 22
2	5, 6, 7, 11, 17
3	9, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

4.4 RESUMEN

En este capítulo se presentó el servicio de detección de comunidades que se utilizará RBox. Se han descrito los fundamentos mediante los cuales se tomó la decisión de seleccionar tecnologías y

componentes. Posteriormente, se ha descrito la arquitectura del servicio componente a componente. Finalmente, se han enunciado los endpoints que proveerá el servicio y, mediante un ejemplo, se ha mostrado su funcionamiento detectando comunidades para un mismo *dataset* y con distintos algoritmos.

CAPÍTULO 5. ARQUITECTURA Y DISEÑO DE COMPONENTE PARA RBOX

En este capítulo se tratará la arquitectura y diseño de un componente para RBox que permitirá comunicarse apropiadamente con el servicio de detección de comunidades ya descrito en el Capítulo 3. En primer lugar, se realizará una descripción de la arquitectura del componente, incluyendo su inclusión en RBox. Luego, se mostrará el diseño de clases con los componentes que ha sido necesario construir y se describen las razones de inclusión de cada uno de ellos. Posteriormente, se tratan los impactos que este componente ha tenido sobre las distintas capas de RBox. Finalmente, se detalla la comunicación entre componentes y como se detona la detección de comunidades al recomendar.

5.1 DESCRIPCIÓN ARQUITECTURAL

En esta sección se describe, en líneas generales, la arquitectura de RBox y su integración con las nuevas piezas de *software* que permitirán consumir el servicio de detección de comunidades descrito anteriormente. La Figura 5-1 muestra los componentes utilizados y sobre los cuáles se ha diseñado.

En primer lugar, se requiere de un mecanismo de persistencia para la información requerida y generada por el sistema:

- **3-Ontology *Database Schema*:** es un esquema relacional basado en el modelo conceptual de la 3-Ontology. Su función es persistir y proveer información, desde y hacia RBox, respecto de los distintos eventos que ocurran en la(s) red(es) social(es) que se estén monitoreando, con el objetivo final de generar una recomendación.

El siguiente componente es RBox, que se compone de distintos submódulos. Originalmen-

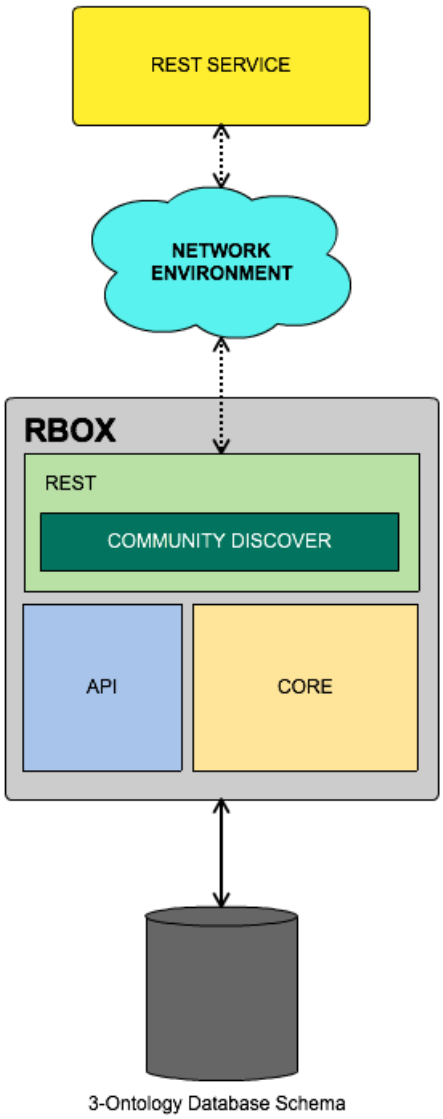

Figura 5-1: *Diagrama que representa la arquitectura formada por los componentes de RBox.*

te, y antes del aporte de este trabajo de Tesis, solamente existían dos submódulos:

- **API:** es un conjunto de interfaces que definen las propiedades estándar de todos los objetos que quieran extender, utilizar las propiedades de RBox o construir recomendadores. En este submódulo también se incluye la definición de las interfaces que representan a los contenedores de sentido y la lógica de la 3-Ontology.
- **CORE:** una implementación del API basada en distintos componentes. En este submódulo se implementa la capa *Data Access Object* (DAO) a través de componentes que utilizan JDBC. Además, se incluyen implementaciones base para distintos tipos de eventos y recomendadores.

Luego de finalizado este trabajo de Tesis, se ha agregado otro módulo a RBox, que tiene que ver con el manejo de servicios REST.

- **REST:** un submódulo de RBox que permite realizar request de tipo REST a distintos servicios. En particular, se ha incluído un submódulo llamado *Community Discover*, que es una implementación de un cliente REST para comunicarse con el servicio de detección de comunidades.
- ***Community Discover*:** un submódulo que se preocupa de proveer desde y hacia RBox un mecanismo de consumo, procesamiento y encapsulación de la información necesaria para el manejo de las comunidades detectadas por el servicio de detección de comunidades. La persistencia de las comunidades detectadas se realiza a través de la capa DAO, que existe en el *core* de RBox.

Como se puede apreciar en la Figura 5-1, es el componente REST de RBox el que se comunica, a través de un *Network Environment* acorde a la realidad de la aplicación, con un servicio REST y, en este caso en particular, la comunicación se realiza con el servicio de detección de comunidades. Ahora, es necesario describir con un mayor detalle el componente *Community Discover*, y las piezas que se han creado para su integración con RBox.

5.2 DISEÑO DE CLASES

En esta sección se describe detalladamente el componente *Community Discover* mencionado en la arquitectura. Primero cada uno los submódulos definidos y luego, para cada uno de ellos, una descripción clase a clase.

La Figura (5-2) muestra un diagrama con la comunicación entre los componentes de *Community Discover*. Se ha incluído el Core de RBox para ilustrar que el recién mencionado:

I Solo se ve impactado por la respuesta del servicio de *Community Discover*.

II Puede consumir únicamente una capa de métodos utilitarios de conversión de datos llamada Converter.

5.2.1 Model

Este componente define un *Data Transfer Object* (DTO), el cual es un modelo que será utilizado en los proceso que involucren al servicio REST. La Tabla 5.1 muestra las clases que contiene este componente:

Tabla 5.1: Clases involucradas en la composición de *Model*.

Clase	**Descripción**
CommunityModel	Clase que define un modelo de comunidad, a partir de un arreglo de *NodeModel*.
EdgeModel	Clase que define un modelo de arista (en inglés, *edge*), a partir de un origen y un destino, representados por un ID de tipo Long.
NodeModel	Clase que define un modelo para un nodo (en inglés, *node*), a partir de un ID de tipo Long y un String representativo como nombre.

5.2.2 Wrapper

Este componente provee un medio de encapsulación de los datos que se envían desde y hacia el servicio Web. Puede ser descrito como un contenedor de información que se envía o recibe

dependiendo de si es un request al servicio o un response del servicio. La Tabla 5.2 muestra las clases que contiene este componente:

Tabla 5.2: Clases involucradas en la composición de *Wrapper*.

Clase	**Descripción**
CommunityWrapper	Esta clase define un contenedor de comunidades a partir de un arreglo de objetos *CommunityModel*. Es como se traduce la respuesta del servicio de detección de comunidades.
GraphWrapper	Esta clase define un contenedor de grafo a partir de una lista de elementos *EdgeModel* y una lista de elementos *NodeModel*. Es como se envía la solicitud al servicio de detección de comunidades.

5.2.3 *Client*

Este componente consiste en una clase cuyo propósito es realizar el request correspondiente al servicio rest, a partir de un grafo encapsulado de tipo *GraphWrapper* y también de persistir los resultados del response a través del uso de *CommunityDAO* existente en el CORE de RBox. La Tabla 5.3 muestra las clases de este componente:

Tabla 5.3: Clases involucradas en la composición de *Client*.

Clase	**Descripción**
Community Discover Client	Esta clase define a un cliente que permite conectarse y consumir el servicio de detección de comunidades, dado un *GraphWrapper* como request, recibe un *CommunityWrapper* como response.
GraphWrapper	Esta clase define una versión particular para un cliente del servicio de detección de comunidades, ya que permite obtener y recrear el ejemplo de uso de los algoritmos de los endpoints del servicio, de tal forma como se mostró en el Capítulo 4. Al ser solo servicios GET, solo envía una solicitud a una URL en el request. El response del servicio es un objeto *CommunityWrapper*.

5.2.4 Service

Este componente consiste en una especialización de los servicios de la 3-Ontology. En este caso en particular, como *Portrait* (del español Retrato, ver Capítulo 2) es el servicio de la 3-Ontology que tiene que ver con las comunidades, se provee una especialización de el. La especialización del servicio también tiene que ver con adoptar una estrategia inteligente de persistencia en dos niveles. Este concepto ha sido denominado *community cache*. Lo anterior se resume en la Tabla 5.4.

El mecanismo de *community cache*, en un primer nivel, permite controlar si la detección y persistencia se realiza solamente cuando el usuario activo no tiene comunidades asociadas en el modelo de la 3-Ontology. En el segundo nivel, tiene que ver si para la persistencia se toma en cuenta el hecho de que la comunidad ya forma parte del modelo de la 3-Ontology, considerando el conjunto de usuarios que la forman.

Tabla 5.4: Clases involucradas en la composición de *Service*.

Clase	**Descripción**
CommunityExtension*Portrait*	Esta clase hereda de Generic*Portrait* que existe en el Core de RBox y resume un servicio que permite detectar y persistir comunidades para un usuario, de manera forzada o bien, cuando no existen explícitamente en el esquema relacional de datos. El manejo de una estrategia de *community cache* para persistir comunidades también está definido en este servicio.

5.2.5 Converter

Este componente permite realizar transformaciones desde un tipo de dato a otro. Es utilizado en *Community Discover* para transformar una serie de *Event*os del esquema de 3-Ontology a un *GraphWrapper*, que es el formato aceptado por el servicio de detección de comunidades. Además, permite realizar otro tipo de transformaciones, como por ejemplo, transformar un *GraphWrapper* en un objeto *Graph* para visualizar su estructura y conexiones, tal como se realizó en el ejercicio del Club de Karate de Zachary. La Tabla 5.5 describe las clases que forman esta pieza.

