UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Estrategias de planificación para motores de búsqueda verticales

Danilo Fernando Bustos Pérez

Profesor Guía: Dra. Carolina Bonacic Castro
Profesor Co-guía: Dr. Mauricio Marín Caihuán

Trabajo de Titulacíon presentado en conformidad a los requisitos para obtener el Título de Ingeniero Civil Informático

SANTIAGO DE CHILE
2013

AGRADECIMIENTOS

Dedicado a ... .

RESUMEN

Resumen en Castellano

Palabras Claves: keyword1, keyword2 .

ABSTRACT

Resumen en Inglés

Keywords: keyword1, keyword2 .

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

La Web 2.0 o "Web Social" se caracteriza por ser una Web centrada en el usuario, la participación, la interactividad y la colaboración (Murugesan, 2007). *Youtube*[1], *Facebook*[2], *Flickr*[3] ahora son compañeros cotidianos y habituales en el mundo tecnológico que nos toca vivir. Antes de este cambio cultural, la Web 1.0 estaba centrada principalmente en el contenido, el cual era actualizado unilateralmente por usuarios expertos; era principalmente comunicación de una sola vía. En cambio, la Web 2.0 elimina las barreras entre productores y consumidores de información, creando canales de doble vía, empoderando al usuario final como creador activo de información. Este cambio produjo una verdadera revolución social del uso de las tecnologías en la red. Sea dicho de paso, esta revolución se vio facilitada por la incorporación de la tecnología de la "sindicación" la cual permite retroalimentación en línea sobre la información de productores y consumidores. En definitiva la Web 2.0 es una forma distinta de usar la Web como canal de comunicación (Padula et al., 2009) que permite a los usuarios finales, además de publicar, interactuar con los distintos contenidos de la Web (textual, multimedia, etc) de diversas formas.

El masivo uso de las aplicaciones de la Web 2.0 ha poblado Internet de una enorme cantidad de información, por este motivo, y para ayudar a los usuarios en la búsqueda de información, se desarrollan dos tipos de sistemas: los motores de búsqueda y sistemas de

[1] http://www.youtube.com/
[2] https://www.facebook.com/
[3] http://www.flickr.com/

recomendación (en adelante SR). Estos últimos entregan información filtrada de acuerdo al comportamiento de otros usuarios (meta-información), de las preferencias propias, ajenas y de los atributos de la información buscada. El comportamiento de los usuarios puede ser modelado como un conjunto de interacciones de valor colaborativo que permiten obtener información sobre la colaboración entre usuarios explícita o implícita. Estas interacciones pueden ser de distintos tipos (Brambilla et al., 2011), e incluyen actividades como etiquetar (*tagging*), comentar (commenting), valorizar (*rating*) entre otros.

El área de investigación de los SR nace a mediados de los 90', cuando *GroupLens*[4] publica el primer artículo, que se basa en el filtrado colaborativo para generar recomendaciones de noticias (Herlocker et al., 1999). En el estado del arte (Adomavicius & Tuzhilin, 2005) se plantea que esta área se ha mantenido en expansión debido a que provee un gran número de aplicaciones prácticas que ayudan a los usuarios a encontrar información de utilidad.

En la actualidad existen varios SR, enfocados a distintos dominios de aplicación. Algunos ejemplos son sitios como *MovieLens*[5] y *Netflix*[6] para recomendación de peliculas; *Google News*[7] para recomendación de noticias; *LastFm*[8] para recomendación de música, y *Amazon*[9] para recomendación de *e-commerce*, entre otras. En este mismo ámbito y con el objetivo de mejorar su SR, *Netflix* creó un concurso llamado *"Netflix Prize"*[10] que premiaba con un millón de dólares a quienes lograran mejorar su algoritmo de recomendación basado en la métrica **RMSE** (error cuadrático medio). Este concurso motivó la investigación en el área promoviendo nuevas formas de abordar el problema de recomendación basada en *rating*.

Los SR tradicionales se basan en dos dimensiones: el usuario y el ítem para calcular una función de utilidad (Adomavicius & Tuzhilin, 2005). Básicamente se construyen perfiles

[4]http://www.grouplens.org/
[5]http://movielens.umn.edu/login
[6]www.netflix.com
[7]news.google.com
[8]www.last.fm
[9]www.amazon.com
[10]http://www.netflixprize.com

de usuario e ítem, como también se modelan las interacciones de los usuarios hacia los ítems. Sin embargo, en el desarrollo de la investigación en el área se están modelando nuevas dimensiones con el objetivo de mejorar el proceso de recomendación (Adomavicius & Tuzhilin, 2011). Ejemplos de estas dimensiones son el prestigio y afinidad entre usuarios para medir la confiabilidad de la recomendación (Victor et al., 2011). La geolocalización (Panagiotis et al., 2011) basándose en la ubicación física y/o virtual del usuario. Otro conjunto de dimensiones importantes se basan en el contexto de interacción del usuario mejorando así el proceso de recomendación (Adomavicius & Tuzhilin, 2011). Luego estos sistemas se han sobre-especializado con el objetivo de entregar recomendaciones de mayor relevancia para el usuario final.

En las empresas se hace cada vez más necesario contar con SR para aumentar las probabilidades de selección de sus productos y/o servicios por parte de sus clientes. Cada empresa cuenta con registros de las interacciones de sus clientes en diversos repositorios de datos que suelen ser transversales a los sistemas de la empresa. Además, existe la información proveniente de las aplicaciones basadas en la Web 2.0, que proveen API's para consultar información de sus clientes, como también aplicaciones innovadoras de las empresas que se basan en la ideología de la Web 2.0.

En el área de investigación de los SR es difícil reproducir y extender los resultados obtenidos en las distintas publicaciones (Ekstrand et al., 2011). Por lo tanto, se debe re-implementar los SR para un dominio particular donde éstos son usados. En este contexto, han aparecido trabajos que intentan aportar estándares en la representación, construcción y validación de SR que permitan la construcción sistemática de SR. Por ejemplo en *Lenskit* (Ekstrand et al., 2011) se provee un *framework* extensible y mantenible para la construcción y evaluación de SR basados en filtrado colaborativo. Por otro lado en *Synergy* (Tareen et al., 2010) se plantea un modelo de datos que representa las distintas interacciones que realizan los usuarios con los ítems. Palomino (2012) plantea un *framework* que busca organizar y estandarizar el área de los sistemas de recomendación en dos aspectos: un modelo de representación de datos, y una

arquitectura para proveer servicios de recomendación en una red social.

1.2 DESCRIPCIÓN DEL PROBLEMA

La representación y construcción de SR son fuente de los siguientes desafíos:

1. La Web 2.0 proporciona nuevas formas de interacción para los usuarios. Luego, existen dificultades para modelar múltiples interacciones para que estas sean usadas por los SR (Tareen et al., 2010).

2. Los SR para obtener mejores resultados sobre un dominio específico se acoplan demasiado a la data (Tareen et al., 2010). Esto provoca una sobre-especialización del SR, dificultando la posibilidad de reutilizar la solución en otro dominio de aplicación.

3. Un SR puede degradar sus resultados en el tiempo debido a cambios en la data referente a la interacción de los usuarios por lo que debería ser re-entrenado o cambiado por otro.

4. Las mejoras obtenidas en los algoritmos son difíciles de generalizar para todos los usuarios del sistema, por ejemplo algunos usuarios pueden usar un campo muy reducido de opciones de interacción y otros más.

Un ingeniero a cargo de realizar un SR espera contar con estándares y patrones para la construcción, por lo tanto no suele resolverlo de manera *ad-hoc* como lo hace un investigador del área de los SR. En efecto, un investigador tiende a especializarse para resolver problemas científicos y no se preocupa de proveer estándares para que su solución sea generalizable y/o reutilizable.

En definitiva, el problema de esta tesis está relacionada con la generalización, contextualización y transferencia de los SR. Esto puede expresarse en la siguiente pregunta que guía este trabajo de tesis: ¿Es posible contar con un *framework* que permita representar y construir SR de distinta complejidad en diversos dominios de aplicación?

1.3 OBJETIVOS Y SOLUCIÓN PROPUESTA

A continuación se presenta el objetivo general del proyecto que se concreta con el cumplimiento de los objetivos específicos. Para finalmente declarar los alcances del trabajo de tesis.

1.3.1 Objetivo General

Construir un *framework* que permita la representación y construcción de SR.

1.3.2 Objetivos Específicos

1. Sistematizar el proceso de construcción de SR desde la literatura actual del área.

2. Caracterizar las dimensiones emergentes para los SR.

3. Diseñar un modelo orientado a eventos de representación de datos para SR.

4. Diseñar un conjunto de operaciones para la obtención de datos desde el modelo de datos propuesto.

5. Diseño e implementación de una herramienta para construir SR.

6. Construir SR's basados en el *framework* propuesto.

7. Publicar los resultados de la investigación en una revista de la especialidad.

1.3.3 Alcances

El trabajo de tesis propuesto tiene los siguientes alcances:

- Dada la amplia variedad de SR que se encuentra en la literatura; la solución aunque es genérica se ejemplifica sólo a los SR de filtrado colaborativo *user-user* y de *tagging social* basados en técnicas de *clustering*.

- No es concluyente en la mejora de rendimientos, sino solamente en la eficacia del *framework* propuesto.

- Sólo se validará con el desarrollo de algoritmos que trabajen exclusivamente con *tags* y *ratings* como dos tipos de eventos colaborativos posibles.

- Este producto de software sólo contempla la construcción de un API y una implementación de esta para la construcción de los SR.

- Tanto el API y la implementación serán escritas en *Java Standard Edition 7*.
- Este producto de software no dispondrá extensiones *Windows* ni *Unix*, *Linux*. Sólo podrá ejecutar en la máquina virtual *Java* disponible en estos sistemas operativos.
- No se contempla establecer los procedimientos de evaluación de algoritmos dentro de la herramienta construida.

1.3.4 Solución propuesta

Este proyecto de investigación será del tipo investigación aplicada **I+D** (Investigación + Desarrollo):

1. **Investigación**: se propondrá un modelo que permita la representación de datos basándose en un framework proveniente del área de los sistemas colaborativos y un método de construcción de SR que se encuentren en el estado del arte del área.
2. **Desarrollo**: se construirá una API basada en el modelo y método propuesto.

Finalmente se construirán dos SR usando el API anterior para probar la eficacia del modelo propuesto.

1.3.5 Características de la solución

Las características de la solución propuesta son las siguientes:

- Una especificación o API[11] extensible basada en patrones de diseño que aseguren la flexibilidad requerida basada en un bajo acoplamiento, mantenibilidad y alta cohesión.
- Se construirá una herramienta basada en el API propuesta que permita la representación de datos basado en un método para construir SR.
- La representación de datos se realizará bajo un esquema orientado a eventos que permita modelar el comportamiento de los usuarios en la Web 2.0. Dada esta forma de representación se construirá un conjunto de operaciones para la adquisición de datos del modelo como entrada para los SR.
- La construcción de algoritmos se realizará caracterizando el proceso de recomendación.
- Prueba empírica con la herramienta de software mediante la construcción de SR, que expresen la utilidad del modelo propuesto.

1.3.6 Propósito de la solución

Los propósitos de la solución son los siguientes:

- Proveer un modelo de representación de datos extensible. Para la concreción de lo anterior, se utilizará un *framework* conceptual del área de los sistemas colaborativos denominado *3-Ontology* propuesto por (Leiva-Lobos & Covarrubias, 2002) que permite modelar las interacciones dentro de un ambiente colaborativo.
- Un método que permita la construcción de SR del estado del arte.

[11] *Application Programming Interface*

- Proveer al área de los SR una herramienta de uso personal o empresarial extensible basada en el modelo de representación de datos y el método de construcción.

1.4 METODOLOGÍA Y HERRAMIENTAS DE DESARROLLO

1.4.1 Metodología

La metodología usada para este trabajo como se explicó anteriormente se basa en un proyecto de investigación aplicada I+D.

En el ámbito de la investigación se realizará un estudio exploratorio sobre la representación de datos en los SR y su relación con un modelo particular de representación del contexto de trabajo en los sistemas colaborativos. Para lograr este objetivo se modelará un *framework* para SR. Por lo tanto, el tipo de investigación que se llevará a cabo será de índole exploratorio donde el objetivo es examinar un tema o problema de investigación poco estudiado o que no ha sido abordado antes (Sampieri et al., 2005). Se considera exploratorio ya que se ha encontrado poca literatura que aborde la representación de datos en SR. A partir de esto se contemplan las siguientes etapas:

- Revisión del estado del arte de SR para realizar una categorización.
- Revisión del *framework* conceptual *3-Ontology* para especificar los conceptos básicos del área que serán usados en la solución.

- Modelamiento del proceso de generación de SR.
- Modelamiento de dimensiones emergentes de SR.
- Comparación con otros *framework*.
- Evidenciar la completitud del *framework* mediante la representación teórica de diversos tipos de SR.
- Dar evidencia empírica del *framework* mediante la construcción de SR de filtrado colaborativo.

La metodología de desarrollo del software usada se basa en la filosofía ágil de **SCRUM**[12]. Sin embargo dado que el trabajo fue realizado de forma personal, no se especificaron los roles de la metodología. Se justifica una metodología ágil ya que los requerimientos de la herramienta usada varían durante el desarrollo *framework*. Luego, se tiene la flexibilidad necesaria para abordar cambios de requerimientos a lo largo del desarrollo. La documentación que se genera es simple y básica:

- Documentación del código mediante *Javadocs*[13].
- Diseño de clases.
- Modelo de componentes: especifica las interacciones de los distintos componentes.
- Diagrama de módulos.

[12]https://www.scrum.org/
[13]http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

1.4.2 Herramientas de desarrollo

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

- Sistema operativo: *Linux* con su distribución *Ubuntu 12.04*.
- *Java SE Development kit 7*[14]: se utiliza para el desarrollo del software. Se justifica su uso por ser un lenguaje orientado a objetos de alto nivel que permite una representación legible del *framework* construido. Además, posee una comunidad bastante amplia y gran número de librerías que facilitan el desarrollo.
- *TeXstudio*[15]: herramienta para la construcción de documentos en **LaTeX**.
- *Eclipse Kepler*[16]: el **IDE** *eclipse* en su versión *kepler* que es usada para desarrollo de aplicaciones estándar de *Java*.
- *DIA*[17]: para el desarrollo de los diagramas subyacentes al código y los modelos construidos.
- *Notebook* personal: *Toshiba L505D. AMD Turion(tm) II Dual-Core Mobile M520 × 2*. 4 GB de memoria RAM.
- *Gradle*[18]: herramienta que permite el control del ciclo de vida del software, desde su construcción al despliegue. Adopta los estándares propuestos por *Apache Maven* como estructura de directorios y manejo de dependencias.

[14]http://www.oracle.com/technetwork/java/javase/overview/index.htmltwork/java/javase/overview/index.html
[15]texstudio.sourceforge.net
[16]http://www.eclipse.org/kepler/
[17]http://dia-installer.de/
[18]http://www.gradle.org/

- *Git*[19]: herramienta para el control de versiones del código fuente y documentos asociados al trabajo de tesis.
- *BitBucket*[20]: repositorio remoto de *Git* donde se administra el control de versiones.
- Servidor *Huelen* del Departamento de Ingeniería Informática, **USACH**.

1.5 RESULTADOS OBTENIDOS

La representación de SR basado en el *framework* conceptual *3-Ontology* provee una forma de situar las interacciones de los usuarios hacia los ítems dentro de aplicaciones colaborativas como son los SR. Por lo tanto, se asume que las interacciones tienen un contexto que les da un sentido espacial, temporal y social. La herramienta construida RBOX 2.0 se basa en el modelo propuesto y cumple con las propiedades sistemáticas de mantenibilidad, flexibilidad, reusabilidad y escalabilidad. La eficacia del modelo se valida mediante la construcción de un SR de filtrado colaborativo *user-user* para *Movielens*, y otro SR de *Tagging Social* bajo un *dataset* propietario de una empresa de noticias. En cada SR construido se muestra la correspondencia del dominio de aplicación al modelo propuesto.

[19]http://git-scm.com/
[20]http://bitbucket.org/

1.6 ORGANIZACIÓN DEL DOCUMENTO

En el Capítulo 2 se presenta un marco teórico donde se exponen los fundamentos del trabajo como son los conceptos básicos de los SR, características emergentes de los SR dentro del contexto de la información social y el *framework* conceptual *3-Ontology*. A continuación, en el Capítulo 3 se realiza la definición del modelo propuesto comenzando con una revisión de los trabajos relacionados para luego definir formalmente el modelo. En el Capítulo 4 se presenta el diseño e implementación de RBOX 2.0, la herramienta de software basada en el modelo de representación propuesto. Luego en el Capítulo 5 se presentan dos casos de estudio donde se valida la eficacia del modelo propuesto. En el Capítulo 6 se presentan las conclusiones, para finalmente terminar con las referencias del trabajo.