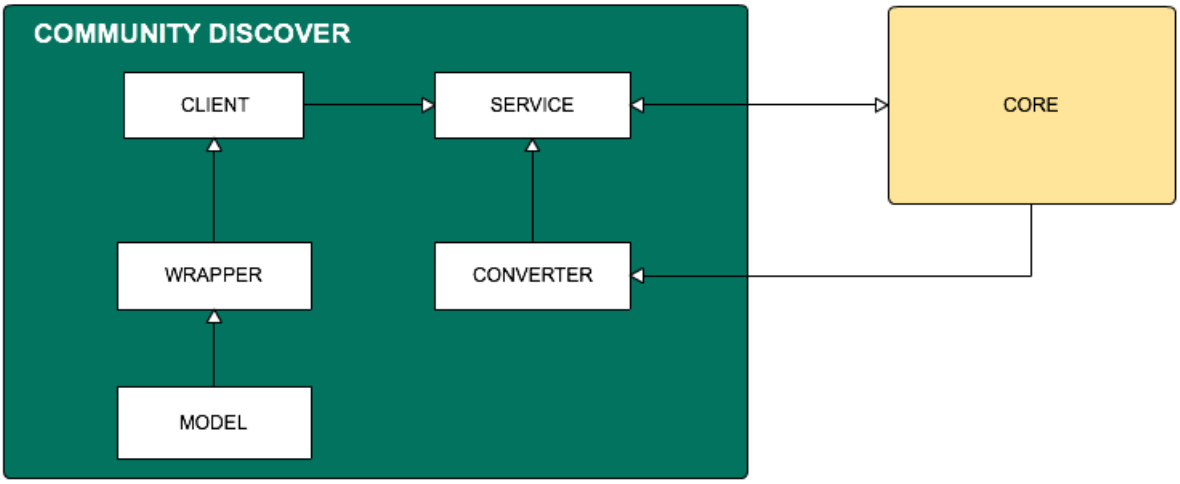

Figura 5-2: *Diagrama con la comunicación entre los componentes de Community Discover*

Tabla 5.5: Clases involucradas en la composición de *Converter*.

Clase	**Descripción**
EventWrapperConverter	Esta clase utiliza el patrón Generics de JAVA para logar, dado una clase que herede de *Event* y una clase que herede de *Item*Dao, convertir instancias de *Event* en un *GraphWrapper* con tal de ser enviado al servicio de detección de comunidades.
GraphWrapperConverter	Esta clase se encarga de realizar una transformación de un objeto *GraphWrapper* en un objeto *Graph* específico de una API llamada *GraphStream*[1]. Esto permite, entre otras cosas, renderizar y visualizar el grafo resultante del proceso de *Wrapping*.

Ya con los componentes descritos, es necesario analizar el impacto que esto tiene en el modelo actual de RBox, repasando capa a capa las modificaciones que han sido necesarias para dar soporte, sobre todo, a la persistencia de datos en un esquema relacional.

5.3 IMPACTO EN EL MODELO ACTUAL

En esta sección se resumen los cambios que se han realizado en el Core de RBox para permitir el funcionamiento del ciclo de detección y persistencia de comunidades.

5.3.1 *Event*os

Se han añadido dos tipos de eventos a RBox: mentioning y *replying*. Ambos son necesarios para contextualizar la herramienta a la red social que se analizará en el capítulo siguiente y poseen un valor de tipo String. Se han añadido versiones mutables e inmutables de ambos eventos.

5.3.2 DAO

Se han implementado la interfaz *CommunityDAO* para el manejo de comunidades con esquemas relacionales a través de JDBC. Además, RBox no tenía implementado un mecanismo de persistencia en esquemas relacionales y solamente estaba enfocado en consumir información de este medio. Por lo tanto, para poder persistir las comunidades detectadas, ha sido necesario añadir esta funcionalidad y preparar statements que faciliten el proceso de insertar comunidades y asociar usuarios a las comunidades a las que pertenecen. Se ha creado un DAO que facilite el acceso a los *Item*s, ya que son parte fundamental de la generación de los *GraphWrapper*.

5.3.3 Extensions

Se ha creado un nuevo módulo en RBox, que tiene que ver con el manejo de extensiones de funcionalidades. El funcionamiento está basado en transformaciones y encapsulamientos de datos, por lo que toda extensión, detección de comunidades incluída, debe definir sus propios servicios de transformación, que hereden de la interfaz Wrapper, desde y hacia 3-Ontology del esquema de datos en particular que cada extensión maneje.

En el caso particular de la extensión de detección de comunidades, se ha definido la interfaz CommunityExtension, que utiliza, mediante Generics, dos clases que heredan de la interfaz Wrapper para definir los servicios que esta extensión provee. En el segmento de código que muestra la FIgura 5-4 está la definición de esta interfaz. En *Community Discover*, la clase *Community Discover Client* es la que implementa esta interfaz, ya que es la que provee el servicio de detección y persistencia de comunidades.

```
public interface CommunityExtension <T extends Wrapper, U extends Wrapper>{

        public U communityDetection (T wrapper, CommunityDetectionAlgorithms cda);

        public void persistCommunities(U wrapper, CommunityDAO communityDAO);

        public void clean(CommunityDAO communityDAO);

        public String getSignature();
}
```

Figura 5-3: *Código que muestra la interfaz que debe implementar todo servicio de detección de comunidades.*

Como se ha descrito, las modificaciones en el Core de RBox son menores y el mayor impacto y la lógica de los servicios y procesos reside en el componente *Community Discover*. Luego, es hora de describir el proceso completo de comunicación entre los componentes mencionados en este capítulo.

5.4 COMUNICACIÓN ENTRE COMPONENTES

En esta sección se detalla el proceso de comunicación con todos los componentes ya descritos, a partir de un ejemplo de código que se muestra en la Figura **??**. En primer lugar, se obtiene una instancia de JDBCDAO y, con esta, se obtienen los eventos registrados en el medio persistente. Luego, se genera una instancia de GenericTrace utilizando como parámetro a los eventos obtenidos anteriormente. Posteriormente, se obtiene una lista de *User*History, agrupando los eventos para cada usuario.

```
JDBCDAO jdbcdao = getJDBCDAO();

List<Event> events = jdbcdao.getEvents();

GenericTrace genericTrace = new GenericTrace(events);

List<UserHistory<Event>> eventsByUser = genericTrace.getEventsByUser();

EventWrapperConverter<Mentioning, JDBCDAO> ewc = new EventWrapperConverter<>(jdbcdao);

for(UserHistory<Event> userHistory : eventsByUser){

                UserHistory<Mentioning> mentionFilter = userHistory.filter(Mentioning.class);

                ewc.addInteractions(mentionFilter);

}

CommunityExtensionPortrait portrait = new CommunityExtensionPortrait(
                 jdbcdao.getCommunities(),
                 new CommunityDiscoverClient(),
                 jdbcdao,
                 ewc.getGraphWrapper(),
                 CommunityDetectionAlgorithms.LABEL_PROPAGATION,
                 false
);

List<Community> communities = portrait.getCommunitiesForUser(2711128361);

System.out.println(communities.size());
```

Figura 5-4: *Código que muestra la interacción entre los distintos componentes del servicio.*

Una vez que se tiene el listado de *User*History es necesario construir el encapsulador de eventos a un *GraphWrapper*. Para esto se instancia un *EventWrapperConverter* utilizando como definición de Generics a la clase *Mentioning*, y a JDBCDAO. Esto implica que los eventos que serán encapsulados son aquellos que sean de tipo *Mentioning*. Posteriormente, y para cada evento por usuario, se filtran los eventos para obtener sólo a aquellos que correspondan a *Mentioning* y son esas interacciones las que se añaden al encapsulador. Cuando se tienen todos los eventos encapsulados, se crea el servicio *Portrait* que provee *Community Discover*. Para esto necesita:

1. Todas las comunidades existentes en el sistema.

2. Un cliente del servicio REST de detección de comunidades, para detectar comunidades según se configure el parámetro Nº6.

3. Una instancia de JDBCDAO para permitir la persistencia de comunidades, según corresponda.

4. Una instancia de *GraphWrapper* que encapsule los eventos, que será enviado como request al servicio de detección de comunidades.

5. El algoritmo que se quiere que el servicio utilice al momento de detectar comunidades.

6. Un valor booleano para determinar si se hace uso del primer nivel de *community cache*. En otras palabras, si las comunidades se detectan siempre (true) o solamente cuando no existan comunidades para un usuario (false).

Finalmente, se obtienen las comunidades para un usuario en particular. Si el usuario no tiene comunidades estas serán detectadas y se muestra la cantidad de comunidades a las que este usuario pertenece.

5.5 RESUMEN

En este capítulo se ha presentado el componente que se ha añadido a RBox para permitir la comunicación con el servicio de detección de comunidades presentado en el capítulo anterior. Además, se ha realizado una descripción componente a componente de cada uno de los submódulos de *Community Discover*, el módulo REST que permite detectar comunidades en base a los eventos almacenados en lógica 3-Ontology. Luego, se ha analizado el impacto que la añadidura de este componente ha tenido sobre el *core* de RBox ya desarrollado, identificando componentes restantes y añadiendo componentes nuevos para permitir el manejo de extensiones. Finalmente, se ha mostrado un ejemplo de integración de todos los componentes necesarios para realizar una detección de comunidades en base a un tipo de evento.

CAPÍTULO 6. EXPERIMENTACIÓN

En este capítulo se tratará el experimento realizado, ejecutado bajo todo el desarrollo tecnológico descrito en los Capítulos 3 y 4, que implica la manera de probar la hipótesis que este trabajo de tesis postula. En primer lugar, se describe el experimento realizado, considerando inputs, outputs y puntos de evaluación. Luego, se caracterizan los *dataset* utilizados, describiendo su dominio y cuantificando su tamaño. Finalmente, se muestran los resultados obtenidos.

6.1 DESCRIPCIÓN DEL EXPERIMENTO

En esta sección se describe el experimento que se ha realizado para cumplir con el objetivo general de este trabajo, que es detectar las comunidades implícitas que se forman a través de las interacciones sociales que contiene un *framework* orientado a eventos, con el objetivo de persistir su existencia y utilizarlas eficientemente como un antecedente ya conocido al momento de realizar nuevas recomendaciones.

Para describir el experimento, es necesario mencionar que, en el flujo de datos propuesto por RBox, al momento de generar una recomendación es posible también obtener comunidades de usuarios con intereses similares (ver Figura 6-1). Esta aseveración era una carencia en la versión anterior de RBox y por esto se han generado dos componentes de *software*, que ya han sido descritos. El tema fundamental pasa por persistir la existencia de las comunidades y utilizarlas como un antecedente que haga más eficiente recomendaciones posteriores, por lo que el experimento buscará evaluar si persistir las comunidades detectadas en una primera recomendación acelera o no el proceso de recomendación en ejecuciones posteriores sobre un sistema de recomendación generado con RBox.