CAPÍTULO 2. MARCO TEÓRICO

En este capítulo se exponen los conceptos teóricos del presente trabajo de tesis. Rellenar al final

2.1 MOTORES DE BÚSQUEDA VERTICALES

A medida que pasa el tiempo y la Web sigue creciendo, los motores de búsqueda se convierten en una herramienta cada vez más importante para los usuarios. Estas máquinas ayudan a los usuarios a buscar contenido dentro de la Web, puesto que conocen en qué páginas de la Web aparecen qué palabras. Sin un buscador los usuarios estarían obligados a conocer los localizadores de recursos uniformes (URL) de cada uno de los sitios a visitar. Además, los motores de búsquedas en cierto modo conectan la Web, ya que existe un gran número de páginas Web que no tienen referencia desde otras páginas, siendo el único modo de acceder a ellas a través de un motor de búsqueda.

Un motor de búsqueda está construído por diversos componentes, y su arquitectura típica la podemos ver en la Figura 2.1. Existe un proceso denominado *crawling*, éste posee una tabla con las páginas Web iniciales en las que se extrae el contenido de cada una de ellas. A medida que el *crawler* comienza a encontrar enlaces a otras páginas Web, la tabla de páginas a visitar crece. El contenido que se extrae en el procedimiento de *crawling* es enviado al proceso de indexamiento, este se encarga de crear un índice de las páginas visitadas por el *crawler*.

Dado el volúmen de datos involucrado en el procesamiento, se debe tener una estructura

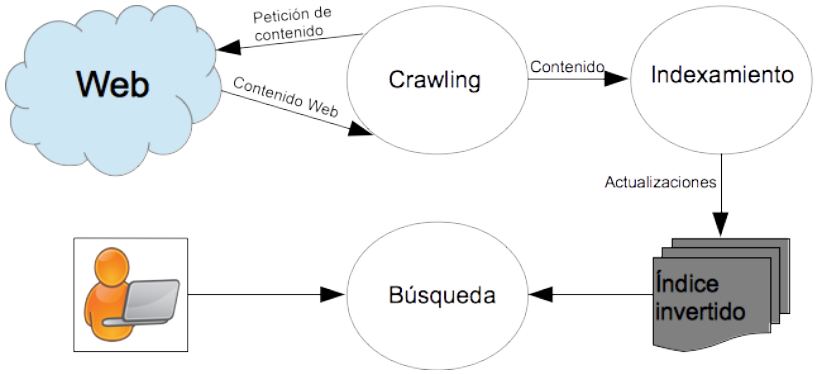

FIGURA 2.1: Arquitectura típica de un motor de búsqueda

de datos que permita encontrar cuáles páginas contienen las palabras presentes en la búsqueda, todo esto dentro de un período de tiempo aceptable. El índice invertido es una estructura de datos que contiene una lista con todas las palabras que el proceso de *crawling* ha visto, y asociada a cada palabra se tiene una lista de todas las páginas Web donde ésta palabra aparece mencionada. El motor de búsqueda construye esta estructura con el objetivo de acelerar el proceso de las búsquedas que llegan al sistema. El proceso de búsqueda es el encargado de recibir la consulta (query), generar un *ranking* de las páginas Web que contienen las palabras de la consulta y finalmente generar una respuesta. Las diversas formas de calcular la relevancia de una página Web será explicado en secciones posteriores.

En un motor de búsqueda se pueden encontrar diversos servicios tales como (a) cálculo de las mejores páginas Web para una cierta consulta; (b) construcción de la página Web con los resultados de la consulta; (c) publicidad relacionada con las *queries*; (e) sugerencia de *queries*; entre muchos otros servicios.

Lo que se hace hoy en día es agrupar computadores para procesar una consulta y producir la respuesta de ésta. Este conjunto de computadores recibe el nombre de *cluster*.

La diferencia entre un motor de búsqueda vertical y uno general, es que el primero se centra solo en un contenido específico de la Web. El *crawler* también debe extraer solo el contenido de aquellas páginas Web que están dentro del dominio permitido. Al ser un dominio acotado,

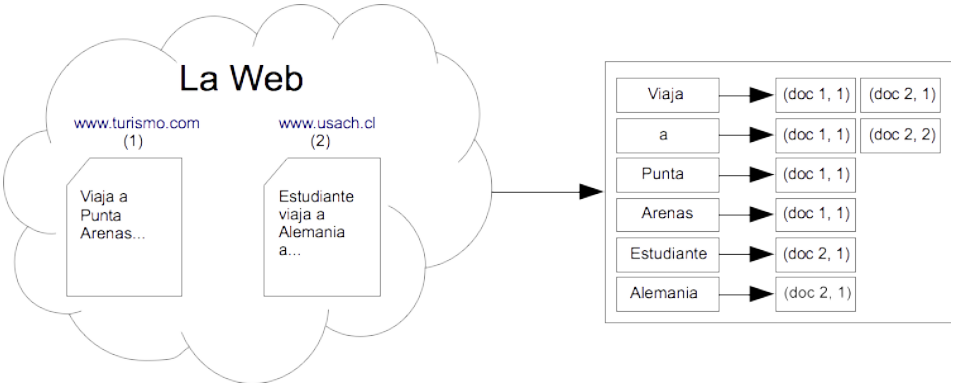

FIGURA 2.2: Índice invertido

las páginas Web a procesar serán menos y por tanto la lista de los términos del índice invertido serán eventualmente de menor tamaño. Sin embargo, en un motor de búsqueda vertical las actualizaciones al índice invertido ocurren con mayor frecuencia.

2.2 ÍNDICE INVERTIDO

Es una estructura de datos que contiene todos los términos (palabras) encontrados. A cada uno de los términos, está asociado una lista de punteros a los documentos (páginas Web) que contienen dicho término. Además se almacena información que permita realizar el *ranking* de las respuestas a las consultas (queries) que llegan al sistema, por ejemplo, el número de veces que aparece el término en el documento. Esta lista recibe el nombre de lista invertida.

Para construir un índice invertido se debe procesar cada palabra que existe en un documento, registrando su posición y la cantidad de veces que éste se repite. Cuando se procesa el término con la información asociada correspondiente, se almacena en el índice invertido (ver Figura 2.2).

El tamaño del índice invertido crece rápido y eventualmente la memoria RAM se agotará antes de procesar toda la colección de documentos. Cuando la memoria RAM se agota, se almacena en disco el índice parcial hasta aquel momento, se libera la memoria y se continúa con el proceso. Además, se debe hacer un *merge* de los índices parciales uniéndo las listas invertidas de cada uno de los términos involucrados.

2.3 ESTRATEGIAS DE EVALUACIÓN DE QUERIES

Existen dos principales estrategias para encontrar los documentos y calcular sus respectivos puntajes de una determinada *query*. Estas son (a) *term-at-a-time* (TAAT) y (b) *document-at-a-time* (DAAT).

2.3.1 TAAT

Este tipo de estrategia procesa los términos de las *queries* uno a uno y acumula el puntaje parcial de los documentos. Las listas invertidas asociadas a un término son procesadas secuencialmente, esto significa que los documentos presente en la lista invertida del término t_i, obtienen un puntaje parcial antes de comenzar el procesamiento del término t_{i+1}. La secuencialidad en este caso es con respecto a los términos contenidos en la *query*.

2.3.2 DAAT

En este tipo de estrategias se valúa la contribución de todos los términos de la query con respecto a un documento antes de evaluar el siguiente documento. Las listas invertidas de cada término de la *query* son procesadas en paralelo, de modo que el puntaje del documento d_j se calcula considerando todos los términos de la query al mismo tiempo. Una vez que se obtiene el puntaje del documento d_j para la *query* completa, se procede al procesamiento del documento d_{j+1}.

2.3.3 Consideraciones

Las estrategias TAAT son mayormente usadas en sistemas de recuperación de información (IR), como son los motores de búsqueda. Cuando se tiene un índice invertido pequeño, las estrategias TAAT rinden adecuadamente, sin embargo cuando los índices invertidos son de gran tamaño las estrategias TAAT poseen dos grandes ventajas: (a) Requieren menor cantidad de memoria para su ejecución, ya que el puntaje parcial por documento no necesita ser guardado y (b) Explotan el paralismo de entrada y salida (I/O) más eficientemente procesando las listas invertidas en diferentes discos simultáneamente.

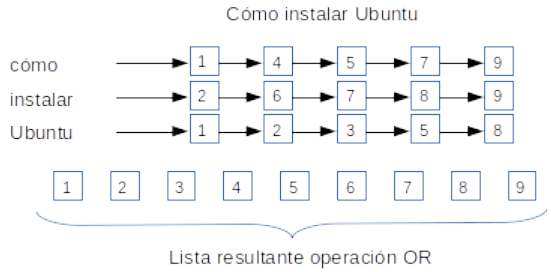

FIGURA 2.3: Operación OR

2.4 OPERACIONES SOBRE LISTAS INVERTIDAS

Cuando una *query* llega al motor de búsqueda, cada término tiene asociado una lista con todos los documentos en los cuales aparece dicho término. El sistema debe decidir qué documentos se analizarán para obtener la respuesa con el conjunto de los K mejores. A continuación se presentan las diferentes formas de operar los documentos pertenecientes a las listas invertidas de una *query*.

2.4.1 OR

Este operador toma las listas invertidas de cada uno de los términos de la *query* y ejecuta la disyunción entre ellas. El resultado de este operador es una lista invertida con todos los documentos que contengan al menos un término de la query. Finalmente, esta lista invertida se ocupará para obtener los mejores K documentos. Un ejemplo sencillo se muestra en la FIGURA 2.3.

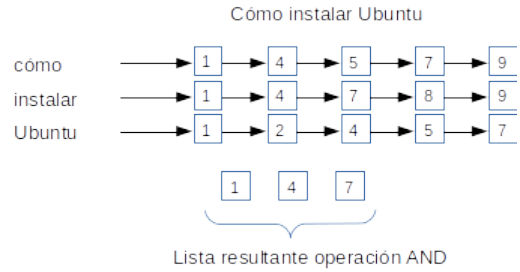

FIGURA 2.4: Operación AND

2.4.2 AND

Este operador ejecuta la conjunción entre las listas invertidas de los términos de una *query*. Se obtiene una lista invertida con los documentos que contengan todos los términos de la *query*. Se debe notar que aquí se obtiene una lista resultante de menor que la obtenida en el operador OR (Ver FIGURA 2.4).

2.4.3 WAND

CAPÍTULO 3. MODELO DE REPRESENTACIÓN DATOS PARA SR

En este capítulo se presenta un modelo para la representación de datos en SR. Se comienza con una revisión exhaustiva sobre la evolución de representaciones de datos para SR mostrando sus ventajas y desventajas. Posteriormente a partir de las deficiencias encontradas se propone un nuevo modelo. Finalmente, se realiza una discusión respecto al modelo propuesto en comparación a otros.

3.1 TRABAJO RELACIONADO

Los SR están en constante evolución y los atributos que participan en las interacciones entre los usuarios e ítems son cada vez más variadas (Palomino 2012). Las relaciones existentes entre la inteligencia colectiva y los SR no está clara. Cabe destacar que los SR son capaces de "capturar" la inteligencia colectiva para generar recomendaciones que son de utilidad para los usuarios. El área de los sistema colaborativos reconoce que las distintas actividades no son eventos aislados sino que se sitúan dentro de un contexto que informa su sentido y carácter (Suchman, 1987).

Los eventos representan la base de los SR que permite obtener relaciones entre usuarios e ítems y así realizar el cálculo de una recomendación para el usuario activo. Estos se basan principalmente en las interacciones de los usuarios hacia los ítems que son reflejadas en los sistemas mediante eventos.

Sea C el conjunto de usuarios, S el conjunto de ítems, un evento se define como la siguiente tupla:

$$e = \{(u, i, v) | s \in C, i \in S, v \, algún \, valor\} \tag{3.1}$$

El valor v corresponde a una preferencia del usuario y puede ser de distintos tipos dependiendo de las posibles interacciones del dominio de aplicación. Puede corresponder a una valoración numérica, comentario, etiqueta, *like*, etc. Por ejemplo, el valor de un evento *Rating* en *Movielens* se define en el dominio real entre 1 y 5. En otros casos como *Delicious*[1] el valor del evento pertenece al conjunto de combinaciones alfanuméricas que representa el *tag* que un usuario asigna a un *bookmark*. Se asume que una preferencia de un usuario a un ítem es única, luego se define E como el conjunto de todos los eventos de un sistema.

Ekstrand et al. (2011) en su herramienta para construir, investigar y estudiar SR de filtrado colaborativo *Lenskit*[2] utilizan una representación básica de eventos de *rating*. En su trabajo definen el concepto de historia de usuario que corresponde a un subconjunto de eventos ordenados temporalmente relacionados a un usuario específico. Formalmente $H = (\mathcal{P}(E), \prec)$ donde H es una historia para el usuario u, E el conjunto de todos los eventos y $\prec$ una relación de orden temporal entre los eventos.

La representación básica de evento que considera solo al usuario e ítem no permite:

- Describir un conjunto de interacciones sobre el mismo ítem (Tareen et al., 2010), por ejemplo no se puede representar un *rating* y *tag* sobre un mismo ítem. Disponer de una representación de este tipo permite a los SR aprovechar los diversos comportamientos que tienen los usuarios hacia un mismo ítem.

- El evento posee escasa información referente al contexto donde fue efectuado. Un tipo de información que es representada es una marca temporal del evento que permite situarlo en el tiempo, entregando información al SR sobre el contexto (Adomavicius & Tuzhilin,

[1] https://delicious.com/
[2] http://lenskit.grouplens.org/

2011).

- Modelar la información social referente a la Web 2.0 que proporciona nuevas formas de interacción hacia los ítems (Bobadilla et al., 2013).

Dado lo anterior aparecen trabajos que tratan de abordar estas dificultades. Adomavicius & Tuzhilin (2001) presentan un modelo multidimensional basado en *data warehousing* que permite representar dimensiones adicionales a las clásicas usuario e ítem. El modelo supone que una interacción entre usuario e ítem depende de múltiples dimensiones. En este trabajo se basan en eventos de valoración numérica, luego definen la función R:

$$R : D_1 \times D_2 \times D_3 \times D_4 \times \cdots \times D_n \rightarrow ratings \tag{3.2}$$

Este modelo resuelve el problema de utilizar data multi-dimensional en SR y permite situar las interacciones de un usuario en distintos contextos. Sin embargo, su implementación es compleja debido a la utilización de agregaciones jerárquicas (**OLAP**) y un lenguaje de consultas llamado **RQL** (Palomino, 2012).

Varios trabajos señalan la importancia de utilizar información contextual en el proceso de recomendación (Adomavicius et al., 2005), (Adomavicius & Tuzhilin, 2001), (Adomavicius & Tuzhilin, 2011), (Palomino, 2012) y (Yujie & Licai, 2010). Dado lo anterior, Adomavicius & Tuzhilin (2011) definen los **CARS** *(Context Aware Recommender Systems)* (véase sección **??**) como SR que utilizan información contextual para la predicción de valoraciones numéricas. Formalmente se define la función de R:

$$R : User \times Item \times Contexto \rightarrow ratings \tag{3.3}$$

Tanto la representación que realizan Adomavicius & Tuzhilin (2001) en la definición (3.2) y Adomavicius & Tuzhilin (2011) en la definición (3.3) son válidas solo para SR basados en eventos de valoración numérica(*ratings*). Sin embargo, la aparición de la Web 2.0 ha otorgado

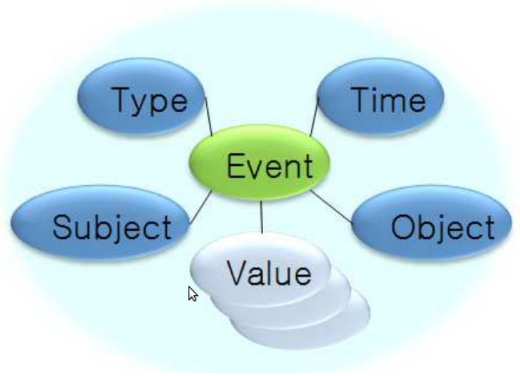

FIGURA 3.1: Modelo conceptual de *Synergy* tomado de (Tareen et al., 2010)

nuevas formas de interacción para los usuarios como etiquetas (*tagging*), *liking*, *commenting*, *following*, etc. En particular, la información referente a los *tags* se ha modelado mediante la extensión de la matriz item-usuario a un cubo ítems-usuarios-tags (Song et al., 2011).

Con el objetivo de modelar las nuevas interacciones, Tareen et al. (2010) presentan *Synergy* una herramienta para la creación, ejecución y comparación de diferentes algoritmos de filtrado colaborativo. *Synergy* esta construido en base a dos tipos de eventos *rating* y *tagging*, permitiendo la hibridación de algoritmos utilizando estos dos tipos de eventos. Para modelar diversas formas de interacción y abstraerse del dominio de aplicación proponen el modelo conceptual de la Figura 3.1.