Con este objetivo es que se ha generado un *dataset* que será descrito en la sección

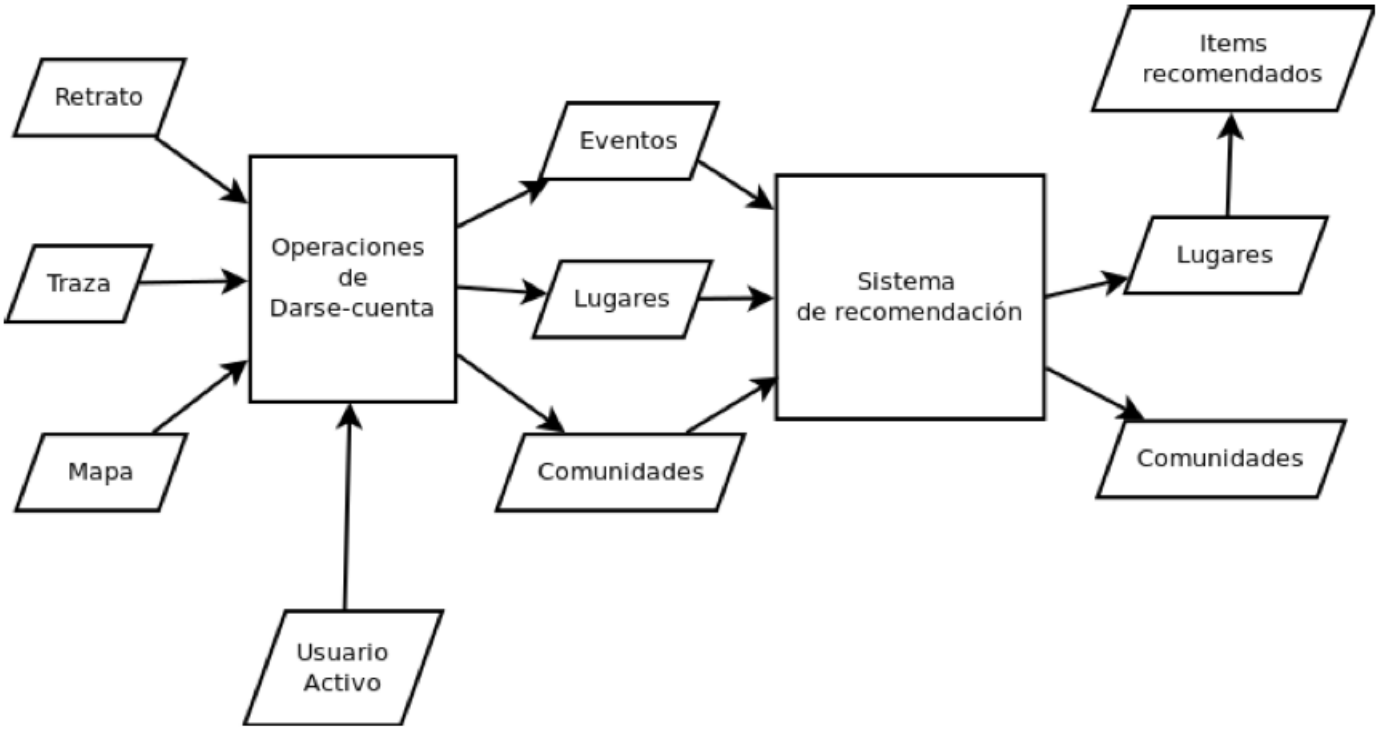

Figura 6-1: *Ciclo de recomendación en RBox.*

siguiente. Sobre él se generará una recomendación de *tags* considerando detección de comunidades, en dos escenarios: persistiendo comunidades detectadas y no persistiendo comunidades detectadas. La detección de comunidades será llevada a cabo mediante los distintos algoritmos que provee el servicio, ya descritos en el Capítulo 4. Luego, se comparan los tiempos de ejecución considerando ambos escenarios, para cada uno de los algoritmos de detección de comunidades.

6.2 CARACTERIZACIÓN DE *DATASET*

El *dataset* que se ha utilizado para llevar a cabo el experimento ha sido obtenido a partir de la red social *Twitter*. En concreto, se han generado un extractor de *tweets* utilizando Ruby como lenguaje de programación. El extractor utiliza una gema llamada *tweetstream* para acceder a la Streaming API de *Twitter*, permitiendo obtener el acceso a *tweets* de índole público, en tiempo real. Cada uno de los *tweets* es transformado a lógica 3-Ontology por el extractor y es persistido con la ayuda de un ORM llamado Sequel.

El *dataset* corresponde a una extracción dirigida de *tweets*, orientado a la Copa Mundial

de la Fifa Brasil 2014, para la que *Twitter* ha promovido el uso de hash*tags* particulares para cada país, como muestra la Tabla 6.1. Se ha hecho seguimiento a los comentarios que tengan que ver con los equipos sudamericanos que participan en la competición. La cuantificación de este *dataset* se puede apreciar en la Tabla 6.3. De cada uno de los *tweets*, interesa saber de las siguientes interacciones:

I ***Mentioning*:** es el acto de referencia explícita, en un tweet, a un usuario. Denota un mensaje directo hacia otro usuario o una invitación a compartir una conversación respecto a un tema.

II ***Tagging*:** es el acto de resaltar un concepto del mensaje utilizando un *hashtag*. Generalmente, marca un tópico relevante que conlleva un interés particular del usuario, que motiva y fundamenta el contenido que ha compartido en la red social.

III ***Replying*:** es el acto de replicar con un tweet la aseveración, comentario o mención de otro usuario.

Tabla 6.1: Hash*tags* particulares de cada país durante la copa mundial de la FIFA en *Twitter*.

Hashtag	**País**
CHI	Chile.
BRA	Brasil.
ARG	Argentina.
COL	Colombia.
ECU	Ecuador.

Tabla 6.2: Cuantificación del *dataset* de la copa mundial de la FIFA.

Medida	**Valor**
Event	445342
Item	23739
User	91489
Tweets	64120
Mentioning	92519
Tagging	349577
Replying	3246

Como primer análisis, se considera el top 3 de usuarios que han compartido más *tweets*, que incluyen menciones a otros usuarios (ver Tabla 6.3). Si se considera a estos usuarios, como activos, el grafo que representa sus interacciones se puede apreciar en las Figuras 6-2, 6-3 y 6-4,

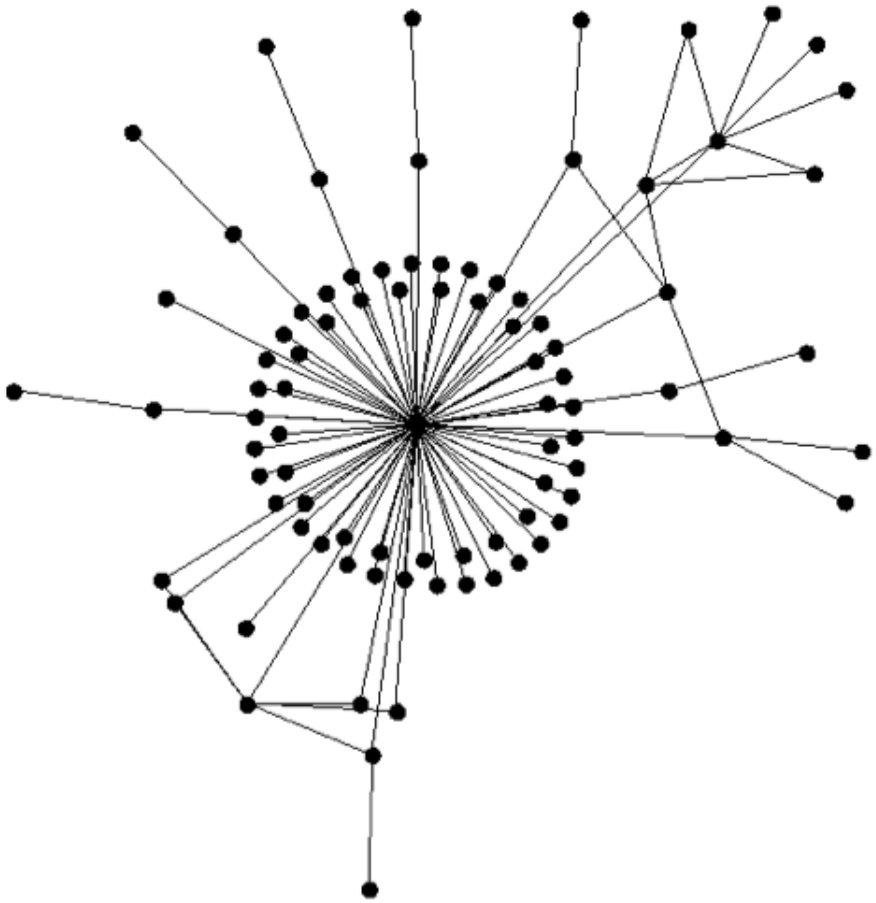

Figura 6-2: *Grafo de interacciones generado por el usuario 1.*

respectivamente. Si se realiza una detección de comunidades sobre el grafo de interacciones para estos usuarios, mediante, por ejemplo, del método *Multilevel* (ver Capítulo 3), las comunidades detectadas pueden apreciarse en las Figuras 6-5, 6-6 y 6-7, respectivamente.

Tabla 6.3: Top 3 de usuarios con mas *tweets* que consideran menciones a otros usuarios.

***User* (Alias)**	**Nº de Menciones**
539210296 (1)	95
2320149583 (2)	85
2463599637 (3)	69

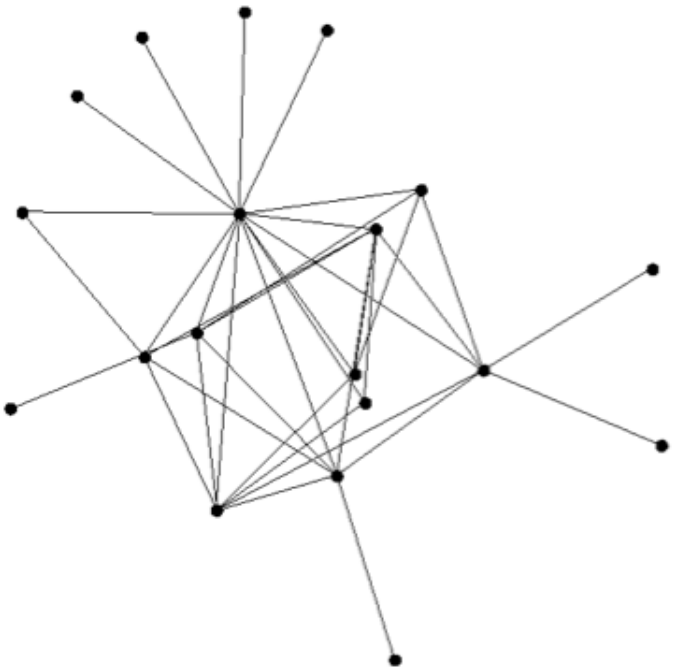

Figura 6-3: *Grafo de interacciones generado por el usuario 2.*

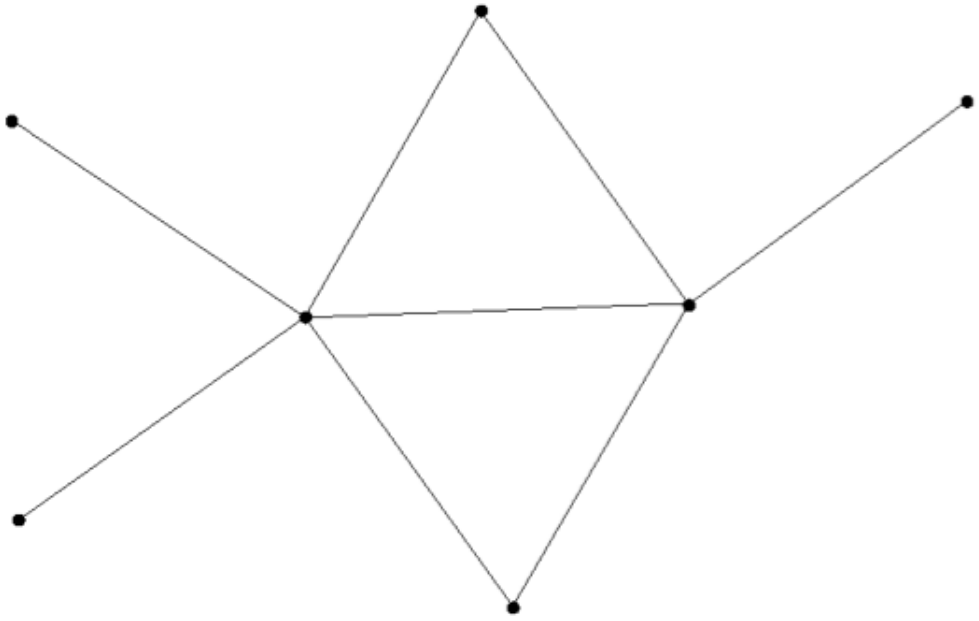

Figura 6-4: *Grafo de interacciones generado por el usuario 3.*

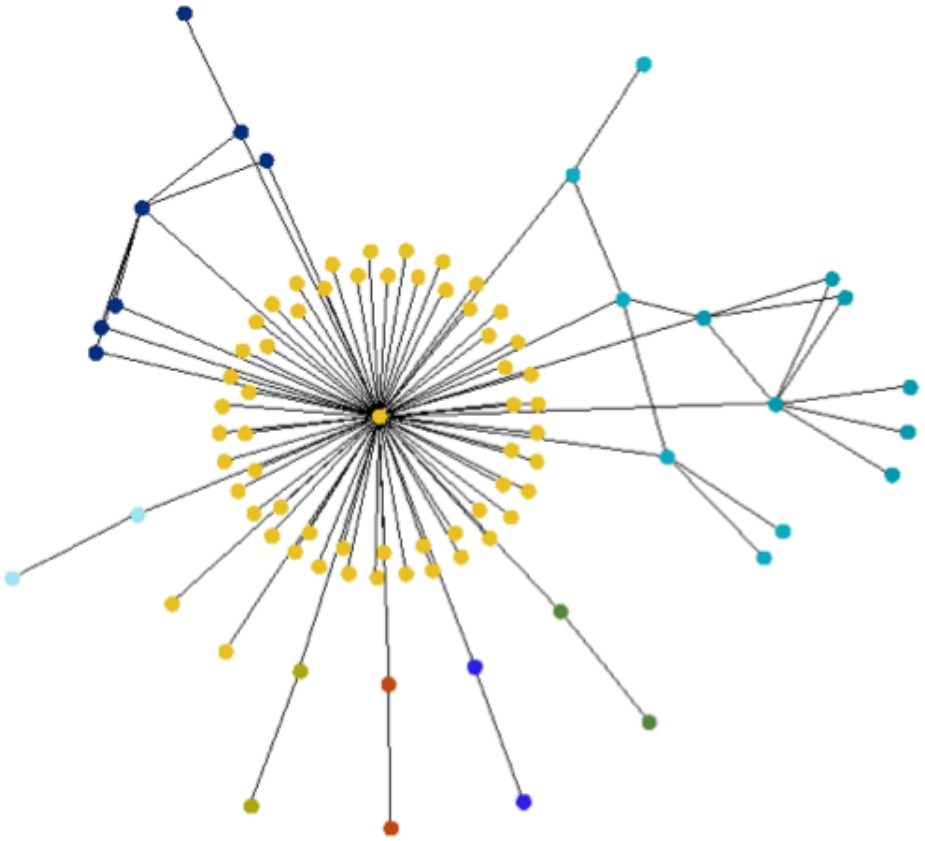

Figura 6-5: *Comunidades detectadas en base al grafo de interacciones generado por el usuario 1.*

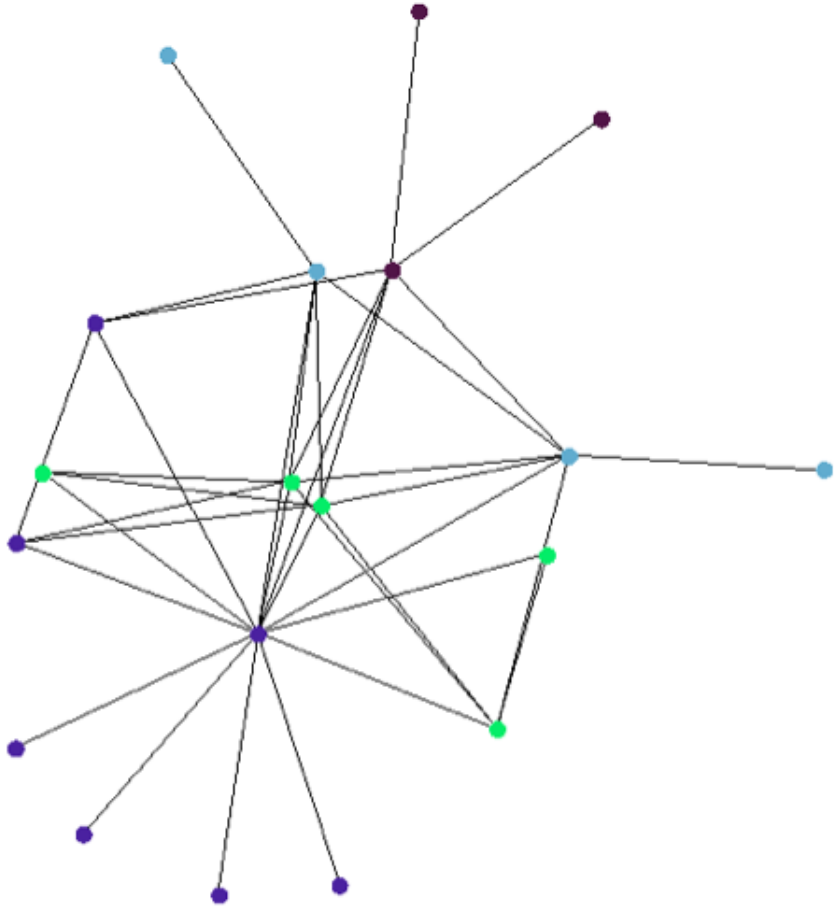

Figura 6-6: *Comunidades detectadas en base al grafo de interacciones generado por el usuario 2.*

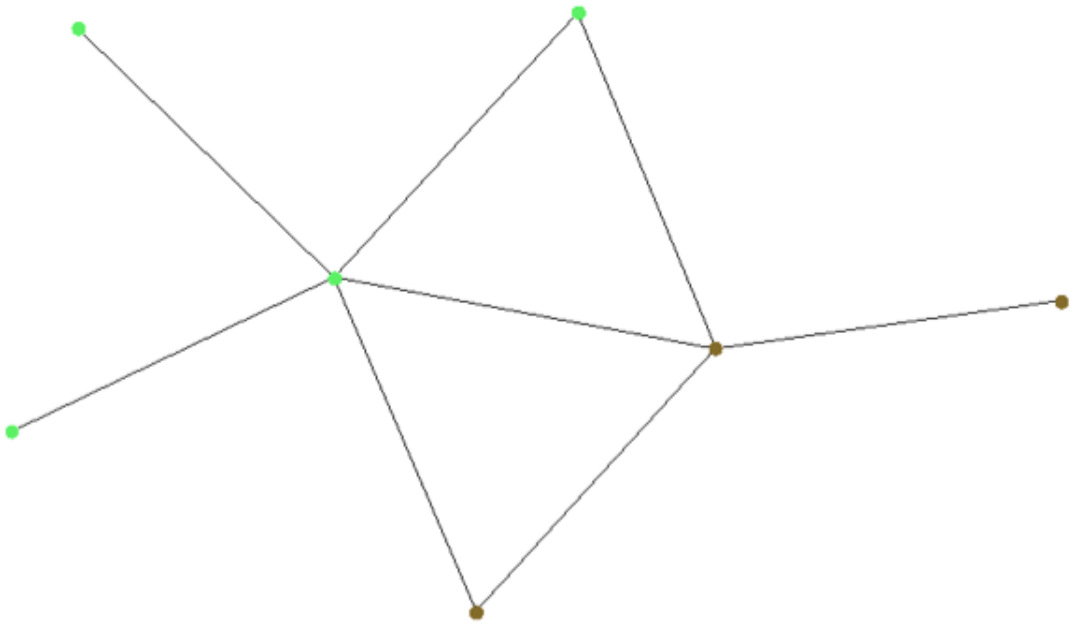

Figura 6-7: *Comunidades detectadas en base al grafo de interacciones generado por el usuario 3.*

6.3 RESULTADOS

Para experimentar con un proceso de recomendación utilizando comunidades, se ha implementado un algoritmo de recomendación de *tags*, cuyo objetivo es retornar los top N *tags* con mayor ocurrencia en una comunidad. Se han seleccionado dos usuarios (U), cuyos ids son 539210296 y 2320149583, para llevar a cabo la experimentación como usuarios activos. La Tabla 6.4 muestra los tiempos de ejecución requeridos para obtener la recomendación para los usuarios activos, dadas diez iteraciones (It), en las que se detectan comunidades, activando el primer nivel de *community cache*, mediante distintos métodos (Mi): M1 (*Multilevel*), M2 (LeadingEigenvector) y M3 (Infomap). La Tabla 6.5 muestra los tiempos de ejecución para un ejercicio similar, considerando detección de comunidades de manera forzada, al desactivar el primer nivel de *community cache*.