El modelo de *Synergy* se define de la siguiente forma:

$$
\begin{aligned}
U &= \{u_1, u_2, \cdots, u_n\} \\
C &= \{c_1, c_2, \cdots, c_n\} \\
I &= \{i_1, i_2, \cdots, i_n\} \\
E &= \{(s, p, o, t) | s \in U, p \in I, o \in C, \quad t\, es\, el\, tiempo\} \\
V &= \{(e, a) | e \in E, a\, es\, algún\, valor\}
\end{aligned}
\tag{3.4}
$$

Donde U es el conjunto de todos los usuarios, C de todos los ítems, I de todas las posibles interacciones, E de eventos y V los valores que puede tomar un evento. Este modelo conceptual

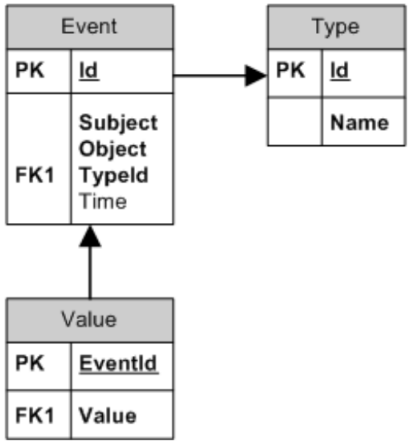

FIGURA 3.2: Diagrama entidad-relación *Synergy* tomado de (Tareen et al., 2010)

es plasmado en el diagrama entidad-relación de la Figura 3.2 que permite almacenar los datos dentro de una base de datos relacional. La principal ventaja de este modelo es su flexibilidad para describir varios tipos de interacciones, sin embargo no permite situarlas dentro de un contexto espacial y social.

Palomino (2012) presenta un modelo conceptual que extiende de *Synergy* basándose en el *framework* conceptual *3-Ontology*. En este modelo sitúa los eventos dentro de un lugar y una comunidad. En la definición 3.5 se define formalmente la representación, donde S son las comunidades. En la Figura 3.3 se muestra el modelo conceptual propuesto por Palomino (2012). Al igual que en *Synergy* se desarrolla un diagrama entidad-relación para almacenar la información (véase Figura 3.4).

$$
\begin{aligned}
U &= \{u_1, u_2, \cdots, u_n\} \\
C &= \{c_1, c_2, \cdots, c_n\} \\
I &= \{i_1, i_2, \cdots, i_n\} \\
S &= \{s_1, s_2, \cdots, s_n | S \subseteq U\} \\
E &= \{(s, p, o, t, y, l) | s \in U, p \in I, o \in C, t\, es\, el\, tiempo, y \in S, l\, es\, el\, lugar\} \\
V &= \{(e, a) | e \in E, a\, es\, algún\, valor\}
\end{aligned}
\tag{3.5}
$$

FIGURA 3.3: Modelo conceptual de *Synergy* tomado de (Palomino, 2012)

El trabajo de Palomino permite describir las interacciones que suceden dentro de una red social basado en el *framework* conceptual *3-ontology*. Sin embargo, el modelo presenta las siguientes deficiencias:

- Es solo un modelo conceptual que no posee una implementación concreta.
- No permite modelar los perfiles del usuario, ítem, comunidad y lugar.
- No modela las tres formas representaciones de la *3-Ontology* trazas, retratos y mapas.
- No modela explícitamente las relaciones existentes entre los tres contenedores de sentido de la *3-ontology*.

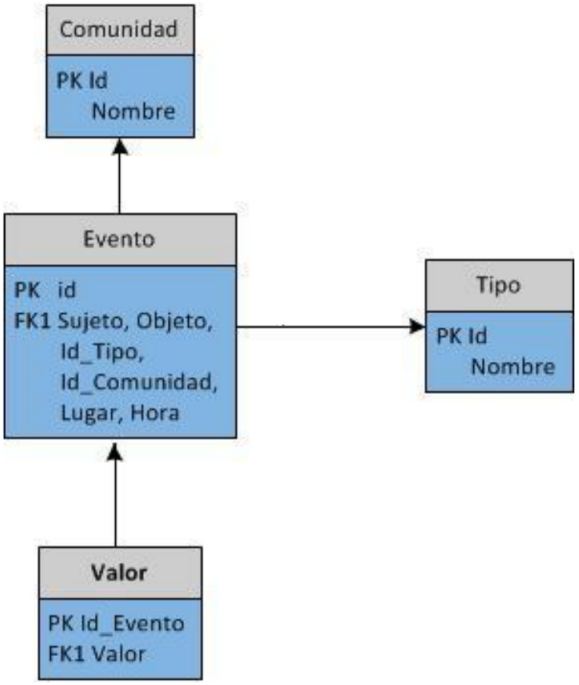

FIGURA 3.4: Diagrama entidad-relación tomado de Palomino (2012)

3.2 MODELO DE REPRESENTACIÓN DE INTERACCIONES PARA SR BASADO EN LA 3-ONTOLOGY

En esta sección se presenta un modelo de representación de interacciones para SR basado en la *3-Ontology* que tiene como objetivo modelar las características de los SR modernos.

3.2.1 Contenedores de sentido de la 3-Ontology

Los SR utilizan la información de ambientes colaborativos donde los usuarios interactúan entre sí y con ítems. Dado esto el *framework 3-Ontology* define tres contenedores de sentido

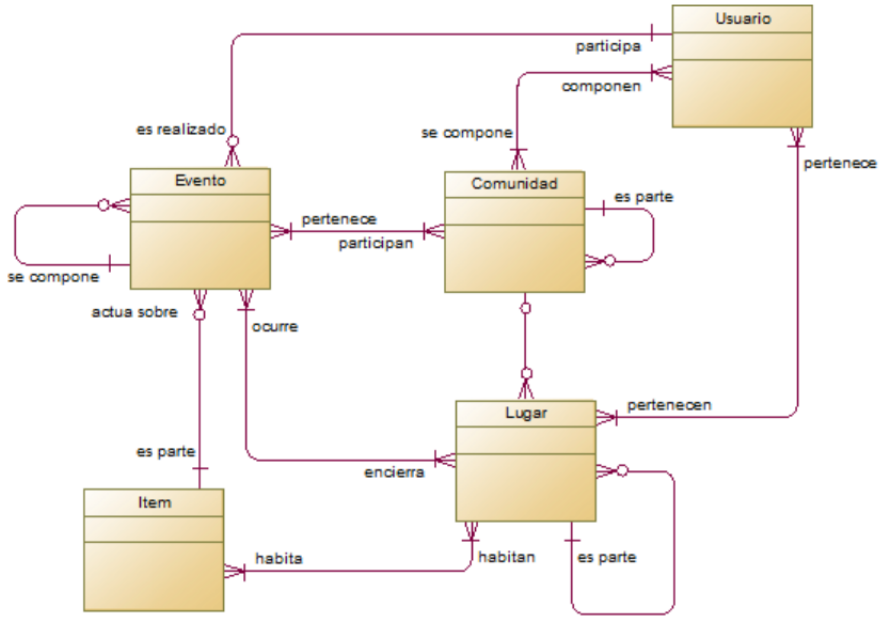

FIGURA 3.5: Modelo conceptual propuesto

eventos, comunidades y lugares (véase sección **??**) para situar las interacciones dentro de un contexto social, temporal y espacial. El modelo conceptual propuesto se presenta en el diagrama entidad-relación de la Figura 3.5. Se pueden observar las relaciones existentes entre los elementos básicos de los SR y los contenedores de sentido de la *3-Ontology*.

A continuación se definen formalmente cada uno de los elementos básicos usuario e ítem y los contenedores de sentido de la *3-Ontology*.

3.2.1.1 Usuarios e ítems

Los usuarios son quienes asignan su preferencia a un ítem. Esta puede ser de distintos tipos dependiendo del dominio de aplicación. Cada usuario tiene un conjunto de atributos como

nombre, fecha de nacimiento, información demográfica, en general. Los primeros SR se basaban en este conjunto de atributos para realizar recomendaciones.

Se define formalmente un usuario u, donde $attr_i^u$ es un atributo del usuario que posee un conjunto de valores dependiendo del dominio de aplicación:

$$u = (attr_1^u,\ attr_2^u,\ attr_3^u, \cdots,\ attr_n^u) \tag{3.6}$$

Los ítems son objetos a los cuales el usuario asigna una preferencia. Son de distinto tipo y dependen del dominio de aplicación, por ejemplo en *Movielens* son películas, en *Delicious bookmarks*, en *Lastfm* canciones y en *Amazon* productos. Poseen un conjunto de atributos como en el caso de las películas como nombre, director, actor principal, etc. Los primeros SR se basaban en estos atributos para realizar recomendaciones.

Se define formalmente un ítem c, donde $attr_i^c$ es un atributo del ítem que posee un conjunto de valores dependiendo del dominio de aplicación:

$$c = (attr_1^c,\ attr_2^c,\ attr_3^c, \cdots,\ attr_n^c) \tag{3.7}$$

3.2.1.2 Lugares

Según la *3-Ontology* un lugar corresponde a un espacio físico y/o virtual donde habitan los usuarios e ítems y ocurren las interacciones. En el caso de los SR se simplifica definiendo un lugar como un conjunto de ítems y el espacio donde ocurren las interacciones. La información referente a los lugares puede ser usada en SR que utilicen el contexto espacial de los eventos (Adomavicius & Tuzhilin, 2011), (Panagiotis et al., 2011), (Bobadilla et al., 2013). Sea $C = \{c_1, c_2, \cdots, c_n\}$

el conjunto de todos ítems y $A = (attr_1^{\boldsymbol{l}}, attr_2^{\boldsymbol{l}}, attr_3^{\boldsymbol{l}}, \cdots, attr_n^{\boldsymbol{l}})$ el conjunto de atributos de un lugar, entonces un lugar $\boldsymbol{l}$ se define como la siguiente estructura:

$$\boldsymbol{l} = (\mathcal{P}(C), List(attr^{\boldsymbol{l}})) \tag{3.8}$$

Los ítems que habitan un lugar pueden ser de distinto tipo y los atributos corresponden a características propias del lugar como coordenadas geográficas, nombre e identificador.

3.2.1.3 Comunidades

Según la *3-ontology* una comunidad corresponde a un conjunto de usuarios que habitan un lugar y son protagonistas de los eventos. La información referente a la comunidad puede ser usada en SR que utilicen información de comunidades emergentes, reputación y confiabilidad entre usuarios (Victor et al., 2011). Sea $U = \{u_1, u_2, \cdots, u_n\}$ el conjunto de todos usuarios y $A = (attr_1^{\boldsymbol{c}}, attr_2^{\boldsymbol{c}}, attr_3^{\boldsymbol{c}}, \cdots, attr_n^{\boldsymbol{c}})$ el conjunto de atributos de una comunidad, entonces una comunidad $\boldsymbol{c}$ se define:

$$\boldsymbol{c} = (\mathcal{P}(U), List(attr^{\boldsymbol{c}})) \tag{3.9}$$

Los usuarios pueden pertenecer a una comunidad de forma explícita y/o implícita como se mencionó en el capítulo 2 y los atributos pueden corresponder a características como nombre, identificador y dirección virtual.

3.2.1.4 Eventos

Según la *3-Ontology* los eventos son de carácter colaborativo y están situados en un tiempo y lugar. Pueden ser atómicos o compuestos (conjunto de eventos que tienen un significado común). En el caso de los SR se considera solo eventos atómicos. La definición propuesta (3.10) de un evento extiende de las presentadas por Tareen et al. (2010) y Palomino (2012) agregando el concepto de situar al usuario dentro de una comunidad y al ítem en un lugar. Además esta definición permite modelar diversos tipos de interacciones sobre el mismo ítem. Por lo tanto, esta representación permite modelar información del contexto social, temporal y espacial. Características emergentes de los SR (Adomavicius & Tuzhilin, 2005), (Adomavicius & Tuzhilin, 2011), (Palomino, 2012), (Bobadilla et al., 2013). R corresponde a un retrato (representación de las comunidades) y M a un mapa (representación de los lugares). U es el conjunto usuarios del sistema, y C el de ítems. $3onto$ es un *functor* que permite situar los eventos en una comunidad, lugar y/o tiempo.

$$I = \{i_1, i_2, \cdots, i_n\}$$
$$\boldsymbol{e} = \{(u, c, i, v, 3onto(co, l, t) | u \in U, c \in C, i \in I, co \in R, l \in M, t\, es\, el\, tiempo, v\, es\, algún\, valor\} \tag{3.10}$$

3.2.2 Meta-modelo de la 3-Ontology para SR

La *3-Ontology* como se especifico en la sección **??** construye una representación que corresponde a un meta-modelo de relaciones entre los contenedores de sentido. En la Figura 3.6 se describe el modelo de nivel base representado por el plano inferior donde habitan los

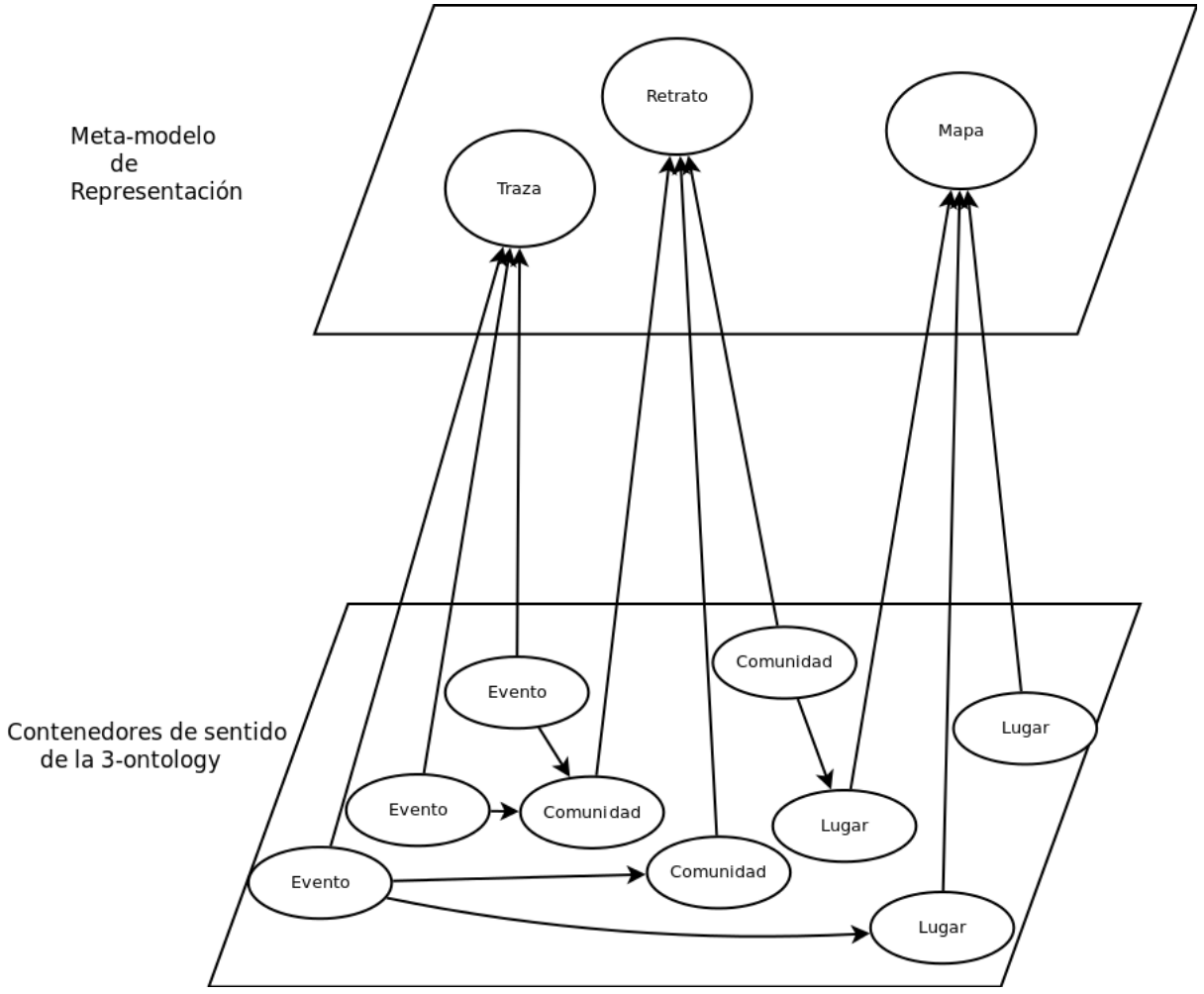

FIGURA 3.6: Meta-modelo de la *3-Ontology*

contenedores de sentido, y en el plano superior el meta-modelo donde se encuentran las trazas, mapas y retratos. Además, es importante señalar que los usuarios e ítems son parte de las comunidades y lugares respectivamente.

Los SR utilizan información referente sobre la interacción entre usuarios e ítems. En la Figura 3.7 se muestra el flujo de datos propuesto para la construcción de un SR de películas basándose en la representación de la *3-Ontology*. Se propone que la entrada de un SR son una traza, un retrato y un mapa. Luego se realiza un conjunto de operaciones pertenecientes a la lógica de la *3-Ontology* que permite obtener los eventos, lugares y comunidades de interés

para el cálculo de la recomendación. Posteriormente, el SR procesa la información para generar comunidades (recomendación de personas y/o grupos) e ítems (recomendación de ítems). Para el caso particular de recomendación de películas basado en eventos de tipo *rating* para *Movielens* se tendrían las siguientes pasos:

1. Definir la traza como todos de eventos de tipo *rating* y *tagging* del sistema, una comunidad como todos los usuarios que han colocado un *rating* o *tag* a una o varias películas y un mapa como todas las películas que han sido sujeto de interacción del usuario.

2. Aplicar un conjunto de operaciones de la lógica de la *3-Ontology* para obtener solo los eventos de tipo *rating*, usuarios y películas relevantes para el cálculo de la recomendación para un usuario activo.

3. El SR utilizando alguna heurística o algoritmo de predicción realiza el cálculo de la preferencia que colocaría el usuario activo. Se seleccionan los ítems con mayor valor de preferencia para ser recomendados.

Es importante señalar que la lógica de la *3-ontology* provee métodos que apoyan a dos de las etapas descritas por Adomavicius & Tuzhilin (2011) para los CARS como son las de pre y post-filtrado de eventos permitiendo mejorar el proceso de recomendación mediante el uso de información contextual. Para el pre-filtrado se proveen métodos de obtención de eventos dado algún patrón temporal como días de la semana y fines de semana. Por otro lado, una vez calculada la lista de recomendaciones se puede post-filtrar por algún criterio contextual. Otra forma es pre-filtrar es por tipo de eventos dependiendo de las posibles interacciones que la aplicación provee al usuario (por ejemplo utilizar solo los eventos de tipo *tagging*).

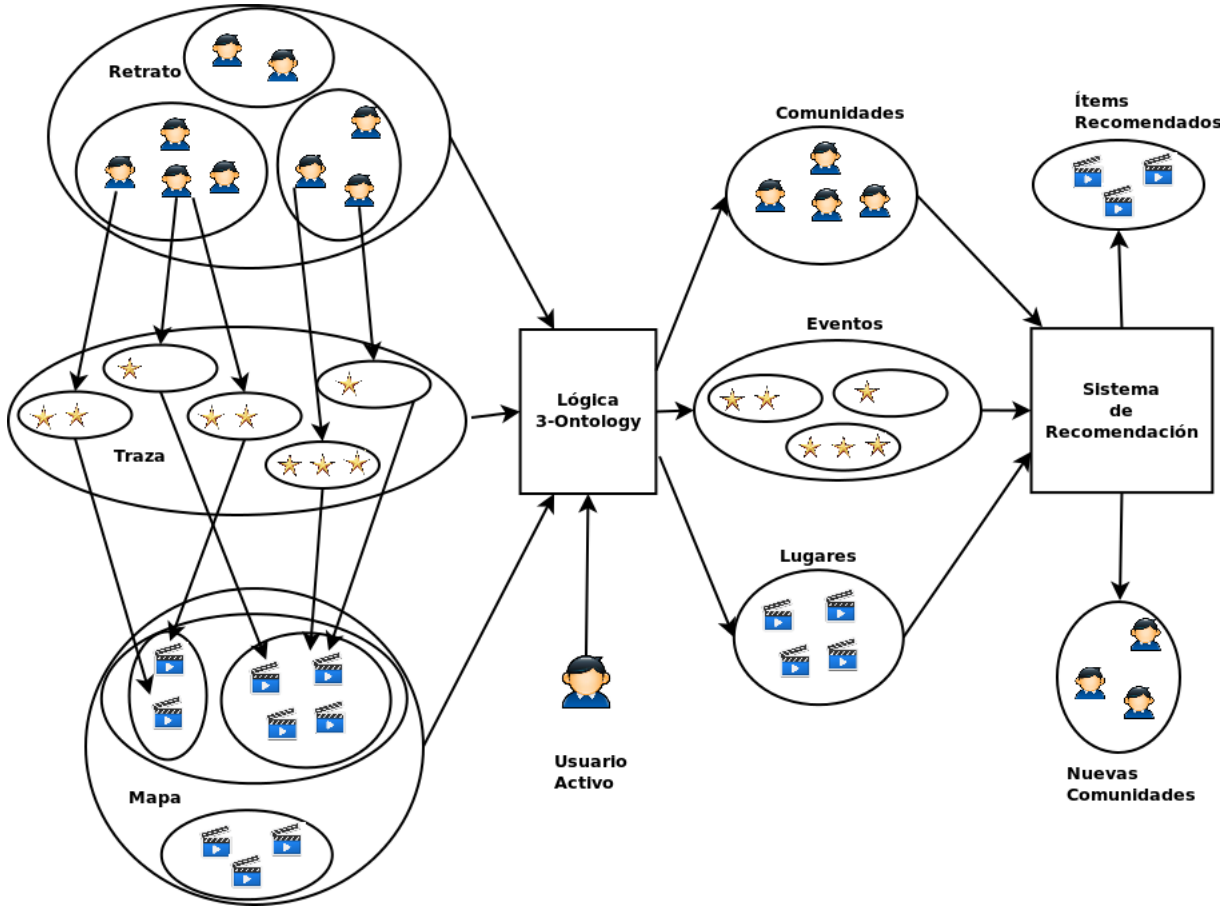

FIGURA 3.7: Flujo de datos propuesto

3.2.2.1 Trazas

Las trazas (ver sección **??**) corresponden a una representación de eventos relacionados temporalmente. Formalmente una traza T_r se define como una estructura con una relación de orden parcial $\prec$ sobre el conjunto potencia de los eventos E.

$$T_r = (\mathcal{P}(E), \prec) \tag{3.11}$$

Las funciones que se definen sobre las trazas para obtener las relaciones entre los eventos pueden ser temporales, por usuario, tipo de interacción e ítem. Las relaciones temporales entre eventos se definen formalmente como una función f_t con dominio en el tiempo T y recorrido el conjunto potencia de una traza T_r. Esto permite obtener subconjuntos de eventos relacionados temporalmente.

$$\begin{gathered} f_t \colon T \to \mathcal{P}(T_r) \\ t \to f_t(t) \end{gathered} \tag{3.12}$$

Las relaciones de usuarios con eventos se definen formalmente como una función f_u denominada historia de usuario con dominio en el conjunto de usuarios U y recorrido en el conjunto potencia de una traza T_r. Esto permite obtener historias de usuario que son definidas por Ekstrand et al. (2011) como un conjunto de eventos relacionados con un usuario.

$$\begin{gathered} f_u \colon U \to \mathcal{P}(T_r) \\ u \to f_u(u) \end{gathered} \tag{3.13}$$

Las relaciones de interacciones con eventos se definen formalmente como una función f_i con dominio en el conjunto de posibles interacciones I y recorrido en el conjunto potencia de una traza T_r. Esto permite obtener eventos relacionados con un tipo de interacción específico, permitiendo modelar diversas formas de interacción de un usuario con un ítem.

$$\begin{aligned} f_i &: I \to \mathcal{P}(T_r) \\ i &\to f_i(i) \end{aligned} \tag{3.14}$$

Las relaciones de ítems con eventos se definen formalmente como una función f_c con dominio en el conjunto de ítems C y recorrido en el conjunto potencia de una traza T_r. Esto permite obtener eventos relacionados con un ítem específico.

$$\begin{aligned} f_c &: C \to \mathcal{P}(T_r) \\ c &\to f_c(c) \end{aligned} \tag{3.15}$$

3.2.2.2 Retratos

Los retratos (ver sección **??**) corresponden a una representación de las comunidades. Formalmente un retrato R se define como una estructura con una relación de composición $\prec$ sobre el conjunto potencia de comunidades C.

$$R = (\mathcal{P}(C), \prec) \tag{3.16}$$

La funciones construidas sobre los retratos permiten obtener relaciones entre los usuarios y comunidades. Se define la función f_u con dominio en el conjunto de usuarios U y recorrido en el conjunto potencia de los retratos R. Esto permite conocer a que comunidades pertenece un usuario u.

$$\begin{aligned} f_u &: U \to \mathcal{P}(R) \\ u &\to f_u(u) \end{aligned} \tag{3.17}$$

3.2.2.3 Mapas

Los mapas (ver sección **??**) corresponden a una representación de los lugares relacionados de forma física y/o virtual. Formalmente un mapa M se define como una estructura con una relación de composición $\prec$ sobre el conjunto potencia de los lugares L.

$$M = (\mathcal{P}(L), \prec) \tag{3.18}$$

Las funciones que se construyen sobre los mapas permiten obtener relaciones entre los ítems y lugares. Se define la función f_c con dominio en el conjunto de ítems U y recorrido en el conjunto potencia de los retratos M. Esto permite conocer a que lugares pertenece un ítem c.

$$\begin{aligned} f_c &: C \to \mathcal{P}(M) \\ c &\to f_c(c) \end{aligned} \tag{3.19}$$

3.3 DISCUSIÓN SOBRE LA *3-ONTOLOGY* PARA LA REPRESENTACIÓN DE DATOS EN SR

Para realizar una discusión sobre la representación de datos propuesta se presentan las siguientes criterios y el grado de cumplimiento que provee la representación propuesta.

- **Independencia del dominio**: se refiere a que los SR son independientes del dominio de aplicación. El meta-modelo propuesto de trazas, retratos y mapas permite la abstracción del dominio de aplicación.

- **Perfilamiento de usuarios e ítems**: se refiere a que los SR pueden utilizar información sobre el perfil de los usuarios e ítems.

- **Modelamiento de interacción**: se permite modelar y obtener información sobre distintos tipos interacciones de los usuarios hacia los ítems. La definición propuesta de evento soporta la mayoría de formas de interacción en la Web 2.0.

- **Modelamiento del contexto**: se refiere a la capacidad del SR de obtener información sobre el contexto de la interacción. Esto se logra situando la interacción en un tiempo, lugar y comunidad.

- **Modelamiento de la comunidad**: se refiere a la capacidad del SR de obtener información sobre la comunidad del usuario como es la reputación y confiabilidad de los usuarios. Dado que los usuarios pertenecen a una comunidad, se pueden determinar medidas de confiabilidad de la recomendación.

- **Modelamiento del espacio**: se refiere a que el SR puede utilizar información referente al espacio tanto físico como virtual. Dado que la representación sitúa al ítem dentro de un lugar, se puede agregar esta información dentro del proceso de recomendación.

Dado lo anterior se puede deducir que la capacidad de representación de la *3-ontology* permite modelar la interacción en dominios colaborativos que son el insumo de los SR. En la tabla 3.1 se muestra una comparación con otros modelos presentados en la sección 3.1. En esta tabla se observa que el modelo de Tareen et al. (2010) permite la independencia del contexto y modelar diversas formas de interacción. De esta forma solo se pueden desarrollar SR de filtrado colaborativo que no utilicen información sobre el usuario e ítem en el proceso de recomendación. Por otro lado, el modelo de Palomino (2012) a pesar de cumplir con casi todos los criterios de evaluación no tiene una implementación concreta que asegure la eficacia del modelo, solo posee la definición de evento y no construye un meta-modelo de representación sobre los contenedores

de sentido que los doten de lógica. Finalmente, cabe destacar que el modelo propuesto demuestra la capacidad de la *3-Ontology* como meta-modelo para representar ambientes colaborativos.

TABLA 3.1: Comparación con otros modelos propuestos

	Modelos			
Características	Modelo OLAP Adomavicius & Tuzhilin (2001)	Modelo de Tareen et al. (2010)	Modelo de Palomino (2012)	Modelo Propuesto
Independencia del dominio		X	X	X
Perfilamiento de usuarios e ítems	X			X
Modelamiento de interacción			X	X
Modelamiento del contexto	X	X	X	X
Modelamiento de la comunidad			X	X
Modelamiento del espacio			X	X

Por otro lado este modelo resuelve varios de los problemas abordados en este trabajo de tesis:

1. *Los cambios en la Web 2.0 y la forma en que los usuarios interactúan con esta. Además, existe dificultad para modelar múltiples interacciones para que estas sean usadas por los SR (Tareen et al., 2010)*: el modelo propuesto en este trabajo permite representar cualquier forma de interacción usuario-ítem en una aplicación colaborativa. Esto se logra gracias a la definición de evento propuesta que acepta cualquier valor (real, numérico, nominal, etc) entre la interacción usuario-ítem.

2. *Los SR para obtener mejores resultados sobre un dominio específico se acopla demasiado a la data (Tareen et al., 2010). Esto provoca una sobre-especialización del SR, dificultando la posibilidad de reutilizar la solución en otro dominio de aplicación dentro de la empresa*: el modelo presentado permite representar los datos bajo un esquema común que logra un bajo acoplamiento hacia los algoritmos de recomendación subyacentes. Esto permite utilizar distintos algoritmos de recomendación sin la necesidad de realizar una representación distinta de los datos.

3. *Un SR puede degradar sus resultados en el tiempo debido a cambios en la data referente a la interacción de los usuarios por lo que debería ser cambiado por otro*: el modelo presentado permite re-entrenar algoritmos mediante la misma representación basada en la *3-ontology* utilizando otros datos asegurando así una mejora continua del rendimiento de los SR.

4. *Las mejoras obtenidas en los algoritmos son difíciles de generalizar para todos los usuarios del sistema, por ejemplo algunos usuarios pueden usar un campo muy reducido de opciones de interacción y otros más*: el modelo permite obtener mediante las trazas eventos relevantes a un usuario específico y realizar el cálculo de la recomendación, de esta forma se puede determinar que SR utilizar dado el comportamiento del usuario y el tipo de interacción que más utiliza.

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN DE RBOX 2.0

En este capítulo se presenta el diseño e implementación de RBOX 2.0 una herramienta de software construida en *Java* basada en el modelo de representación de datos propuesto en este trabajo de tesis.

El *framework* esta construido en *Java SE 7* por las siguientes razones:

- **Popularidad**: según el *ranking* presentado por **TIOBE**[1] mensualmente *Java* es el segundo lenguaje más popular, precedido sólo por C.
- **Portabilidad**: las aplicaciones construidas en *Java* son ejecutadas bajo su máquina virtual. Por lo tanto pueden ejecutarse en cualquier dispositivo físico que posea la máquina virtual.
- **Gran cantidad de librerías**: en *Java* se proveen varias librerías que agilizan el desarrollo como por ejemplo *Apache Commons*[2], *fastutils*[3], etc. En el caso de SR se proveen varias librerías de minería de datos y aprendizaje de máquina tales como: *Weka*[4], *Apache Mahout*[5], *Mallet*[6] entre otras.
- **Integración con ambientes empresariales**: la versión empresarial *Java EE 6*[7] provee un estándar para la construcción de aplicaciones empresariales.

[1] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
[2] http://commons.apache.org/
[3] http://fastutil.di.unimi.it/
[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://mahout.apache.org/
[6] http://mallet.cs.umass.edu/
[7] http://www.oracle.com/technetwork/java/javaee/overview/index.html

4.1 ARQUITECTURA DE RBOX 2.0

RBOX[8] 2.0 es una herramienta para el diseño e implementación de SR basados en el modelo de representación de datos propuesto que permite que los SR tengan la capacidad de situar los eventos.

En la Figura 4.1 se muestra la arquitectura de RBOX 2.0 donde se observan los principales componentes de la herramienta.

- **Representación de datos**: se compone de dos componentes que representan los contenedores de sentido y la lógica de la *3-Ontology*.
- **Algoritmos de recomendación**: pueden estar construidos con herramientas de construcción de SR de filtrado colaborativo como *Lenskit* o de minería de datos como *Weka*.
- **RBOX Recommender**: corresponden a los componentes que permiten dar el servicio de recomendación.

RBOX 2.0 se compone de los módulos API y CORE el primero establece la definición de las interfaces y el segundo una implementación para la construcción de SR. En la Figura 4.2 se presenta un diagrama de paquetes de RBOX 2.0 con sus dos módulos principales y sub-módulos.