Los dos ejercicios recién mencionados consideran un ambiente sin comunidades en cada iteración, para evaluar la partida en frío de cada detección de comunidades. No obstante, un tercer caso de pruebas es con comunidades existentes en el medio persistente, evaluando el segundo nivel de *community cache*. En ese sentido la Tabla 6.6 muestra los tiempos de ejecución de aquel ejercicio.

Los resultados del primer caso de pruebas muestran que en la primera iteración el tiempo que se requiere para obtener la recomendación siempre es mayor que en las iteraciones posteriores. Esto es debido a que el proceso de recomendación parte "en frío", es decir, antes de recomendar debe, en primera instancia, detectar y persistir las comunidades para el usuario activo.

En iteraciones posteriores el tiempo necesario para la recomendación se reduce debido a que, como la detección de comunidades no es forzada y dado que existen comunidades para ese usuario activo, debido al segundo nivel de *community cache*, el *framework* no requiere realizar el proceso de detección y persistencia. Luego, los resultados del segundo caso de pruebas muestran, nuevamente, el fenómeno de partida "en frío". No obstante, los promedios de ejecución para todos los casos han aumentado en relación con el ejercicio anterior. Esto es debido a que, como se fuerza la detección de comunidades en cada iteración, el *framework* detecta nuevamente las comunidades para el usuario activo, con la importante salvedad de que, gracias al segundo nivel de *community cache*, no persiste aquellas que ya existen, eliminando duplicidad de comunidades.

Finalmente, el tercer caso de pruebas muestra que el tiempo de partida "en frío" se ha reducido, en general, para ambos usuarios. Esto último se confirma en la Tabla 6.7 donde se

muestran los tiempos de ejecución promedio para los tres ejercicios ya mencionados. Las Figuras 6-8 y 6-9 muestran respectivamente para cada usuario, el tiempo de ejecución promedio necesario para recomendar, considerando los tres experimentos. La reducción, en promedio, del tiempo necesario para obtener una recomendación para el usuario activo considerando el primer escenario y el último se ha reducido para todos los casos, incluso a pesar de que en el tercer caso existen comprobaciones constantes de igualdad de comunidades debido a la *community cache*.

Tabla 6.4: Tiempo en segundos que demora obtener una recomendación en el primer caso de pruebas. Que considera para cada *set* de iteraciones un medio persistente sin comunidades y un primer nivel de *community cache* activado.

	U1			**U2**		
It	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
1	15,46	19,22	16,09	37,86	23,83	22,01
2	2,68	4,42	4,77	12,19	9,74	9,18
3	4,55	3,74	3,65	9,92	7,24	7,09
4	4,55	5,01	3,41	7,42	5,51	5,84
5	4,61	5,48	4,81	4,73	5,33	5,56
6	4,98	5,28	5,43	5,09	5,94	5,17
7	5,08	5,16	7,43	4,66	5,07	4,90
8	4,60	7,01	7,26	4,44	4,12	5,20
9	4,57	4,86	5,80	4,56	8,03	4,53
10	5,24	4,65	4,14	4,26	5,76	3,44

Tabla 6.5: Tiempo en segundos que demora obtener una recomendación en el segundo caso de pruebas. Que considera para cada *set* de iteraciones un medio persistente sin comunidades y un primer nivel de *community cache* desactivado.

	U1			**U2**		
It	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
1	9,05	8,72	8,98	26,60	23,05	15,78
2	5,23	4,26	4,87	13,00	10,70	10,26
3	5,83	4,63	4,67	9,52	8,07	8,14
4	5,01	5,56	4,90	7,82	7,02	6,68
5	4,60	5,03	5,92	8,90	6,42	5,92
6	5,02	4,98	5,93	6,70	5,30	5,81
7	5,17	5,94	4,87	5,42	5,03	5,82
8	4,89	5,32	5,58	5,73	5,30	5,51
9	5,39	4,85	5,54	5,30	5,07	4,71
10	5,66	5,23	4,64	5,42	5,53	4,82

Tabla 6.6: Tiempo en segundos que demora obtener una recomendación en el tercer caso de pruebas. Que considera para cada *set* de iteraciones un medio persistente con las comunidades detectadas por el medio anterior y un primer nivel de *community cache* desactivado.

	U1			**U2**		
It	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
1	9,27	7,07	8,69	15,61	16,01	14,64
2	4,75	5,71	5,39	8,82	10,20	8,94
3	3,80	4,78	3,63	8,27	8,75	6,37
4	4,79	5,79	4,48	7,57	6,85	5,63
5	5,52	6,29	4,53	6,46	5,07	5,74
6	5,22	5,45	3,98	4,80	4,25	4,90
7	4,72	5,06	5,57	5,48	5,52	4,09
8	4,97	5,30	5,34	4,65	5,55	5,54
9	5,14	5,62	4,81	4,17	5,42	4,59
10	4,54	4,89	5,24	5,93	5,12	5,32

Tabla 6.7: Tiempos promedio que demora la obtención de la recomendación detectando comunidades con los distintos escenarios (E).

	U1			**U2**		
It	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
1	15,46	19,22	16,09	37,86	23,83	22,01
2	2,68	4,42	4,77	12,19	9,74	9,18
3	4,55	3,74	3,65	9,92	7,24	7,09
4	4,55	5,01	3,41	7,42	5,51	5,84
5	4,61	5,48	4,81	4,73	5,33	5,56
6	4,98	5,28	5,43	5,09	5,94	5,17
7	5,08	5,16	7,43	4,66	5,07	4,90
8	4,60	7,01	7,26	4,44	4,12	5,20
9	4,57	4,86	5,80	4,56	8,03	4,53
10	5,24	4,65	4,14	4,26	5,76	3,44

6.4 RESUMEN

En este capítulo se ha descrito y presentado los resultados del experimento aplicado con el propósito de conseguir el objetivo general planteado en este trabajo de Tesis. En primer lugar, se ha descrito detalladamente el experimento realizado, con sus componentes y distintos escenarios. Luego, se ha caracterizado el *dataset* utilizado para realizar las pruebas, describiendo

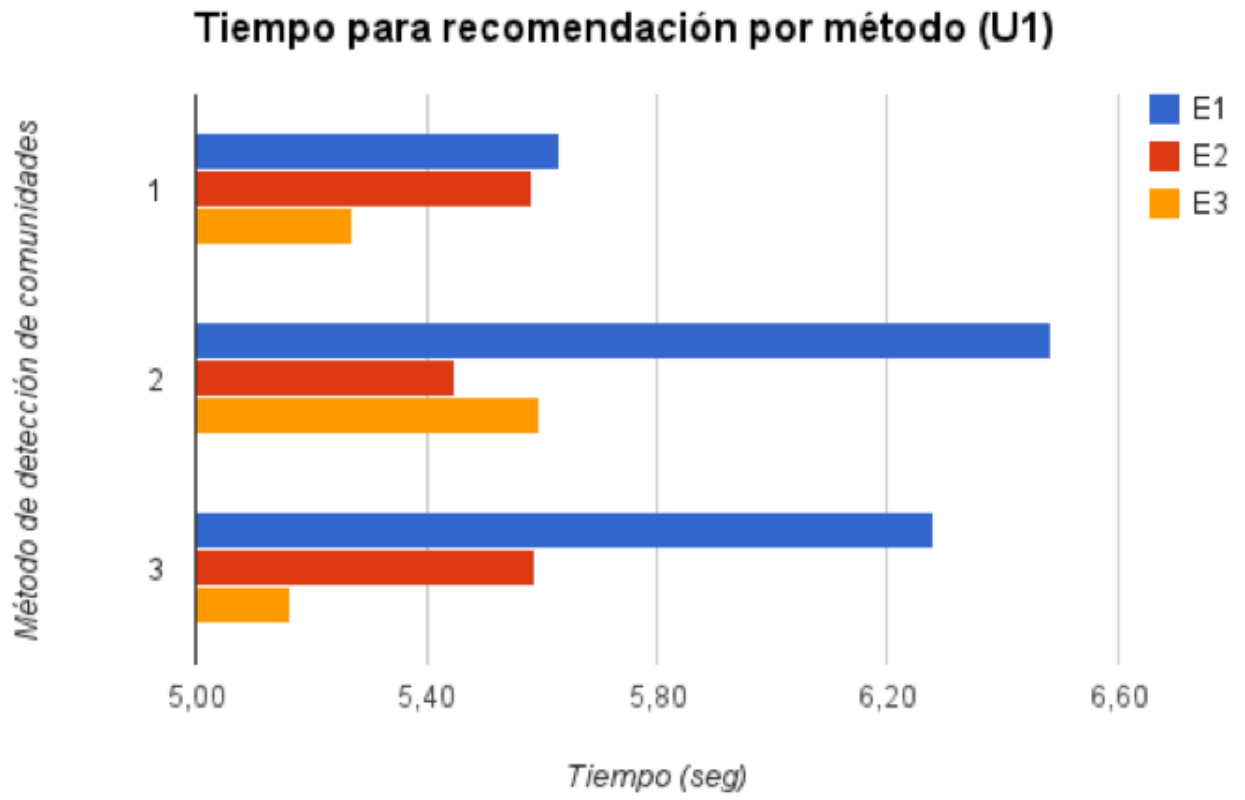

Figura 6-8: *Tiempos promedio para entregar una recomendación, en los distintos escenarios (E) de prueba, con el usuario 1 como activo.*

el mecanismo de extracción, construcción, persistencia y cuantificación del tamaño de información contenida. Posteriormente, se han mostrado ejemplos de detección de comunidades utilizando el *dataset* descrito. Finalmente, se han expuesto los resultados de la experimentación, logrando reducir el tiempo requerido para realizar una recomendación, mediante el uso del *framework* y componentes descritos en los Capítulos 3 y 4.