[8]Acrónimo de **R**ecommendation **BOX**

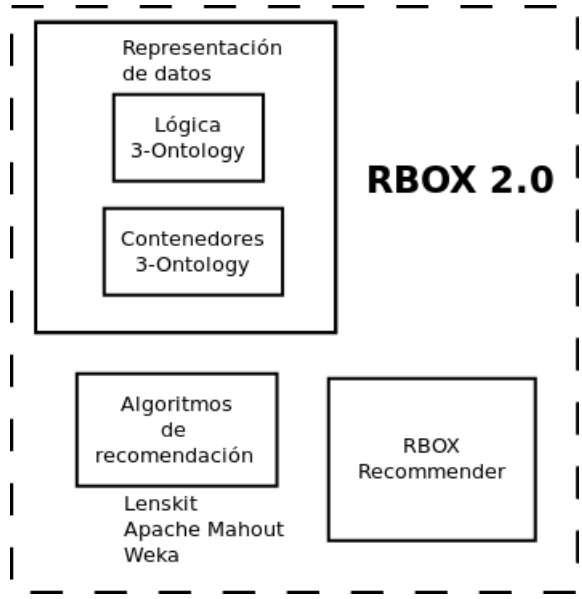

FIGURA 4.1: Arquitectura RBOX 2.0

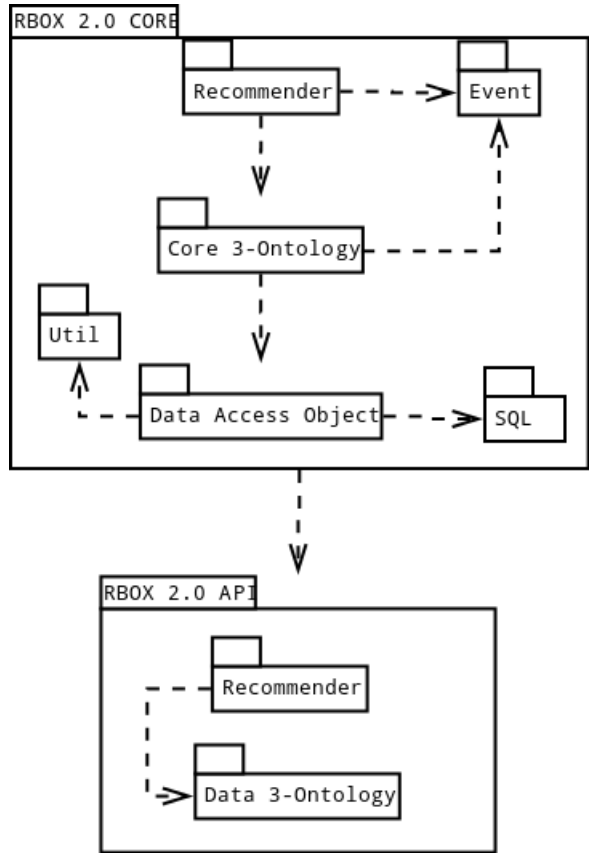

FIGURA 4.2: Diagrama de paquetes de RBOX 2.0

4.1.1 RBOX 2.0 API

Corresponde a la definición de las interfaces que permiten la representación de datos *(data 3-ontology)* y construcción de *(recommender)*. Este módulo se compone de los siguientes sub-módulos:

- **Recommender**: corresponde a la definición de las interfaces que permiten construir un recomendador.
- **Data 3-Ontology**: corresponde a la definición de las interfaces que representan los contenedores de sentido y la lógica de la *3-Ontology*.

4.1.2 RBOX 2.0 CORE

Corresponde a una implementación del API y un conjunto de módulos que permiten construir un SR. Este módulo se compone de los siguientes sub-módulos:

- **Recommender**: provee una implementación de las interfaces de recomendación.
- **Event**: implementación de componentes para la representación de distintos tipos de eventos.
- **Core 3-Ontology**: componentes que describen el comportamiento del *framework 3-ontology*.
- **Utils**: componentes utilitarios como por ejemplo de lectura de archivos.

- **Data access object**: definición e implementación de los componentes DAO definidos.
- **SQL**: definición e implementación de componentes para el acceso a base de datos relacionales mediante JDBC.

4.2 DISEÑO DE CLASES DE RBOX 2.0

En esta sección se realiza la definición de las clases utilizadas que representan el modelo propuesto en el capítulo 3.

4.2.1 Contenedores de sentido

La *3-ontology* define tres contenedores de sentido eventos, comunidades y lugares. Cada uno de estos conceptos es representado mediante una interfaz para proveer la flexibilidad y extensibilidad requerida por los SR. Por otro lado, se presentan las representaciones de dos conceptos básicos para un SR como son el usuario e ítem.

En el código 4.1 se presenta la definición de la interfaz que representa a un evento que puede ser extendida a nuevos tipos de eventos dado que el valor es de tipo genérico. Se proveen implementaciones para dos eventos interactivos y sociales:

- **Rating**: en este caso el valor es de tipo real bajo un dominio determinado.
- **Tagging**: en este caso el valor es una cadena de caracteres (*String*).

```
1  public interface Event {
2
3     long getUserId();
4
5     long getItemId();
6
7     long getTimestamp();
8  dos
9     long getPlaceId();
10
11    long getCommunityId();
12
13    <T> T getValue();
14 }
```

CÓDIGO 4.1: Interfaz Event

```
1  public interface Community extends List<User>{
2
3      long getId();
4
5      String getName();
6
7      LongSet getUserIds();
8  }
```

CÓDIGO 4.2: Interfaz Community

En el código 4.2 se muestra la definición de la interfaz que representa a una comunidad. Esta hereda de *List* y se compone de usuarios. Un usuario puede explícitamente indicar su pertenencia a una comunidad implícitamente o se puede deducir dado su perfil e historial de eventos.

En el código 4.3 se muestra la definición de la interfaz que representa a un lugar. Esta hereda de *List* y se compone de ítems. Un lugar corresponde al espacio físico y/o virtual donde habitan los ítems y son realizados los eventos.

Un usuario es el protagonista de un evento, pertenece a una o varias comunidades. La definición propuesta en este trabajo identifica al usuario en la interfaz del código 4.4.

```java
public interface Place extends List<Item> {

    long getId();

    String getName();

    LongSet getItemIds();
}
```

CÓDIGO 4.3: Interfaz Place

```java
public interface User {

    long getId();

    String getName();

    LongSet getCommunityIds();
}
```

CÓDIGO 4.4: Interfaz User

En el código 4.5 se muestra la interfaz que representa a un ítem. Este corresponde al objeto que es valorado por un usuario y puede habitar uno y/o varios lugares.

4.3 DISEÑO DE UNA CAPA DE ACCESO A DATOS

El diseño aplicado para representar la *3-Ontology* se constituye de 3 capas lógicas (*tiers*) presentadas en la Figura 4.3, la primera consiste en repositorios de datos donde se almacena la información relacionada a los 3 contenedores de sentido de la *3-Ontology* (lugares, eventos y comunidades). Sobre estos repositorios de datos se construye un conjunto de operaciones de

```
1  public interface Item {
2
3      long getId();
4
5      String getName();
6
7      String getContent();
8
9      LongSet getPlaceIds();
10 }
```

CÓDIGO 4.5: Interfaz Item

FIGURA 4.3: Capa de datos de RBOX 2.0

acceso a los datos. Esto permite obtener de forma unificada los eventos, lugares, comunidades y relaciones existentes entre ellos. Finalmente, se construye una capa de lógica que permite situar los eventos.

- **Repositorios de datos**: corresponde a un conjunto de repositorios que pueden contener información de utilidad para los SR.
- **Operaciones de acceso a datos**: proveen un acceso a los datos almacenados dentro de múltiples repositorios. Dado que la capa es construida con el patrón DAO se provee abstracción al tipo de repositorio de datos usado.
- **Lógica 3-Ontology**: provee un conjunto de operaciones que permiten obtener

información relevante para las distintas fases de construcción de un SR.

4.3.1 Alternativas de solución para distintos tipos de repositorios de datos

Los repositorios donde están albergados los datos de relevancia de los SR pueden estar dispersos en distintas plataformas, por ejemplo: los datos referentes a las transacciones de los usuarios pueden estar albergadas en una base de datos de análisis (OLAP). Por otro lado, se puede realizar consultas de información referente a las preferencias de sus usuarios dentro de las redes sociales y almacenar en algún repositorio, también se pueden obtener datos geo-localizados de interacciones que los usuarios realicen en alguna aplicación móvil.

A continuación se presentan dos esquemas posibles de solución dependiendo de las necesidades de construcción de un SR.

En un primer esquema se asume que la información proviene de distintos repositorios que pueden ser bases de datos relacionales, objetuales, analíticas, archivos de texto, xml, etc. Cada uno de estos repositorios tiene información referente a los tres contenedores de la *3-ontology* eventos, comunidades y lugares. En este caso, basta solo con implementar la capa **DAO** para los repositorios específicos. En la Figura 4.4 se observa un diagrama explicitando la arquitectura propuesta.

En un segundo esquema se provee un modelo de datos relacional que es capaz de albergar la información de utilidad de un SR. En este caso se asume que se debe realizar una consolidación de los datos de los diversos repositorios para que sean mapeados dentro de la base de datos relacional de la *3-ontology*. Molins (2012) provee una herramienta que permite realizar el mapeo desde bases de datos relacionales, archivos de texto y xml al esquema de datos de *Synergy*. En el trabajo futuro expone las directrices para modificar su herramienta y aplicarla a un

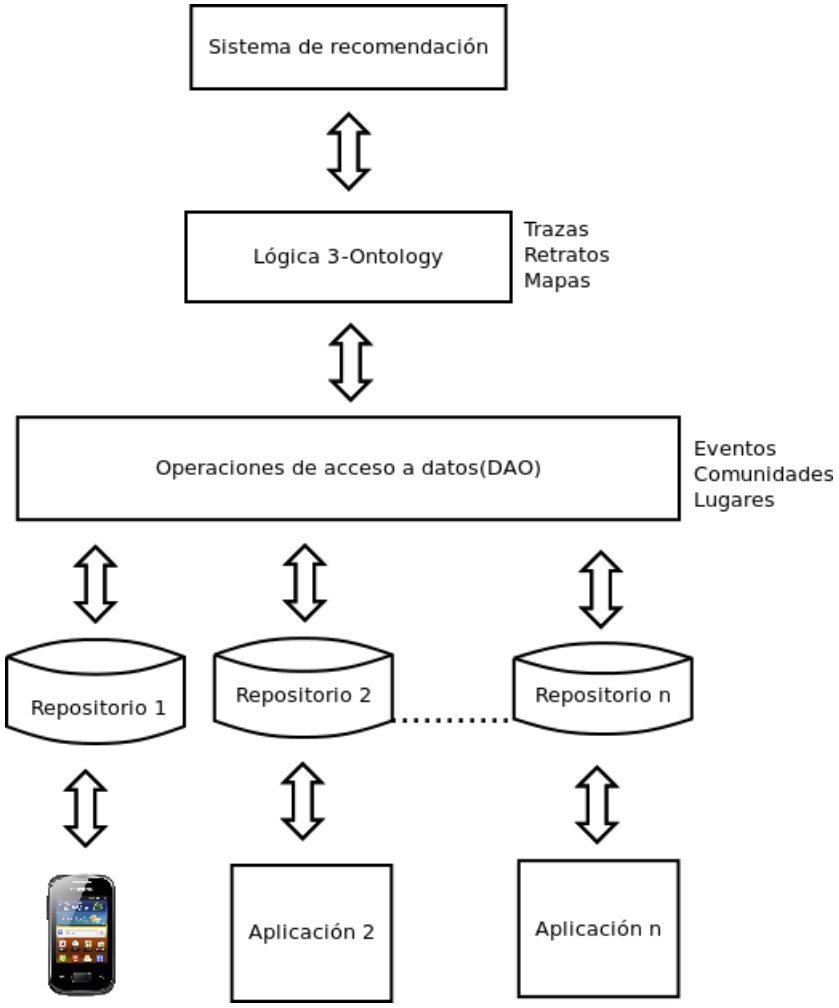

FIGURA 4.4: Múltiples repositorios

esquema destino distinto como es el de la *3-ontology*. En la Figura 4.5 se muestra la arquitectura propuesta para el segundo caso propuesto.

4.3.2 Operaciones de acceso a datos

Las operaciones de acceso a datos permiten realizar el típico conjunto de operaciones de base de datos **CRUD** *(Create, Read, Update, Delete)* sobre los eventos, comunidades y lugares. Como ya se mencionó estos pueden estar en distintos repositorios (organizacionales, redes sociales, aplicaciones móviles, etc). Las operaciones de acceso a datos también permiten obtener información referente a los usuarios e ítems.

El diseño de esta capa se realiza mediante el patrón DAO *(Data Access Object)*[9] que permite la abstracción del repositorio de datos. Cada implementación de las interfaces DAO propuestas permitirá el acceso a un repositorio de datos específico. La solución incorpora una implementación genérica para el acceso a una base de datos relacional con el esquema *3-Ontology* mediante JDBC (*Java DataBase Connectivity*).

Cabe destacar que las operaciones más utilizadas serán de lectura. Sin embargo, las operaciones crear, actualizar y remover pueden ser usadas para un proceso de migrado o para almacenar la información obtenida del proceso de recomendación como por ejemplo nuevas comunidades.

[9]http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

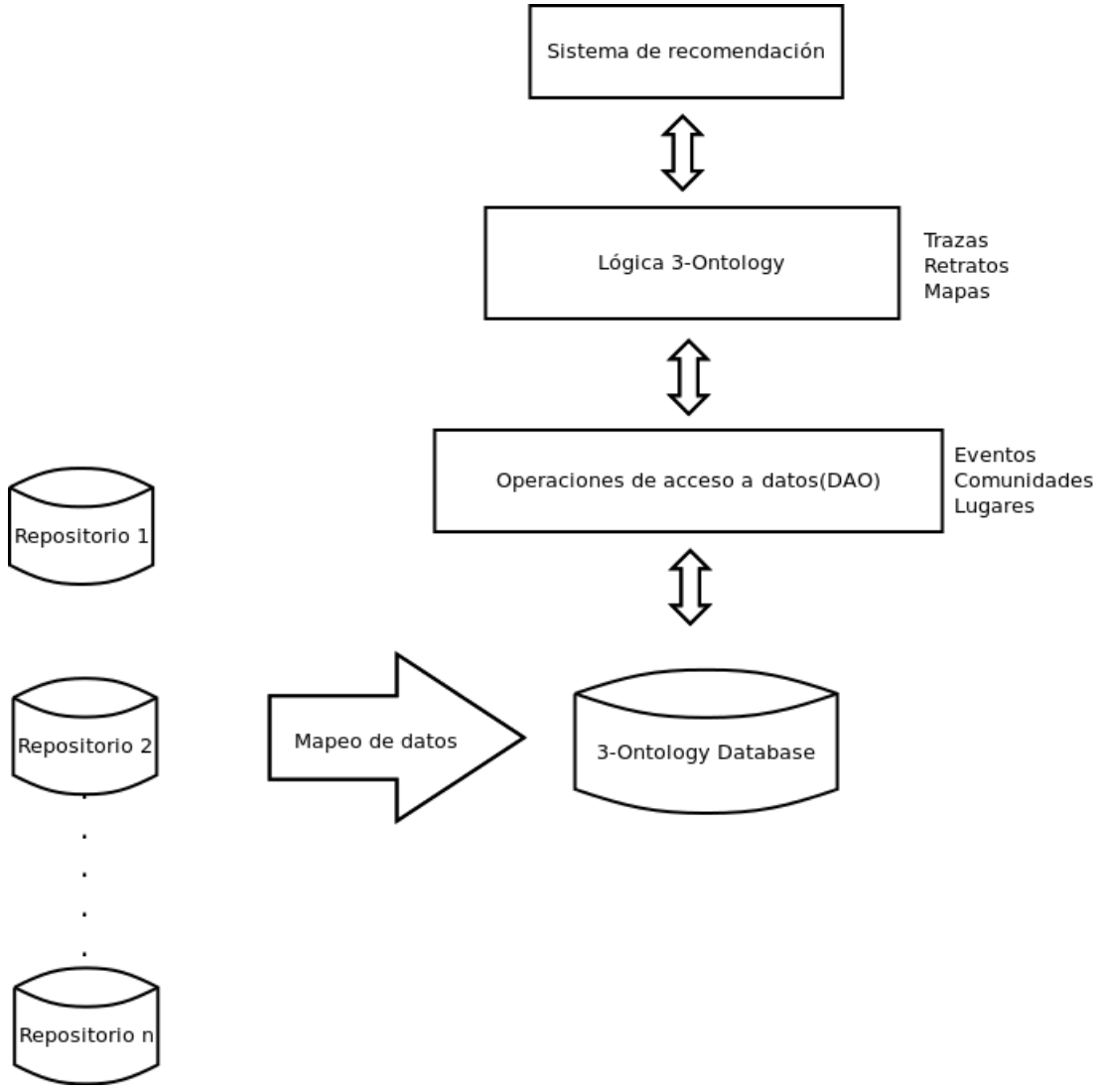

FIGURA 4.5: Repositorio con la estructura de la *3-Ontology*

4.3.3 Lógica de la 3-Ontology

Como se mencionó en el capítulo 3 la *3-ontology* se compone de tres formas de representación (trazas, retratos y mapas) que dotan de lógica a los eventos, comunidades y lugares permitiendo obtener las siguientes ventajas:

- Representar la composición y recursividad de las comunidades y lugares.
- Representar la secuencialidad temporal de los eventos.
- Situar a los eventos dentro de una comunidad y lugar.
- Representar distintos tipos de interacciones hacia los ítems.
- Representar la pertenencia de los ítems y usuarios a un lugar físico y/o virtual.
- Representar la pertenencia de usuarios a una o varias comunidades.
- Representar la pertenencia entre comunidades y lugares.

La representación más un conjunto de métodos permite dar una implementación a la *3-Ontology*. El conjunto de los métodos y la representación son denominados lógica de la *3-Ontology*.

Las trazas denotan la relación existente entre los eventos. Además, de los métodos sobre la causalidad temporal de los eventos se proveen métodos que permiten obtenerlos dadas las características propias de estos para el caso de los SR. Dado lo anterior, se expone un conjunto de métodos dentro de una interfaz que representan a una traza basado en el modelo propuesto. En el código 4.6 se muestra la interfaz propuesta para una traza. Los eventos pueden ser obtenidos de forma ordenada mediante el uso de comparadores. Además, los métodos propuestos permiten realizar apoyo a operaciones de pre y post filtrado en CARS obteniendo sólo los eventos relevantes para el cálculo de la recomendación dado un contexto.

```
1  public interface Trace{
2
3      List<Event> getEvents();
4
5      List<Event> getEvents(Comparator<Event> comparator);
6
7      <E extends Event> List<E> getEvents(Comparator<Event> comparator, Class<E>
           type);
8
9      <E extends Event> List<E> getEvents(Class<E> type);
10
11     List<Event> getEventsForItem(long itemId);
12
13     List<Event> getEventsForItemByTimestamp(long itemId, long timestamp1, long
           timestamp2);
14
15     List<Event> getEventsByTimestamp(long timestamp1, long timestamp2);
16
17     <E extends Event> List<E> getEventsForItem(long itemId, Class<E> type);
18
19     <E extends Event> List<E> getEventsForItem(long itemId,Comparator<Event>
           comparator, Class<E> type);
20
21     List<UserHistory<Event>> getEventsByUser();
22
23     UserHistory<Event> getEventsForUser(long userId);
24
25     UserHistory<Event> getEventsForUserByTimestamp(long userId, long timestamp1,
           long timestamp2);
26
27     <E extends Event> UserHistory<E> getEventsForUser(long userId, Class<E> type);
28
29     LongSet getUsersForItem(long item);
30
31     <E extends Event> LongSet getUsersForItem(long itemId, Class<E> type);
32
33     LongSet getItemsForUser(long userId);
34
35     <E extends Event> LongSet getItemsForUser(long userId, Class<E> type);
36 }
```

CÓDIGO 4.6: Interfaz Trace

```
1  public interface Portrait{
2
3      Community getAllUsers();
4
5      List<Community> getCommunities();
6
7      <E extends Community> List<E> getCommunities(Class<E> type);
8
9      List<Community> getCommunitiesForUser(long userId);
10
11     <E extends Community> List<E> getCommunitiesForUser(long userId, Class<E> type);
12 }
```

CÓDIGO 4.7: Interfaz Portrait

```
1  public interface Map {
2
3      Place getAllItems();
4
5      List<Place> getPlaces();
6
7      <E extends Place> List<E> getPlaces(Class<E> type);
8
9      List<Place> getPlacesForItem(long itemId);
10
11     <E extends Place> List<E> getPlacesForItem(long itemId, Class<E> type);
12 }
```

CÓDIGO 4.8: Interfaz Mapa

Los retratos representan las relaciones existentes entre las comunidades. Se proveen métodos que permiten obtener información referente a las relaciones entre usuarios y comunidades. Por ejemplo, se puede determinar a que comunidades pertenece el usuario u. En el código 4.7 se presenta la definición de la interfaz propuesta para un retrato.

Los mapas representan las relaciones existentes entre los lugares. Se proveen métodos que permiten obtener información referente a a las relaciones entre los ítems y lugares. Por ejemplo, se pueden obtener todos los lugares a los que un ítem c pertenece. En el código 4.8 se presenta la definición de la interfaz propuesta para un mapa.

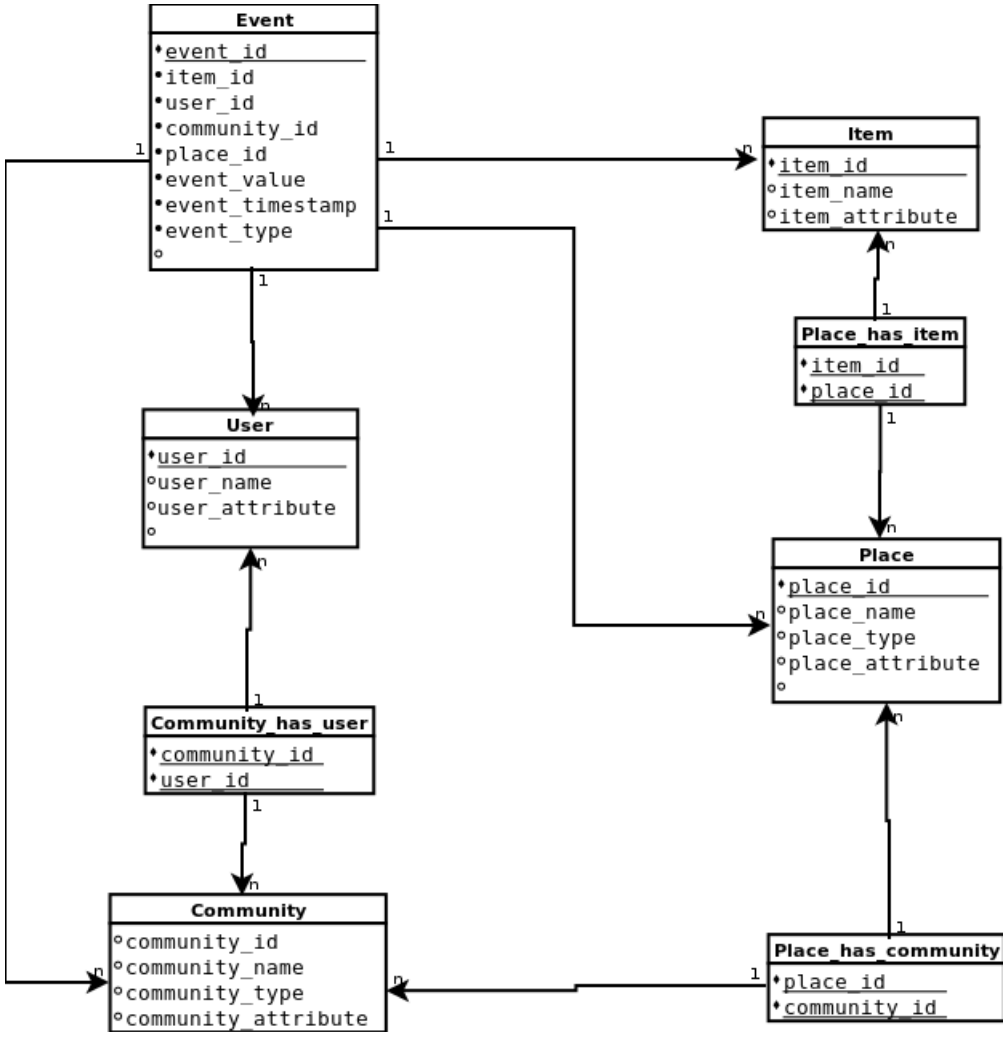

FIGURA 4.6: Modelo entidad-relación

4.4 MODELO ENTIDAD-RELACIÓN PARA LA 3-ONTOLOGY

Para permitir una representación al nivel de repositorio de datos se provee un modelo entidad-relación que permite realizar un almacenamiento en una base de datos relacional. En la Figura 4.6 se muestran las entidades fundamentales (contenedores de sentido) mediante las tablas *Event*, *Community* y *Place*. Los ítems y usuarios son reflejados por las tablas *Item* y *User*.

La columna "*event_type*"de la tabla *event* permite modelar varios tipos de interacciones sobre el mismo ítem. En el modelo físico todas las relaciones n-arias del diagrama entidad-relación son reflejadas mediante las siguientes tablas:

- *Place_has_item*
- *Place_has_community*
- *Community_has_user*

Estas tablas permiten situar los eventos, usuarios e items en un contexto temporal, espacial y social. El modelo presenta pocos atributos obligatorios para no restringir la representación de los repositorios de datos que no poseen todos los datos necesarios del modelo propuesto. De esta manera, se puede instanciar este modelo de datos para un dominio específico y agregar los atributos que se consideren relevantes. Los únicos atributos obligatorios del evento son el identificador del usuario, objeto, valor y el tipo del evento. Por lo tanto, en las otras tablas solo los identificadores son obligatorios dotando de flexibilidad de representación al modelo propuesto.

4.5 CONSTRUCCIÓN DE UN SR

Para construir SR en RBOX 2.0 se define la interfaz *RBOXRecommender* (véase el código 4.9) que permite prestar servicios de recomendación. Los métodos definidos reciben distintos parámetros de entrada para realizar el cálculo de la recomendación para el usuario activo. La salida es una lista de *ScoredItem* que corresponden a ítems con un puntaje calculado por un algún algoritmo de recomendación. Se recomiendan los ítems que tienen el puntaje mayor.

```
1  public interface RboxRecommender {
2
3      String getName();
4
5      List<ScoredItem> recommend(long idUser, int n);
6
7      List<ScoredItem> recommend(long idItem);
8
9      List<ScoredItem> recommend(long idUser, int n, List<Item> candidates,
10         List<Item> exclude);
11
12     List<ScoredItem> recommend(long idUser, List<Item> candidates);
13 }
```

CÓDIGO 4.9: Interfaz RBOXRecommender

El método definido para construir un SR basado en el modelo propuesto se presenta en la Figura 4.7. Primero se deben seleccionar los repositorios de datos que disponen de datos relevantes. Una vez identificados se debe modelar los contenedores de sentido del modelo, en otras palabras se determina que como serán los eventos, comunidades y lugares. Luego se debe determinar si se consolidarán los datos usando la base de datos relacional o implementar la capa **DAO** para cada repositorio. Una vez que se dispone de los datos, se debe seleccionar un algoritmo de recomendación o de aprendizaje de máquina desde algún API como *Lenskit*, *Weka*, *Apache Mahout* o *Mallet*. Si el algoritmo no cumple con las expectativas se puede modificar y/o cambiar por otro. Finalmente, se implementa la interfaz *RBOXRecommender* utilizando las trazas, retratos y mapas como entrada de datos. Es importante señalar que durante el desarrollo del algoritmo de recomendación se procesan las distintas características provistas por el modelo propuesto.

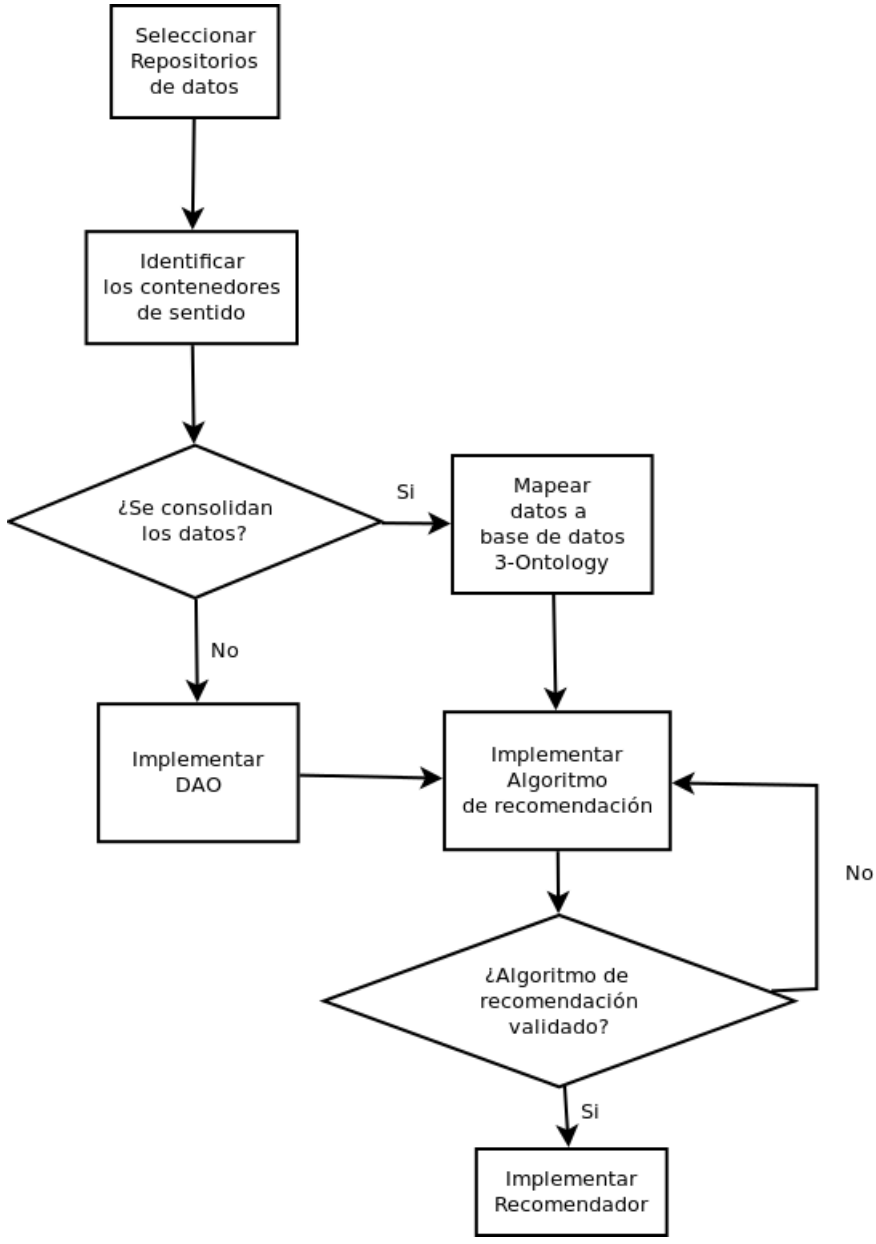

FIGURA 4.7: Flujograma de creación de un SR

4.6 HABILIDADES NECESARIAS DESARROLLAR CON RBOX 2.0

En esta sección se definen el conjunto de habilidades y conocimientos que debe tener un desarrollador para implementar SR con RBOX 2.0.

- **Lenguaje de programación**: el desarrollador debe poseer un nivel intermedio del lenguaje de programación *Java*. Esto quiere decir que debe tener conocimiento en:
  - Fundamentos de programación.
  - Uso de interfaces y clases abstractas.
  - Colecciones.
  - Anotaciones.
  - Tipos genéricos.
  - Utilización de librerías.
- **Diseño orientado a objetos y patrones de diseño**: el desarrollador debe tener conocimientos intermedios de orientación a objetos (abstracción, polimorfismo, encapsulación y herencia) que le permita extender el sistema. Además, debe tener conocer los patrones de diseño usados del sistema.
- **Control del ciclo de vida del software**: este conocimiento se refiere al uso de herramientas que apoyen al proceso de compilación, pruebas y despliegue del software. En el caso de RBOX 2.0 se utiliza *Gradle*, este conocimiento es opcional debido a que se pueden construir SR sin el uso de este tipo de herramientas.
- **Minería de datos y aprendizaje automático**: Se asume que el desarrollador tiene conocimientos de técnicas de minería de datos y aprendizaje automático que le permiten

construir algoritmos de recomendación. Por ejemplo como se específico en la sección 2.1, los SR basados en *rating* se basan en un problema de predicción, luego el desarrollar debe conocer las técnicas del aprendizaje de máquina disponibles para abordar el problema de predicción.

4.7 INTEGRACIÓN CON OTRAS APLICACIONES

La solución provista permite la construcción de SR en *Java* que pueden ser utilizados por aplicaciones Web o móviles que soporten el uso de servicios del tipo *RESTful*. Para lo anterior, se debe encapsular el SR usando *JavaEE 6*[10], permitiendo así disponer del SR en un contenedor.

Los pasos sugeridos para la construcción son los siguientes:

1. Construir un SR con el modelo propuesto en este trabajo.
2. Construir un *SessionBean* (*Enterprise Java Bean*) de tipo *Stateless* que encapsule en métodos de negocio la interfaz de recomendación propuesta por RBOX 2.0.
3. Construir servicios de tipo *RESTful* a partir del *SessionBean* creado.
4. Desplegar los servicios en un contenedor dentro de un servidor de aplicaciones.
5. Consumir los servicios desde aplicaciones clientes.

Esta integración requiere que el desarrollador conozca de arquitectura de aplicaciones empresariales en específico JavaEE, luego implementar la integración sugerida no puede ser aplicada por un desarrollador que no tenga experiencia.

[10]Java Enterprise Edition 6

4.8 COMPARACIÓN CON OTRAS HERRAMIENTAS

En esta sección se realiza una comparación de RBOX 2.0 con otras herramientas de software que permiten la construcción de SR.