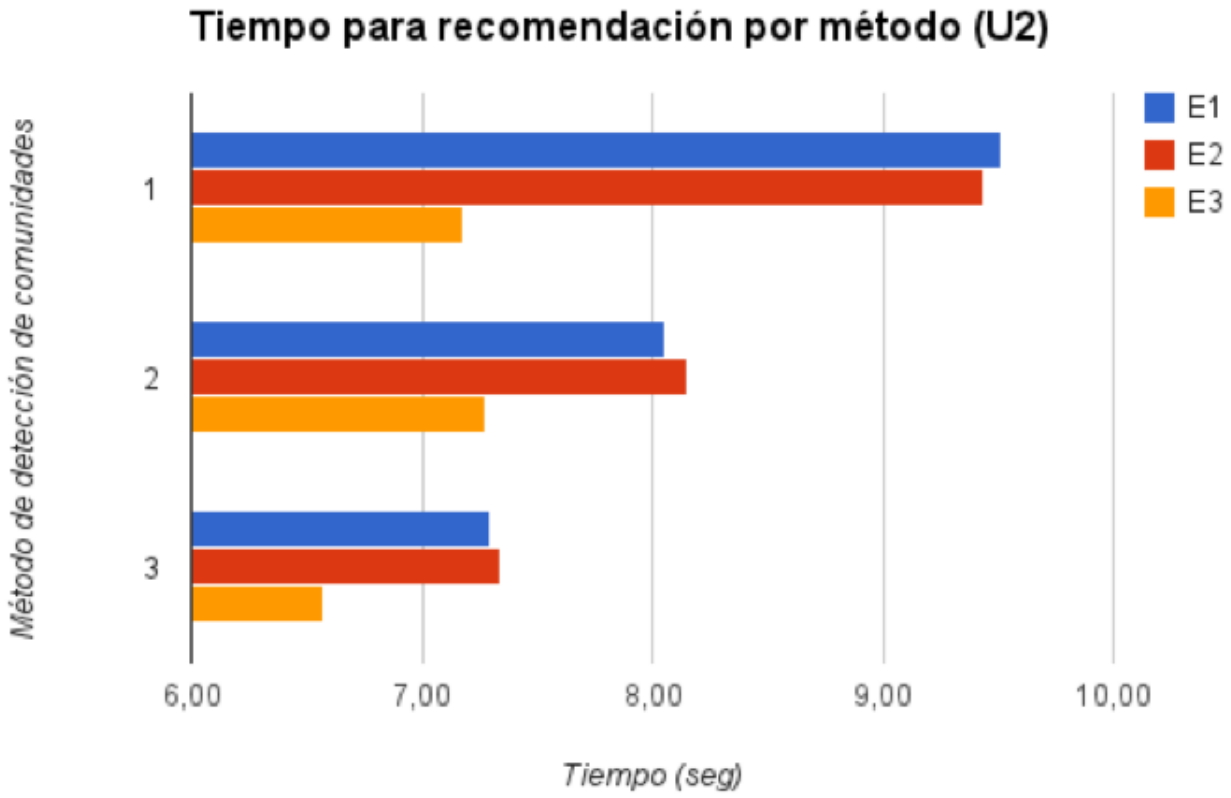

Figura 6-9: *Tiempos promedio para entregar una recomendación, en los distintos escenarios (E) de prueba, con el usuario 2 como activo.*

CAPÍTULO 7. CONCLUSIONES

En este capítulo, se expone respecto a las conclusiones obtenidas del trabajo de tesis desarrollado. En primer lugar, se describe y evidencia el cumplimiento de los objetivos específicos. Luego, se explicitan los aportes realizados tanto al área de los SR, como al área de detección de comunidades. Luego, se presentan aquellas aristas de investigación futuras, nacidas en base al trabajo aquí expuesto. Para terminar, se enuncian reflexiones finales en relación al presente trabajo de tesis.

7.1 RESPECTO A LOS OBJETIVOS ESPECÍFICOS

Con respecto a los objetivos específicos definidos para el trabajo de tesis:

1. Seleccionar técnicas y/o algoritmos que permitan la detección de comunidades.

   Se evaluaron distintas alternativas de implementación de algoritmos para la detección de comunidades, en el Capítulo 3, en la sección 3.6 se definen aquellos algoritmos que permiten la detección de comunidades (Blondel, Guillaume, Lambiotte, Lefebvre, 2008; Raghavan, Albert, Kumara, 2007; Newman, 2006; Rosvall Bergstrom, 2008; Traag Bruggeman, 2009).

   Estos algoritmos serán implementados desde una librería llamada python-igraph. Esta librería ha sido construída en base a un kernel en C, y cuenta con abstracciones hacia ruby, R y python. Precisamente es esta última abstracción la que se utiliza para desarrollar un servicio de tipo REST, al que RBox se conecta, permitiendo la detección de comunidades. Uno de los algoritmos es de tipo no determinista al aplicar un método de Label Propagation, el cual fue incorporado al servicio solamente para disponibilizar todos aquellos algoritmos que igraph

provee y no fue utilizado para las posteriores pruebas del sistema.

2. Definir la arquitectura del servicio que se añadirá a RBox, junto con un proceso de encapsulamiento de los datos del *framework* para estandarizar el consumo del servicio.

   En el Capítulo 3, se describe la arquitectura del servicio que se ha construido para detectar comunidades, la Figura 7-1 muestra el diagrama de la arquitectura definida. El servicio está compuesto de distintas piezas de *software* que permiten recibir, interpretar, calcular y transformar la información desde un formato orientado a vértices y nodos hacia una comunidad.

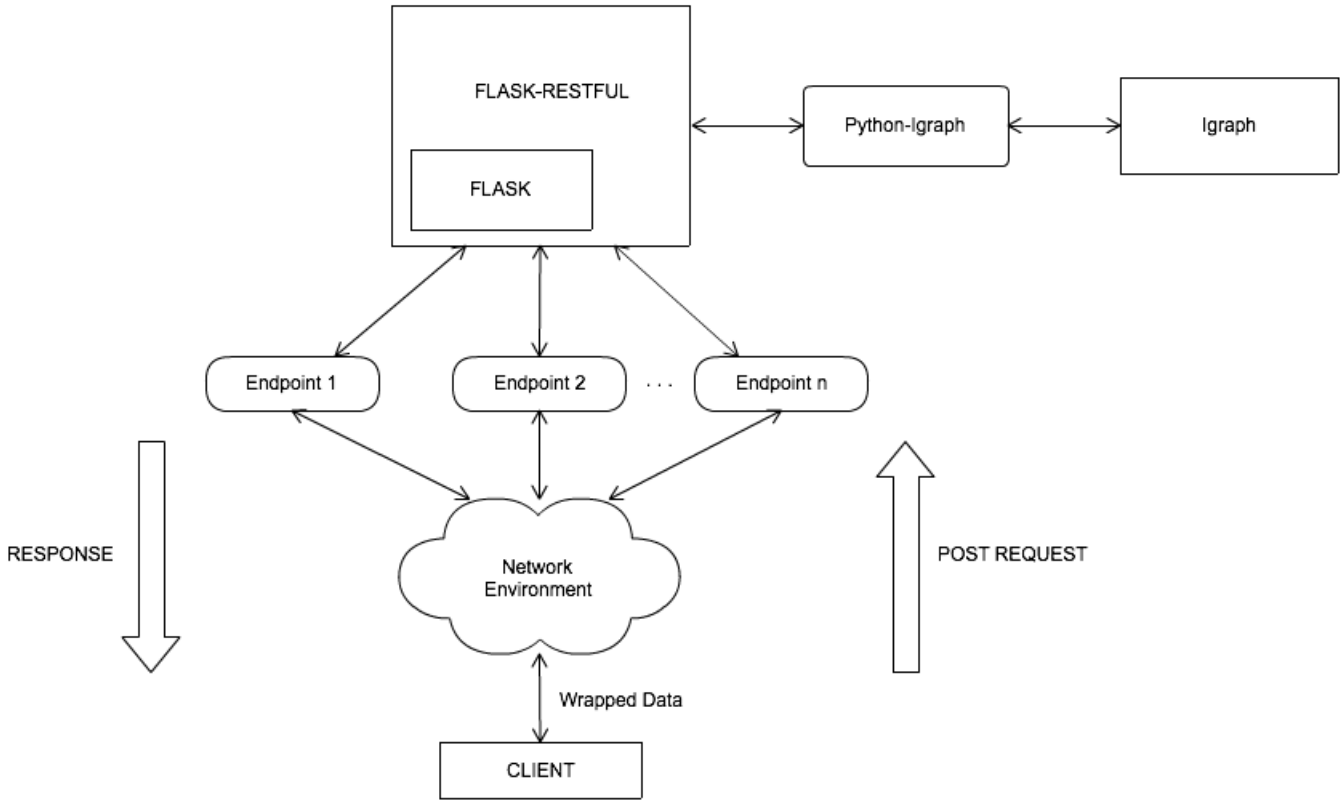

Figura 7-1: *Arquitectura definida para el servicio de detección de comunidades.*

Su añadidura a RBox se define en el Capítulo 4, donde se describe una pieza que permite comunicarse con servicios de tipo REST. En particular, en la sección 4.4 se describe el mecanismo que permite manejar la respuesta del servicio REST hacia una representación

de comunidad definida en la 3-Ontology. Este mecanismo (ver Figura 7-2) comprende una comunicación desde una pieza de encapsulamiento de las interacciones almacenadas en el modelo de la 3-Ontology (Wrapper) hasta el consumo del servicio y reconversión de las comunidades detectadas hacia el esquema.