RBOX 2.0 para soportar las nuevas características de los SR modernos se rige bajo los siguientes principios de diseño:

- **Mantenibilidad**: se refiere al esfuerzo ahorrado para revisar y corregir errores.
- **Flexibilidad**: se refiere al esfuerzo ahorrado para extender y realizar cambios de configuración del sistema.
- **Reusabilidad**: se refiere al esfuerzo ahorrado de reutilizar módulos y/o componentes de software.
- **Escalabilidad**: se refiere a la capacidad del sistema de soportar una calidad de servicio cuando la carga de eventos aumenta.

Ekstrand et al. (2011) presentan *Lenskit*[11] un *toolkit* para construir, investigar y estudiar SR de filtrado colaborativo basados exclusivamente en eventos de tipo *rating*. Se enfoca principalmente en resolver las dificultades referentes a la investigación con SR. *Lenskit* se basa en tres objetivos de diseño:

- **Modularidad**: los algoritmos de recomendación se descomponen dentro de varias piezas constituyentes como normalización, funciones de similaridad y reglas de predicción. Varios de los componentes anteriores no son parte de un algoritmo particular. Esta diseñado para ser altamente modular y reconfigurable.

[11] http://lenskit.grouplens.org/

- **Claridad**: se debe mantener un núcleo simple pero con interfaces que provean integración y extensión en sistemas en tiempo real.

- **Eficiencia**: se prefiere el código claro antes que obscuras optimizaciones. Puede procesar *Movielens 10m* con 10 millones de eventos de tipo *rating*.

Las ventajas de *Lenskit* son su modularidad, configurabilidad y alto rendimiento para trabajar con SR de filtrado colaborativo. La principal desventaja es no proporcionar una forma estándar de representar los datos que permita modelar características contextuales de las interacciones. Cabe señalar que los algoritmos de filtrado colaborativo utilizados en RBOX 2.0 son extraídos de *Lenskit* ya que son escalables y modulares. La mantenibilidad es asegurada mediante el uso de patrones de diseño y orientación a objetos, la flexibilidad mediante el uso de interfaces simples, la reusabilidad mediante el uso de componentes con alta cohesión y bajo acoplamiento, la escalibilidad mediante el uso de estructuras de datos especializadas como *Sparse Vector*.

Tareen et al. (2010) presentan *Synergy* una herramienta para la creación, ejecución y comparación de diferentes algoritmos de filtrado colaborativo. La idea fundamental de *Synergy* es que los algoritmos de recomendación deben ser reconfigurados o cambiados por otros durante el tiempo. Las ventajas de *Synergy* son la posibilidad de modelar diversas formas de interacción, la hibridación de algoritmos de filtrado colaborativo y la generación de SR en formato *plug-in*. Su principal dificultad es que no se encuentra disponible el código por lo tanto no se dispone de información referente a las cuatro propiedades sistémicas evaluadas.

Cullache (2011) y Cortés (2013) presentan RBOX 1.0 una herramienta de software para experimentar con SR de filtrado colaborativo inspirado por *Synergy* y *Weka*. El uso excesivo del patrón de diseño "*Abstract Factory*" dificulta la mantenibilidad y flexibilidad del código ya que la mayoría de los objetos de la aplicación son creados con este patrón. Respecto a la reusabilidad se logra mediante la agregación de *plug-ins*. La escalabilidad no es resguardada teniendo serios problemas de rendimiento ya que no posee estructuras de datos especializadas

como "*Sparse Vector*".

RBOX 2.0 cumple con la mantenibilidad mediante el uso de patrones de diseño y orientación a objetos. La flexibilidad se asegura mediante el uso de interfaces simples que puedan ser extendidas como es el caso de los eventos *ratings* y *taggings*. La reusabilidad se asegura mediante el diseño de componentes que posean una alta cohesión y bajo acoplamiento. Finalmente, la escalabilidad se asegura mediante el uso de *Lenskit* para algoritmos de SR de filtrado colaborativo y de *Fast-utils*[12] para estructuras de datos complejas. En la tabla 4.1 se muestra un resumen del cumplimiento de las propiedades sistémicas para cada herramienta. Además, los siguientes patrones de diseño, técnicas de programación y librerías apoyan al cumplimiento de los principios de diseño de RBOX 2.0:

- **Tipos genéricos**: provee la capacidad de re-usar el mismo código para distintas entradas.
- **Builders**: patrón creacional que permite la construcción de objetos complejos que no siempre disponen de todos los atributos. Es una alternativa del anti-patrón *telescopic constructor*.
- **Data access object**: patrón estructural que permite la abstracción de los datos.
- **Dependency Injection**: patrón creacional que permite resolver las dependencias de objetos en tiempo de ejecución y/o compilación.
- **Fast-utils**[13]: es un *framework* que extiende de *Java Collections*[14] y provee mapas, conjuntos, listas y colas con baja utilización de memoria, rápido acceso e inserción.
- **Preferencia de objetos inmutables**[15]: su uso asegura que el estado de un objeto no cambiará después que es construido. Son útiles en el desarrollo de aplicaciones concurrentes.

[12]http://fastutil.di.unimi.it/
[13]http://fastutil.di.unimi.it/
[14]http://docs.oracle.com/javase/tutorial/collections/
[15]http://docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html

TABLA 4.1: Comparación con otras herramientas de software

	Herramientas de software			
Propiedad	Lenskit	Synergy	RBOX 1.0	RBOX 2.0
Mantenibilidad	X			X
Flexibilidad	X			X
Reusabilidad	X		X	X
Escalabilidad	X			X

En la tabla 4.1 se puede observar que *Lenskit* al igual que RBOX 2.0 cumplen con las todas propiedades sistémicas evaluadas. Por otro lado, al no tener disponible el código fuente de *Synergy* no se puede determinar si cumple o no con las propiedades evaluadas. En el caso de RBOX 1.0 se intenta utilizar ciertos patrones de diseño para asegurar algunas propiedades, sin embargo el uso inadecuado de patrones de diseño termina limitando la herramienta. Se observa que las propiedades sistemáticas del software RBOX 2.0 y *Lenskit* son idénticas, sin embargo la principal ventaja que posee RBOX 2.0 sobre *Lenskit* está relacionada con la conceptualización proporcionada por el modelo propuesto, que basa su implementación en la *3-Ontology*. Lo anterior logra que RBOX 2.0 sea una herramienta más comprensible que *Lenskit* permitiendo crear rápidamente prototipos para ámbitos empresariales y científicos agregando todas las ventajas presentadas en la sección 3.3.

CAPÍTULO 5. EJEMPLOS DE APLICACIÓN

En el presente capítulo se muestran dos ejemplos de aplicación del *framework* donde se realiza la construcción de SR que permiten validar la eficacia del modelo propuesto.

5.1 SISTEMAS DE RECOMENDACIÓN PARA *MOVIELENS*

Movielens[1] es un servicio libre, no comercial y personalizado ofrecido por *Grouplens*[2] de la Universidad de *Minnesota*[3]. Este permite realizar recomendaciones de películas basándose en interacciones de tipo *rating*. El *dataset* usado de *Movielens* en este trabajo fue presentado en el *workshop HetRec* 2011[4]. Este se basa en dos tipos de interacciones *rating* y *tagging*, el detalle estadístico del *dataset* es el siguiente:

- 2113 usuarios.
- 10197 películas.
- 13222 *tags*.
- 47957 asignaciones de *tags*, por ejemplo tuplas [*user*, *tag*, *movie*].
- Un promedio de 22.696 asignaciones de *tags* por usuario.

[1] http://movielens.umn.edu/
[2] http://grouplens.org/
[3] www.umn.edu/
[4] The 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems

- Un promedio de 8.117 asignaciones de *tags* por ítem.
- 855598 *ratings*.
- Un promedio de 404.921 *ratings* por usuario.
- Un promedio de 84.637 *ratings* por película.

Según la método propuesto para la construcción(véase sección 4.7) el primer paso consiste en realizar la selección del repositorio de datos a usar. En este caso corresponde a un conjunto de archivos de texto que poseen la información sobre los usuarios, películas e interacciones.

Una vez seleccionado el repositorio de datos se debe decidir que estrategia de acceso a los datos se usará, una consolidación dentro una BD *3-Ontology* o un acceso mediante una implementación DAO. En este caso, se decide realizar una implementación de la capa DAO dado que el objetivo fundamental es experimentar para realizar mejoras en el filtrado colaborativo basado en *rating*. Por lo tanto, se hace uso de un FileDAO que implementa las interfaces EventDAO, CommunityDAO y PlaceDAO para un acceso de datos a archivos de texto plano. En el código 5.1 se presenta el constructor del FileDAO. Cabe destacar que en esta etapa se determinan los tres contenedores de sentido provenientes desde los archivos de texto.

Una vez implementado el acceso a los datos con su respectiva representación en los contenedores de sentido se debe implementar un algoritmo de recomendación. Existen diversas API que proveen implementaciones de algoritmos de recomendación, en este caso se hará uso de las implementaciones provistas por *Lenskit* para filtrado colaborativo. Específicamente se usará el enfoque de vecinos más cercanos *user-user*, que es una heurística que dada las preferencias de usuarios cercanos realiza una aproximación de la preferencia del usuario activo (véase la sección 2.2).

A partir del FileDAO se construye una traza, un retrato y un mapa que son la representación lógica de los tres contenedores de sentido de la *3-Ontology*. Luego cada objeto relacionado con el algoritmo de recomendación hace uso de la traza, retrato y mapa, para

```
1   public FileDAO(String pathRatingFile, String pathTaggingFile, char delim) throws
        IOException{
2      ratingFile = new ReaderRatingFile(pathRatingFile, delim);
3      taggingFile = new ReaderTaggingFile(pathTaggingFile, delim);
4      events = Lists.newArrayList(ratingFile);
5      events.addAll(Lists.newArrayList(taggingFile));
6      ratingFile.close();
7      taggingFile.close();
8      userIds = new LongOpenHashSet();
9      itemIds = new LongOpenHashSet();
10     communities = Lists.newArrayList();
11     places = Lists.newArrayList();
12     loadData();
13  }
```

CÓDIGO 5.1: Constructor FileDAO

obtener los eventos, comunidades, lugares, usuarios e ítems relevantes para realizar el cálculo de la recomendación. Finalmente, se implementa la interfaz RboxRecommender para realizar el cálculo de la recomendación para un usuario activo. En el código 5.2 se muestra un ejemplo de implementación de un SR basado en el método propuesto.

El uso del SR se realiza mediante una lista de ScoredItem que corresponden a ítems con un puntaje asignado por los algoritmos de recomendación. Un ejemplo de uso del algoritmo se presenta en el código 5.3 donde se imprimen las 10 primeras recomendaciones para el usuario 75. Este algoritmo está basado en memoria por esta razón se debe re-calcular la lista de recomendaciones cada vez que el sistema es consultado.

Es importante notar que la lógica de la *3-Ontology* provee métodos de pre-filtrado que permiten la implementación de CARS (Adomavicius & Tuzhilin, 2011), obteniendo así eventos relevantes para el cálculo de la recomendación en un contexto dado. Por ejemplo, para realizar una recomendación de películas un día sábado, se pueden obtener solo los eventos sucedidos los sábados. De esta forma se calcula una lista de recomendación para el usuario basada en el contexto temporal donde él asigna una valoración a la película.

```
1       /*Creacion del FileDAO para acceder a los datos*/
2       FileDAO fileDAO = new FileDAO("/home/rodrigo/user_ratedmovies-timestamps.dat",
3            "/home/rodrigo/user_taggedmovies-timestamps.dat", '\t');
4       /*Creacion de las 3 representaciones de la 3-ontology a partir de los
            contenedores de sentido*/
5       Trace trace = new GenericTrace(fileDAO.getEvents());
6       Portrait portrait = new GenericPortrait(fileDAO.getCommunities());
7       Map map = new GenericMap(fileDAO.getPlaces());
8
9       /*Creacion de los objetos relacionados con el algoritmo de recomendacion*/
10      VectorSimilarity similarity = new PearsonCorrelation();
11      UserSimilarity userSimilarity = new UserVectorSimilarity(similarity);
12      VectorNormalizer norm = new MeanVarianceNormalizer();
13      UserVectorNormalizer vectorNormalizer = new DefaultUserVectorNormalizer(norm);
14      NeighborhoodFinder finder = new SimpleNeighborhoodFinder(trace,10,
            userSimilarity, vectorNormalizer);
15      ItemScorer userUserItemScorer = new UserUserItemScorer(trace, finder,
            vectorNormalizer);
16      TopNItemRecommender itemRecommender = new TopNItemRecommender(trace,
            portrait, map, userUserItemScorer);
17      ItemScorer baseline = new ItemMeanRatingItemScorer.Builder(trace, 0).get();
18      PreferenceDomain domain = new PreferenceDomain(0, 5);
19      SimpleRatingPredictor ratingPredictor = new
            SimpleRatingPredictor(userUserItemScorer, baseline, domain);
20
21      /*Construccion de un SR basado en la logica de la 3-Ontology*/
22      RboxRecommender rboxRecommender = new RboxRecommenderBasedItemScorer(trace,
            map, portrait, itemRecommender, ratingPredictor.getBaselineScorer());
```

CÓDIGO 5.2: SR *knn-user/user*

```
1   for(ScoredItem scoredItem: rboxRecommender.recommend(75, 10)){
2           System.out.println(scoredItem.toString());
3       }
```

CÓDIGO 5.3: Ejemplo de uso

5.2 SISTEMAS DE RECOMENDACIÓN PARA UN PROCESO DE GENERACIÓN DE NOTICIAS

González (2012) en su trabajo de titulación modela un proceso de negocio de generación de noticias de la empresa *Observatorio News* que incluye un etiquetado de los periodistas para agregar valor al negocio. Rivera (2012) presenta un modelo 4+1 vistas para describir la arquitectura de software que debe ser soportado para el proceso de negocio. Méndez (2013) realiza la descripción arquitectural para el sistema, especificando cada componente del proceso. RBOX 2.0 según estos trabajos es la herramienta que permite generar SR de tipo *plug-in* que pueden ser integrados en la arquitectura propuesta. Bajo estas premisas se propone como caso de estudio el desarrollo de un SR basado en *tags* para el proceso de negocio de la empresa *Observatorio News*, cabe destacar que el desarrollo de este SR está suscrito al proyecto FONDEF D09I1185, luego los datos usados pertenecen a la empresa socia del proyecto *Observatorio News*.

Observatorio News necesita que los periodistas novatos aprendan de los *Senior* a etiquetar noticias, esto se justifica debido a que dependiendo del nivel de etiquetado que apliquen los periodistas se le puede recomendar noticias de mejor calidad a los usuarios. Para lograr este objetivo la lógica de la *3-Ontology* provee el método *getEventsForUser(userId)* que permite obtener solo los eventos que han efectuado los periodistas Senior. De esta forma el algoritmo puede aprender sobre como etiquetan los periodistas Senior y recomendar a los novatos etiquetas que apoyen su proceso de etiquetado.

El *dataset* de *Observatorio News* corresponde a noticias etiquetadas por periodistas en sectores e incluye la siguiente información:

- id_noticia: corresponde al identificador de la noticia.
- id_idioma: idioma de la noticia, pudiendo ser ingles, español o portugués.
- fec_publicacion: fecha de publicación del artículo. Formato DD-MM-YY.

- id_paises: códigos de países separados por coma.
- paises: nombres de países separados por coma.
- id_sector: código del sector asociado al artículo.
- sector: nombre del sector asociado al artículo.
- id_sub_sectores: códigos de subsectores asociados al artículo.
- sub_sectores: nombres de subsectores asociadas al artículo.
- titulo: título del artículo.
- cuerpo: contenido del cuerpo del artículo (párrafos separados por el carácter '—').
- id_usuario: usuario que creó la noticia y selecciono los *tags*.

Los datos obtenidos se consolidan en el esquema relacional de la *3-Ontology*, para esto se siguió el siguiente procedimiento:

- Se eliminan los campos irrelevantes.
- Se cambia el enfoque desde la noticia hacia el evento.
- Cambiar el formato de la fecha a *UNIX timestamp*.
- Se crean los archivos necesarios para la carga en un motor de base de datos relacional con esquema *3-Ontology*.

Los archivos que resultan del proceso de normalización de datos son *event_bna.dat*, *user_bna.dat* y *item_bna.dat*.

Descripción del archivo event_bna.dat:

1. id_event: identificador del evento.

2. id_noticia: identificador de la noticia.

3. id_usuario: identificador del usuario.

4. id_subsector: identificador del subsector.

5. timestamp_noticia: timestamp del evento en formato UNIX timestamp.

6. type_event: corresponde al tipo de evento (en este caso *tagging*).

Los eventos son uni-valuados, luego si se seleccionan más de un subsector se identifican como eventos distintos.

Descripción del archivo user_bna.dat:

1. id_usuario: identificador del usuario.

2. nombre_usuario: nombre del usuario.

Descripción del archivo item_bna.dat:

1. id_noticia: identificador de la noticia.

2. titulo_noticia: título de la noticia.

En resumen el *dataset* dispone de 438571 eventos, 150 periodistas, 209846 noticias. Con los archivos se puede realizar la carga de datos en una base de datos que disponga del esquema de datos de la *3-Ontology* (véase sección 4.4). En este caso la columna *value* dentro de la tabla *event* corresponderá al subsector colocado por el periodista al momento de escribir la noticia.

En este caso, se hará uso del paquete *sql* de RBOX 2.0 que provee un conjunto de clases que permiten el acceso a una base de datos relacional con el esquema *3-Ontology* mediante JDBC. El motor de bases de datos usado fue *Mysql*[5]. En el código 5.4 se muestra como se realiza el acceso a los datos de *ABC News* mediante los componentes de RBOX 2.0.

[5]http://www.mysql.com/

```
MysqlDataSource dataSource = new MysqlDataSource();
Properties properties = new Properties();
properties.load(new FileInputStream("algorithm.properties"));
dataSource.setServerName(tabla.getString("db.principal.ip"));
dataSource.setDatabaseName(tabla.getString("db.principal.schema"));
dataSource.setUser(tabla.getString("db.principal.user"));
dataSource.setPassword(tabla.getString("db.principal.pass"));
Connection con = dataSource.getConnection();
JDBCDAO dao = new JDBCDAO(con,
                    new BasicSQLStatementFactory(),
                    new BasicSQLStatementFactory(),
                    new BasicSQLStatementFactory(),
                    new BasicSQLStatementFactory(),
                    new BasicSQLStatementFactory(),
                    true);
Trace trace = new GenericTrace(dao.getEvents());
Portrait portrait = new GenericPortrait(dao.getCommunities());
Map map = new GenericMap(dao.getPlaces());
```

CÓDIGO 5.4: Ejemplo de uso mediante JDBCDAO

5.2.1 Algoritmo de Tag Clustering

Hernández (2014) realiza un conjunto de implementaciones de algoritmos de recomendación de *tags* para una red social de generación de noticias basados en el modelo de construcción propuesto en este trabajo de tesis. El algoritmo que se presenta como caso de estudio se basa en un modificación del algoritmo presentado por Begelman (2006). La modificación corresponde al reemplazo del algoritmo de *clustering* por RRW *(Repeated random walks)* (Macropol et al., 2009). Este algoritmo a partir de un *tag* busca los *tags* relacionados y los retorna como recomendaciones. Luego se genera un modelo que puede ser consultado para obtener recomendaciones. Este debe ser actualizado cuando deteriore sus resultados y/o existan nuevos eventos en el sistema.

CAPÍTULO 6. CONCLUSIONES

En este capítulo se muestran las conclusiones del trabajo de tesis realizado. De esta forma, se explicitan los aportes realizados al área de los SR, se describe el cumplimiento de los objetivos específicos de este trabajo dando evidencia de su cumplimiento. Luego se presentan los posibles trabajos futuros a los que da lugar este trabajo de tesis. Finalmente, se exponen las reflexiones finales referentes al trabajo realizado.

6.1 APORTES AL ÁREA DE LOS SISTEMAS DE RECOMENDACIÓN

El desarrollo de los SR como área de investigación ha sido acelerada en la última década debido al creciente desarrollo de la Web. En este trabajo se ha presentado un modelo de datos y un método de construcción de SR que otorga un marco de referencia para implementar SR. Dado esto, el primer aporte de este trabajo al área de los SR se basa en un modelo que permite una representación situada de los eventos que permite aprovechar las características contextuales de estos, este modelo tiene como base conceptual la *3-Ontology* un *framework* proveniente del área de los sistemas colaborativos. Luego se observa que los SR permiten capturar la inteligencia colectiva sobre un dominio de aplicación específico en base a tres contenedores de sentido comunidades, eventos y lugares. Estos además permiten persistir la inteligencia colectiva para ser usada posteriormente para entregar recomendaciones de utilidad a un usuario.

Un aporte de este trabajo de tesis corresponde a una herramienta de software llamada

RBOX 2.0 que permite la construcción de SR basado en el modelo propuesto. RBOX 2.0 es de código abierto y extensible, permitiendo el desarrollo de SR tanto para el ámbito empresarial como científico.

6.2 ACERCA DE LOS OBJETIVOS ESPECÍFICOS

A continuación se describe el nivel de logro alcanzado en el desarrollo de este trabajo de tesis.

1. *Sistematizar el proceso de construcción de SR desde la literatura actual del área.*

   En la sección 4.5 se realizó una sistematización que permite la construcción de SR bajo el modelo propuesto. Este se basa en la evolución de los SR y sus representaciones presentadas en el marco teórico de este trabajo.

2. *Caracterizar las dimensiones emergentes para los SR.*

   En el capítulo 2 se presenta el marco teórico donde se definen las principales dimensiones emergentes de los SR. En el trabajo relacionado de la sección 4.1 se muestra como se han agregado nuevas dimensiones a los SR a las representaciones propuestas.

3. *Diseñar un modelo orientado a eventos de representación de datos para SR.*

   En el capítulo 3 se presenta un modelo orientado a eventos que se basa en el *framework 3-Ontology*. Este modelo permite situar los eventos dentro de un contexto social, espacial y temporal. En otras palabras este modelo es capaz de situar la colaboración existente dentro de aplicaciones pertenecientes a la Web 2.0.

4. *Diseñar un conjunto de operaciones para la obtención de datos desde el modelo de datos propuesto.*

   En el capítulo 3 se presenta la definición formal de un conjunto de operaciones que permiten obtener eventos, comunidades y lugares relevantes dependiendo del algoritmo de recomendación usado. Es importante notar que estas operaciones son parte del nivel meta de la *3-Ontology* correspondientes a las trazas, mapas y retratos que dan sentido al nivel base (eventos, comunidades y lugares).

5. *Diseño e implementación de una herramienta para construir SR.*

   En el capítulo 4 se presenta el diseño e implementación de RBOX 2.0 una herramienta de software que permite el diseño e implementación de SR basados en el modelo de representación propuesto. RBOX 2.0 está construido bajo patrones de diseño que aseguran un conjunto de calidades sistemáticas requeridas.

6. *Construir SR's basados en el framework propuesto.*

   En el capítulo 5 se presentan dos casos de estudio donde se presenta la construcción de dos SR basado en el modelo propuesto. El primer SR se basa en un algoritmo de vecinos más cercanos mediante el uso de eventos de tipo *rating*, en este caso se recomiendan las películas con el *rating* más alto predicho por el algoritmo. El segundo SR se basa en un algoritmo de *clustering* basado en eventos de tipo *tagging*, en este caso se recomiendan los *tags* que se encuentren en el mismo *cluster* que el *tag* buscado. Dada la construcción de estos dos SR se valida la eficacia del modelo propuesto para la representación y construcción de SR basado en distintos tipos de eventos.

7. *Publicar los resultados de la investigación en una revista de la especialidad.*

   Se encuentra en desarrollo 2 publicaciones. La primera se basa en el modelo de representación de datos propuesto. La segunda es el diseño e implementación de RBOX 2.0 como herramienta para la representación y construcción de SR.

6.3 TRABAJO FUTURO

En este trabajo se propone e implementa un modelo de representación y construcción de SR basado en el *framework* conceptual *3-Ontology*. En este trabajo se exploró la eficacia del *framework* para construir SR sin utilizar todas sus dimensiones, dado esto se vislumbran los siguientes trabajos futuros de índole científico e ingenieril:

- Investigar la dimensión de la comunidad y como esta permite retro-alimentar el proceso de recomendación. Por ejemplo,detectar comunidades emergentes dadas las valoraciones de los usuarios y almacenarlas dentro del modelo, para luego ser usadas como información adicional para futuras recomendaciones. Este trabajo se encuentra bajo desarrollo como tesis de Magíster en Ingeniería Informática (Ochoa, 2013).

- Investigar la dimensión referente a los lugares mediante la propuesta de un *framework* que explicite los tipos de información espacial que son usados en SR. Construir un conjunto de algoritmos basados en lugares y explicitar las mejoras obtenidas en los SR.

- Realizar una expansión del *framework* propuesto para la evaluación de SR de forma *offline* y *online*, esto otorgaría una forma unificada para comparar diversos SR.

- Investigar la construcción de SR híbridos utilizando las tres dimensiones del *framework* para validar la eficacia y eficiencia de construir SR de recomendación con el modelo propuesto.

- Estudiar la capacidad de la *3-Ontology* para representar la Web 2.0. Esto se justifica en la capacidad de representación de entornos colaborativos que posee la *3-Ontology*. En este trabajo solo estudio su utilización para SR, sin embargo se podría extender en un meta-modelo para todo tipo de aplicación de la Web 2.0.

- La construcción de un IDE basado en el modelo propuesto utilizando la implementación de RBOX 2.0. Esto facilitaría la construcción dado el uso de una interfaz gráfica que guié el proceso de construcción.

6.4 REFLEXIONES FINALES

El desarrollo de este trabajo de tesis ha significado un gran desafio personal. Desde un comienzo he participado del grupo *Social Tagging* perteneciente al proyecto FONDEF D09I1185 Observatorios de la Web en tiempo real, en este equipo incentivados por el Dr. Edmundo Leiva adoptamos un trabajo colaborativo que nos permitió compartir nuestro trabajo entregando y recibiendo ayuda de los miembros del equipo. Esto ha sido una experiencia enriquecedora donde se vive problemas similares a los que tendríamos en nuestra vida profesional.

En este trabajo he tenido que utilizar las capacidades ingenieriles y científicas adquiridas durante mis años de formación en el departamento de informática. Se valora de manera muy especial la entrega de habilidades blandas que permiten un mejor desempeño en el ámbito social y laboral.

Sobre el tema abordado en este trabajo ha sido muy interesante debido a su impacto en la sociedad actual, donde la Web es parte de nuestro diario vivir y los SR nos permiten tener a nuestra disposición información relevante. Dado lo anterior para ámbitos empresariales es necesario contar con representaciones y estándares que permitan construir SR.

REFERENCIAS

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, *23*(1), 103–145.
URL http://doi.acm.org/10.1145/1055709.1055714

Adomavicius, G., & Tuzhilin, A. (2001). Multidimensional recommender systems: A data warehousing approach. En L. Fiege, G. Mühl, & U. Wilhelm (Editores) *Electronic Commerce*, vol. 2232 de *Lecture Notes in Computer Science*, (pág. 180–192). Springer Berlin Heidelberg.
URL http://dx.doi.org/10.1007/3-540-45598-1_17

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, *17*(6), 734–749.
URL http://dx.doi.org/10.1109/TKDE.2005.99

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. En F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Editores) *Recommender Systems Handbook*, (pág. 217–253). Springer US.
URL http://dx.doi.org/10.1007/978-0-387-85820-3_7

Begelman, G. (2006). Automated tag clustering: Improving search and exploration in the tag space. En *In Proc. of the Collaborative Web Tagging Workshop at WWW'06*.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*(0), 109 – 132.
URL http://www.sciencedirect.com/science/article/pii/S0950705113001044

Brambilla, M., Fraternali, P., & Vaca, C. (2011). A notation for supporting social business process modeling. En R. Dijkman, J. Hofstetter, & J. Koehler (Editores) *Business Process Model and Notation*, vol. 95 de *Lecture Notes in Business Information Processing*, (pág. 88–102). Springer Berlin Heidelberg.
URL http://dx.doi.org/10.1007/978-3-642-25160-3_7

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–370.
URL http://dx.doi.org/10.1023/A:1021240730564

Cortés, S. (2013). *Un ambiente configurable para definir y experimentar con algoritmos de recomendación y métricas de evaluación.*. Trabajo de titulación de ingeniería civil en informática, Departamento de Ingeniería Informática.

Cullache, A. (2011). *Software como laboratorio para probar y evaluar algoritmos de recomendación*. Trabajo de titulación de ingeniería en ejecución informática, Departamento de Ingeniería Informática.

Ekstrand, M. D., Ludwig, M., Konstan, J. A., & Riedl, J. (2011). Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. En B. Mobasher, R. D. Burke, D. Jannach, & G. Adomavicius (Editores) *RecSys*, (pág. 133–140). ACM.
URL http://dblp.uni-trier.de/db/conf/recsys/recsys2011.html#EkstrandLKR11

Gao, M., Liu, K., & Wu, Z. (2010). Personalisation in web computing and informatics: Theories, techniques, applications, and future research. *Information Systems Frontiers*, *12*(5), 607–629.
URL http://dx.doi.org/10.1007/s10796-009-9199-3

González, A. (2012). *Comparando la funcionalidad y la usabilidad de dos aplicaciones basado*

en un proceso de negocios de generación de noticias. Trabajo de titulación de ingeniería civil informática, Departamento de Ingeniería Informática.

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. En *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, (pág. 230–237). New York, NY, USA: ACM.
URL `http://doi.acm.org/10.1145/312624.312682`

Hernández, F. (2014). *Selección e implementación de algoritmos de recomendación basados en tags en una red social generadora de noticias*. Trabajo de titulación de ingeniería civil en informática, Departamento de Ingeniería Informática.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., Riedl, J., & Volume, H. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, *40*, 77–87.

Leiva-Lobos, E., & Covarrubias, E. (2002). The 3-ontology: A framework to place cooperative awareness. En J. Haake, & J. Pino (Editores) *Groupware: Design, Implementation, and Use*, vol. 2440 de *Lecture Notes in Computer Science*, (pág. 189–199). Springer Berlin Heidelberg.
URL `http://dx.doi.org/10.1007/3-540-46124-8_13`

Macropol, K., Can, T., & Singh, A. (2009). Rrw: repeated random walks on genome-scale protein networks for local cluster discovery. *BMC Bioinformatics*, *10*(1), 283.
URL `http://www.biomedcentral.com/1471-2105/10/283`

Molins, F. (2012). *Generación de datasets con esquema estándar para entrenar algoritmos de recomendación para redes sociales*. Trabajo de titulación de ingeniería civil en informática, Departamento de Ingeniería Informática.

Murugesan, S. (2007). Understanding web 2.0. *IT Professional*, *9*(4), 34–41.
URL `http://dx.doi.org/10.1109/MITP.2007.78`

Méndez, A. (2013). *Descripción arquitectural de un sistema generador de servicios de recomendación para observatorios de la Web*. Trabajo de titulación de ingeniería civil en informática, Departamento de Ingeniería Informática.

Ochoa, D. (2013). *Definición de un framework para generar sistemas de recomendación sobre comunidades, considerando la inteligencia colectiva*. Propuesta de tema de tesis para el grado de magíster en ingeniería informática, Departamento de Ingeniería Informática.

Padula, M., Reggiori, A., & Capetti, G. (2009). Managing collective knowledge in the web 3.0. En *Proceedings of the 2009 First International Conference on Evolving Internet*, INTERNET '09, (pág. 33–38). Washington, DC, USA: IEEE Computer Society.
URL `http://dx.doi.org/10.1109/INTERNET.2009.12`

Palomino, M. (2012). *Un framework para presentar y experimentar con sistemas de recomendación en la Web 2.0*. Thesis, Departamento de Ingeniería Informática.

Panagiotis, S., Alexis, P., Yannis, M., Pinar, S., & Ismail, T. (2011). Geo-social recommendations based on incremental tensor reduction and local path traversal. 2063228 89-96.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. En J. B. Smith, F. D. Smith, & T. W. Malone (Editores) *CSCW*, (pág. 175–186). ACM.
URL `http://dblp.uni-trier.de/db/conf/cscw/cscw1994.html#ResnickISBR94`

Rivera, C. (2012). *Diseño de una arquitectura para prestar servicios de recomendación dentro de una red social de generación de noticias.*. Trabajo de titulación de ingeniería civil en informática, Departamento de Ingeniería Informática.

Sampieri, R. H., Collado, C. F., & Lucio, P. B. (2005). *Fundamentos de metodología de la investigación: bachillerato*. McHraw-Hill.
URL http://books.google.cl/books?id=zk4GAQAACAAJ

Song, Y., Zhang, L., & Giles, C. L. (2011). Automatic tag recommendation algorithms for social recommender systems. *ACM Trans. Web*, *5*(1), 4:1–4:31.
URL http://doi.acm.org/10.1145/1921591.1921595

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, *2009*, 4:2–4:2.
URL http://dx.doi.org/10.1155/2009/421425

Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human-machine Communication*. Learning in doing : social, cognitive, and computational perspectives. Cambridge University Press.
URL http://books.google.cl/books?id=AJ_eBJtHxmsC

Tareen, B., Lee, J.-w., & Lee, S.-g. (2010). Synergy: A workbench for collaborative filtering algorithms on user interaction data. En *International Workshop on User Data Interoperability in the Social Web*. ACM.

Victor, P., Cock, M., & Cornelis, C. (2011). *Trust and Recommendations*, book section 20, (pág. 645–675). Springer US.
URL http://dx.doi.org/10.1007/978-0-387-85820-3_20

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

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.

URL http://dx.doi.org/10.1007/s11257-008-9048-y