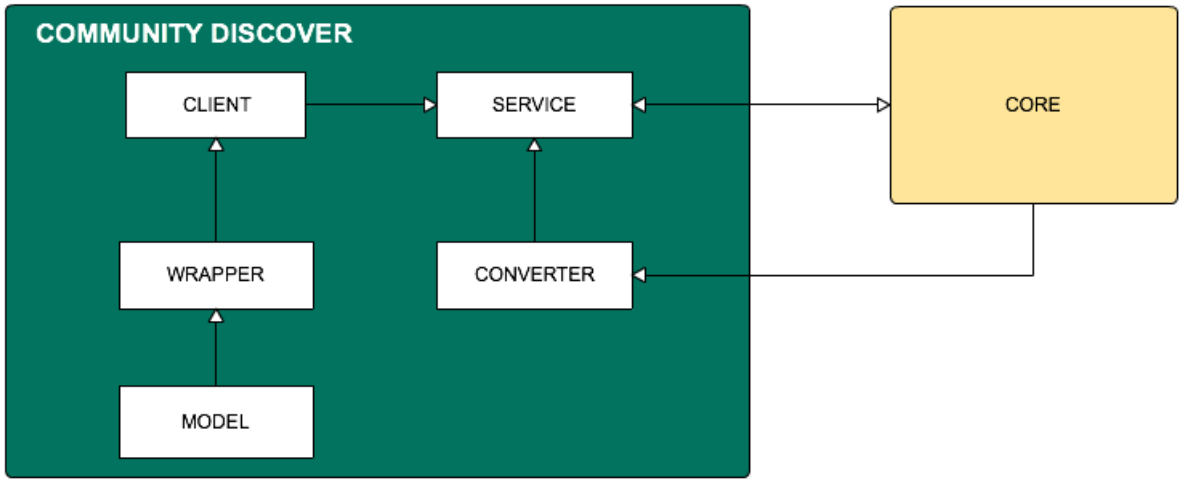

Figura 7-2: *Mecanismo de comunicación de las piezas del componente de detección de comunidades añadido a RBox.*

Por otra parte, el proceso de encapsulamiento de la información hacia el servicio, forma parte de la pieza Wrapper de *Community Discover*. Esta pieza permite transformar desde el esquema de la 3-Ontology hacia un esquema de grafos con el objetivo de enviarlo como parámetro al momento de consumir el servicio.

3. Implementar y añadir a RBox un servicio que detecte y persista las comunidades, *community cache*, a partir de los datos encapsulados.

En el Capítulo 4, se describe una implementación de *Portrait* que considera la detección y persistencia de las comunidades encapsuladas a través del servicio de detección de comunidades. Esta pieza forma parte de *Community Discover* y ha sido añadida como una extensión del Core de RBox. En la Figura 7-3 se muestra el diseño de la arquitectura en la que este servicio está inserto.

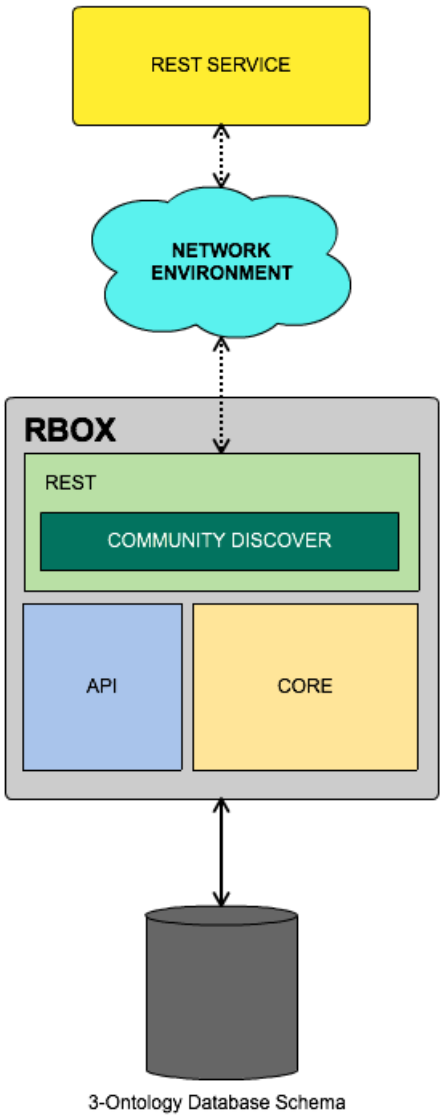

Figura 7-3: *Diseño de arquitectura en la que el servicio de detección de comunidades está inserto.*

El servicio posee una característica llamada *community cache*, que consta de dos niveles. En un primer nivel, determina si la detección y persistencia se realiza únicamente cuando el usuario activo no tiene comunidades asociadas. En el segundo nivel, tiene un mecanismo de control respecto a la persistencia, por cuanto se toma en cuenta el hecho de que la comunidad ya forma parte del modelo de la 3-Ontology, considerando el conjunto de usuarios que la forman.

El servicio toma como antecedentes a todas las comunidades existentes en el sistema, el cliente del servicio REST de detección de comunidades, una instancia del DAO de RBox para

permitir la persistencia de comunidades, una instancia de *GraphWrapper* que encapsule los eventos, que será enviado como request al servicio de detección de comunidades, el algoritmo que se utilizará para detectar comunidades, un valor booleano para determinar si se hace uso del primer nivel de *community cache*.

4. Implementar un algoritmo de recomendación utilizando la información de las comunidades como antecedente.

   En el Capítulo 5 se muestran distintos ejercicios de recomendación basados en las comunidades de un usuario activo, considerando un algoritmo de recomendación de los top N *tags* más utilizados en una comunidad. Los que se busca es medir la performance de la generación de recomendación considerando el tiempo necesario para realizar las operaciones de encapsulación, consumo, transformación y persistencia para detectar comunidades, evaluando si el mecanismo de community-cache existente es realmente una medida de reducción del tiempo necesario para recomendar detectando comunidades a un usuario activo, considerando aquellas comunidades detectadas como una retroalimentación a recomendaciones posteriores.

5. Desarrollar conclusiones en relación a los resultados obtenidos.

   En la Sección 5.3 se exponen conclusiones respecto a los resultados obtenidos de la experimentación. Se marcan las diferencias entre la partida "en frío" de una recomendación y la detección de comunidades cuando esta se fuerza para cada recomendación. Los ejercicios han sido realizados considerando tres estrategias distintas de detección de comunidades, diez iteraciones de recomendación y tres escenarios.

   Estos escenarios son: primero, detectando y persistiendo comunidades solamente cuando el usuario no tenga comunidades asociadas; segundo y tercero, detectando y persistiendo comunidades siempre, para cada iteración. Con la salvedad de que el tanto el primer como el segundo escenario consideran a un medio persistente vacío entre la ejecución de distintas

estrategias.

Los resultados del primer caso de pruebas muestran que en la primera iteración el tiempo que se requiere para obtener la recomendación siempre es mayor que en las iteraciones posteriores. Los resultados del segundo caso de pruebas muestran el funcionamiento de la *community cache*, ya que el *framework* no persiste aquellas que ya existen, eliminando duplicidad. Finalmente, en el tercer caso de pruebas el tiempo de partida "en frío" se ha reducido, en general, para ambos usuarios ya que existen comunidades que ya han sido detectadas en iteraciones precedentes.

6. Publicar los resultados en una revista de la especialidad.

Están en desarrollo dos publicaciones en revistas de la especialidad. Una relacionada con el aporte científico de la mejora en eficiencia de los SR usando el mapping de la 3-Ontology el concepto de *community cache*. La segunda publicación estará basada en la arquitectura que permite construir el *framework* que retroalimenta el esquema de la 3-Ontology con las comunidades detectadas, reduciendo así el tiempo requerido para volver a realizar la misma recomendación.

7.2 APORTES DEL TRABAJO

En primer lugar, este trabajo provee un *framework* de detección de comunidades en base a una arquitectura unificada de componentes. Esto facilita la posibilidad de incorporar nuevos métodos de detección de comunidades al servicio presentado, o bien, desarrollar encapsuladores para otro servicio o API que provea prestaciones similares. Se desprende también de lo recién mencionado que, con la incorporación de un API para implementar clientes de tipo REST, se incorpora a RBox la posibilidad de consumir servicios de distinta índole, como análisis de sentimientos, geolocalización o *named entity recognition*. Además, y como la arquitectura orientada a servicios desarrollada en este trabajo es extensible, se pueden incorporar elementos y algoritmos de análisis de redes sociales

para el uso y disposición de los sistemas de recomendación que sean implementados en RBox.

En segundo lugar, se ha incorporado a RBox un mecanismo de recomendación basado en comunidades que se retroalimenta de las recomendaciones hechas en el pasado como una cache. Esto permite mejoras tanto en los tiempos necesarios para recomendar contenido a los usuarios, desde la perspectiva de las comunidades a las que pertenecen, como en el control de duplicidad de comunidades que puedan existir, ya que no se persistirá dos veces una misma comunidad.

En tercer lugar, el *framework* que permite transformar las interacciones sociales en grafos de interacciones es un modelo flexible, que permite construir comunidades desde distintos puntos de vista y considerando diferentes interacciones existentes en un dominio en particular. En el caso de este trabajo de tesis, las comunidades fueron construidas a partir de las interacciones ocasionadas por las menciones de los usuarios en *Twitter*. No obstante, nada impide construir comunidades distintas considerando otras interacciones, como *replying*, *following* y *retweeting*.

7.3 TRABAJOS FUTUROS

En este trabajo se propone e implementa una arquitectura que permite retroalimentar las comunidades en un esquema 3-Ontology a partir de recomendaciones basadas en un usuario activo. No obstante, simplemente se comienza la exploración de la dimensión de la comunidad y tras esto se vislumbran los siguientes trabajos futuros:

- Incorporar un servicio, plugin o extensión que permita realizar un análisis de redes sociales, orientadas a la dimensión de la comunidad. Permitiendo enriquecer la recomendación en base a la confianza y popularidad de los usuarios que componen las comunidades.
- Explorar la dimensión de los lugares, mediante la incorporación de un servicio, plugin o extensión que explicite los tipos de información espacial utilizados en los sistemas de recomendación.
- Definir una extensión de la arquitectura que permita obtener granularidad y detalle en los datos de las comunidades. Como por ejemplo obtener, en tiempo real desde la fuente de información (sea API, crawling, scrapping, entre otras), mayor cantidad de datos en relación a un usuario

en particular. Esto completaría aún más la construcción dinámica de comunidades en base a las distintas interacciones o tipos de eventos.

- Construir un dashboard o herramienta exploratoria que facilite la navegación por los contenedores de sentido de la 3-Ontology, permitiendo a profesionales especializados la toma de decisiones en base al comportamiento e interacciones presentes en el conjunto de datos.

- Construir, a partir de RBox, observatorios de ciertas fracciones de la Web, permitiendo la definición dinámica de extractores de información a un esquema 3-Ontology. Esto permitiría alimentar a RBox con datos de distintas fuentes en base al interés que la entidad que provee recomendaciones defina.

7.4 REFLEXIONES FINALES

Este trabajo de tesis pone fin a dos años de esfuerzo y dedicación que han supuesto un crecimiento sustancial en mi calidad como profesional y como investigador. Las asignaturas del programa de Magíster en Ingeniería Informática de la Universidad son un buen complemento a la formación inicial obtenida en pregrado. Personalmente, en el año 2011, me titulé de Ingeniero de Ejecución en Computación e Informática y, debido al trabajo y la dedicación puesta en la carrera, fui beneficiado por una media beca de arancel para ingresar al programa de Magíster. Esto ha sido un desafío no menor, ya que típicamente el programa está diseñado para que sea completado por ingenieros civiles (con los dos años extra de formación que implica ese plan de estudios). Sin embargo, la motivación principal ha sido sentar un precedente y demostrar a todos los estamentos (sobre todo a mis colegas de carrera) que las capacidades son transversales a la carrera que uno estudie y la motivación que se requiere para completar un programa de postgrado viene principalmente por las aspiraciones del estudiante, a pesar de que muchas veces el trabajo no es buen amigo del estudio y las empresas no están plenamente conscientes de los beneficios que trae para ellos el contar con un estudiante de postgrado en su equipo de ingenieros. En ese sentido, el desafío que hay que asumir ahora es el de abrir camino a las generaciones que vienen con el hecho de intentar cambiar la tendencia nacional, demostrando pragmáticamente que el rendimiento y el conocimiento de un profesional con postgrado es aporte más que relevante tanto para la línea de innovación de las empresas, como para

la línea de negocios.

El programa de postgrado, en términos profesionales, ha sido un aporte significativo. A pesar de que muchas veces sus contenidos no están actualizados en base a las actuales tendencias de la profesión, si me ha instado a no dejar de estudiar y estar a la vanguardia en un mundo tremendamente competitivo, en el que las tecnologías, patrones, arquitecturas y tendencias no viven más de dos años. Se destaca, sin lugar a dudas, el conocimiento adquirido al trabajar a la par de compañeros y profesores talentosos, en especial con aquellos que pertenecen al grupo FONDEF, que lidera el Dr. Edmundo Leiva. La necesidad de entregar, en este trabajo, una solución tecnológica cuya calidad estuviera a la altura de los distintos componentes incorporados ya al proyecto, ha sido otra manera de impulsar el aprendizaje, la lectura y mejorar progresivamente mis habilidades en diseño y desarrollo de soluciones tecnológicas que están inmersas en un contexto de uso constante y no quedan guardadas en la estantería.

El haber trabajado en una tesis enfocada a *social media* ha supuesto la continuación de una inquietud que nace en el desarrollo de mi memoria de pregrado y que sentí inconclusa. Me ha permitido investigar, analizar y conocer distintos aspectos de un tema que hoy mueve y tendencia fuertemente el uso de Internet como plataforma fundamental para la masificación de contenidos y las relaciones sociales mediada por tecnología. Esta tendencia está manteniéndose en el tiempo y alcanzando distintas aristas, desde meramente compartir información, hasta proveer, con ayuda de la nube, distintos servicios de análisis de tópicos y comunidades enfocadas al *business intelligence* y el cultivo de la inteligencia colectiva. En mi opinión, lo colectivo y lo comunitario son los negocios del mañana y el aplicar los conocimientos involucrados en esta tesis puede contribuir, aunque sea en parte, a esta tendencia.

REFERENCIAS

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, *23*(1), 103–145.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, *17*(6), 734–749.

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender systems handbook*, (p. 217–253). Springer.

Andersen, P. (2007). *What is Web 2.0?: ideas, technologies and implications for education*, vol. 1. JISC Bristol, UK.

Andersen, R., & Lang, K. J. (2006). Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, (p. 223–232). ACM.

AP (2013). Number of active users at facebook over the years.

Arab, M., & Afsharchi, M. (2014). Community detection in social networks using hybrid merging of sub-communities. *Journal of Network and Computer Applications*, *40*, 73–84.

Asur, S., Parthasarathy, S., & Ucar, D. (2009). An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *3*(4), 16.

Backstrom, L., Dwork, C., & Kleinberg, J. (2007). Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, (p. 181–190). ACM.

Bianco, P., Kotermanski, R., & Merson, P. F. (2007). Evaluating a service-oriented architecture. *Software Engineering Institute, Carnegie Mellon University*.
URL `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8443`

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132.

Brambilla, M., Fraternali, P., & Vaca, C. (2011). A notation for supporting social business process modeling. In *Business Process Model and Notation*, (p. 88–102). Springer.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331–370.

Chakrabarti, D., & Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, *38*(1), 2.

Csárdi, G., & Nepusz, T. (2006). The igraph software package for complex network research.

Dourisboure, Y., Geraci, F., & Pellegrini, M. (2007). Extraction and classification of dense communities in the web. In *Proceedings of the 16th international conference on World Wide Web*, (p. 461–470). ACM.

Du, N., Wu, B., Pei, X., Wang, B., & Xu, L. (2007). Community detection in large-scale social networks. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, (p. 16–25). ACM.

Ekstrand, M. D., Ludwig, M., Konstan, J. A., & Riedl, J. T. (2011). Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit. In *Proceedings of the fifth ACM conference on Recommender systems*, (p. 133–140). ACM.

Gibson, D., Kumar, R., & Tomkins, A. (2005). Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*, (p. 721–732). VLDB Endowment.

González, A. (2012). *Comparando la funcionalidad y la usabilidad de dos aplicaciones basado en un proceso de negocios de generación de noticias.*. Ph.D. thesis, Universidad de Santiago de Chile, Departamento de Ingeniería Informática.

Guinard, D., Ion, I., & Mayer, S. (2012). In search of an internet of things service architecture: REST or WS-*? a developers' perspective. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, (p. 326–337). Springer.

Hechter, M. (1988). *Principles of group solidarity*, vol. 11. Univ of California Press.

Hubler, C., Kriegel, H.-P., Borgwardt, K., & Ghahramani, Z. (2008). Metropolis algorithms for representative subgraph sampling. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, (p. 283–292). IEEE.

Java, A., Joshi, A., & Finin, T. (2008). Detecting communities via simultaneous clustering of graphs and folksonomies. In *Proceedings of the Tenth Workshop on Web Mining and Web Usage Analysis (WebKDD)*. ACM.

Kerr, D. (2012). Twitter data shows trending topics tend to change rapidly.
URL `http://www.cnet.com/news/twitter-data-shows-trending-topics-tend-to-change-rapid`

Lam, X. N., Vu, T., Le, T. D., & Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, (p. 208–211). ACM.

Leiva-Lobos, E. P., & Covarrubias, E. (2002). The 3-ontology: a framework to place cooperative awareness. In *Groupware: Design, Implementation, and Use*, (p. 189–199). Springer.

Leskovec, J., & Faloutsos, C. (2006). Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (p. 631–636). ACM.

Lin, Y.-R., Sundaram, H., Chi, Y., Tatemura, J., & Tseng, B. L. (2007). Blog community discovery and evolution based on mutual awareness expansion. In *Proceedings of the IEEE/WIC/ACM international conference on web intelligence*, (p. 48–56). IEEE Computer Society.

Maiya, A. S., & Berger-Wolf, T. Y. (2010). Sampling community structure. In *Proceedings of the 19th international conference on World wide web*, (p. 701–710). ACM.

Martin, R. C. (2003). *Agile software development: principles, patterns, and practices*. Prentice Hall PTR.

Newman, M. E. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, *74*(3), 036104.

Padula, M., Reggiori, A., & Capetti, G. (2009). Managing collective knowledge in the web 3.0. In *Evolving Internet, 2009. INTERNET'09. First International Conference on*, (p. 33–38). IEEE.

Palla, G., Barabási, A.-L., & Vicsek, T. (2007). Quantifying social group evolution. *Nature*, *446*(7136), 664–667.

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*(7043), 814–818.

Palomino, M. (2012). *Modelo extensible para sistemas de recomendación en la Web 2.0*. Ph.D. thesis, Universidad de Santiago de Chile, Departamento de Ingeniería Informática.

Papadopoulos, S., Kompatsiaris, Y., & Vakali, A. (2009). Leveraging collective intelligence through community detection in tag networks. *CKCaR, September*.

Plantié, M., & Crampes, M. (2013). Survey on social community detection. In *Social Media Retrieval*, (p. 65–85). Springer.

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, *76*(3), 036106.

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, *105*(4), 1118–1123.

Schlitter, N., & Falkowski, T. (2009). Mining the dynamics of music preferences from a social networking site. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, (p. 243–248). IEEE.

Simpson, E. (2008). Clustering tags in enterprise and web folksonomies. In *ICWSM*.

Smith, C. (2014). By the numbers: 125 amazing facebook user statistics.
URL `http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/`

Symeonidis, P., Papadimitriou, A., Manolopoulos, Y., Senkul, P., & Toroslu, I. (2011). Geo-social recommendations based on incremental tensor reduction and local path traversal. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, (p. 89–96). ACM.

Tang, L., & Liu, H. (2010). Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, *2*(1), 1–137.

Tang, L., Liu, H., & Zhang, J. (2012). Identifying evolving groups in dynamic multimode networks. *Knowledge and Data Engineering, IEEE Transactions on*, *24*(1), 72–85.

Tang, L., Liu, H., Zhang, J., & Nazeri, Z. (2008). Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (p. 677–685). ACM.

Tang, L., Wang, X., & Liu, H. (2009). Uncoverning groups via heterogeneous interaction analysis. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, (p. 503–512). IEEE.

Tareen, B. K., Lee, J.-w., & Lee, S.-g. (2010). Synergy: A workbench for collaborative filtering algorithms on user interaction data. In *International Workshop on User Data Interoperability in the Social Web*.
URL `http://ids.snu.ac.kr/w/images/8/8f/UDISW2010.pdf`

Traag, V., & Bruggeman, J. (2009). Community detection in networks with positive and negative links. *Physical Review E*, *80*(3), 036115.

Victor, P., De Cock, M., & Cornelis, C. (2011). Trust and recommendations. In *Recommender systems handbook*, (p. 645–675). Springer.

Vásquez, R. (2014). *Modelo de representación de datos para Sistemas de Recomendación*. Ph.D. thesis, Universidad de Santiago de Chile, Departamento de Ingeniería Informática.

Wasserman, S. (1994). *Social network analysis: Methods and applications*, vol. 8. Cambridge university press.

Yujie, Z., & Licai, W. (2010). Some challenges for context-aware recommender systems. In *Computer Science and Education (ICCSE), 2010 5th International Conference on*, (p. 362–365). IEEE.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, (p. 452–473).

Zanker, M., & Jessenitschnig, M. (2009). Case-studies on exploiting explicit customer requirements in recommender systems. *User Modeling and User-Adapted Interaction*, *19*(1-2), 133–166.