Optimización de sistemas de recomendaciones en un entorno médico mediante la utilización de contexto social obtenido de las redes sociales

Proyecto de Grado

Daniel González Bernal
danielgonzalezbernal@gmail.com

Supervisoras

Libertad Tansini
libertad@fing.edu.uy

Regina Motz
rmotz@fing.edu.uy

Montevideo, Uruguay
2013

Resumen

En el presente trabajo se propone un algoritmo de recomendación basado en friendsourcing para ser utilizado en sistemas de recomendaciones de recursos médicos. El objetivo de friendsourcing es utilizar información social para mejorar la experiencia de búsqueda de los usuarios y en este caso el algoritmo propuesto considera características sociales como son las relaciones entre los usuarios en una red social y las valuaciones de recursos previamente hechas por éstos. El algoritmo de recomendación se integra a un sistema de recomendación real y se llevan a cabo pruebas para testear el correcto y esperado comportamiento del mismo. Finalmente se proponen mejoras al algoritmo como trabajo a futuro. En las pruebas realizadas se lograron obtener resultados eficientes (precision = 75 %, recall = 78 %), mostrando que la información contenida en las redes sociales como son las relaciones de amistad entre usuarios y las valuaciones de recursos hechas por usuarios similares son relevantes para la generación de recomendaciones personalizadas en un sistema de recomendación.

Palabras clave: Sistemas de recomendaciones, algoritmo de recomendación, redes sociales, friendsourcing.

Agradecimientos

Agradezco a todas las personas que colaboraron en la realización de este proyecto, especialmente a Libertad Tansini y Regina Motz, supervisores del proyecto; a Alejandro Fernández, Alicia Díaz, Leandro Mendoza y Roberto Guisandez de la Universidad Nacional de La Plata e integrantes del proyecto LACCIR QHIR; y a mi familia por el apoyo incondicional.

Índice general

Capítulo 1

Introducción

El objetivo principal de los sistemas de recomendaciones es ayudar y guiar a los usuarios en la búsqueda y recuperación de información considerando sus intereses y necesidades[1]. La calidad de la información obtenida por un sistema de recomendación es una cualidad deseable, en especial en el ámbito de la salud. Debido a la sensibilidad de la información en entornos médicos, se espera que los resultados sean precisos y relevantes de acuerdo a la intención de búsqueda de los usuarios[1]. La gran cantidad de información disponible en la web así como el uso masivo de las redes sociales imponen importantes desafíos para los sistemas de información, como son el manejo de la información y la habilidad para brindar recomendaciones de manera eficiente[1]. Para lidiar con estos problemas, se pueden utilizar diversas técnicas. En este trabajo se profundizará en una técnica llamada *friendsourcing*, que recolecta y hace uso de información adecuada, disponible sólo para un confiable y eventualmente reducido grupo de personas que están socialmente conectadas en una red[2], y en contraposición con crowdsourcing que recolecta y hace uso de información brindada por una comunidad abierta de personas.

Las recomendaciones forman parte de la vida cotidiana de las personas, por ejemplo cuando se quiere elegir una película o un libro[3]. Usualmente para tomar estas decisiones se confía en algún conocimiento externo[3]. Este conocimiento puede ser obtenido de procesos sociales como son las interacciones de personas en una red social en la web[3]. Este proyecto parte de la hipótesis que la información contenida en el contexto social es clave para poder realizar recomendaciones relevantes. A partir de dicha hipótesis, se hace énfasis en cómo utilizar el contexto social obtenido de las redes sociales para optimizar las recomendaciones en un entorno médico.

El algoritmo de recomendación obtiene información de los amigos de los usuarios de las redes sociales así como información de las valuaciones de recursos hechas por el usuario y sus amigos. Se describe como puede ser extraída dicha información de las redes sociales y la manera en que el algoritmo de recomendación procesa dicha información para obtener los resultados deseados. En las pruebas realizadas para testear el algoritmo se lograron obtener resultados eficientes (precision = 75 %, recall = 78 %), mostrando que la información contenida en las redes sociales es fundamental para los sistemas de recomendaciones.

1.1 Contexto y motivación

Este trabajo está enmarcado en el proyecto de investigación *LACCIR Quality Health Information Retrieval* (QHIR): *Improving Semantic Recommender Systems with Friendsourcing*[4], en el cual se desarrolla una red social en un entorno médico (Red UniSalud[5]) y un sistema de calificación y recomendación de recursos utilizando el contexto social obtenido de la red social. Uno de los objetivos de este trabajo es proveer un algoritmo de recomendación basado en friendsourcing que brinde

recomendaciones personalizadas a los usuarios de la red social y que éstas sean confiables. Se espera eficiencia y rapidez en los resultados brindados por el algoritmo. Finalmente se integrará al sistema de recomendaciones desarrollado en el proyecto LACCIR QHIR.

1.2 Objetivos y resultados esperados

Los objetivos principales de este proyecto son investigar y proponer mecanismos que utilicen el contexto social obtenido de las redes sociales en un entorno médico con el objetivo de personalizar y optimizar las recomendaciones.

Al término del proyecto se espera haber cumplido con los siguientes puntos:

- Estado del arte en sistemas de recomendaciones, técnicas y algoritmos de recomendación, sistemas híbridos de recomendación, redes sociales, crowdsourcing y friendsourcing.

- Especificación de un algoritmo de recomendación basado en friendsourcing que utilice el contexto social obtenido de las redes sociales en un entorno médico para realizar recomendaciones personalizadas.

- Implementación del algoritmo que cumpla con los requerimientos funcionales y no funcionales establecidos en la etapa de análisis.

- Integración del algoritmo en un sistema de recomendación real.

- Generación de casos de prueba para realizar el testeo del algoritmo.

- Evaluación preliminar de los resultados obtenidos con el objetivo de analizar el funcionamiento del algoritmo propuesto.

- Cálculo del orden de complejidad del algoritmo de recomendación y análisis de los tiempos de ejecución.

- Propuestas de mejora al algoritmo de recomendación como trabajo a futuro.

1.3 Organización del documento

A continuación se describe la organización del resto de este documento.

En el capítulo 2 se describe el estado del arte en sistemas de recomendaciones, técnicas y algoritmos de recomendación, sistemas híbridos de recomendación, redes sociales, crowdsourcing y friendsourcing.

El capítulo 3 contiene la especificación del problema a resolver, los requerimientos funcionales y no funcionales que el algoritmo debe satisfacer, y el diseño del algoritmo de recomendación propuesto.

En el capítulo 4 se describe la implementación del algoritmo de recomendación y su inclusión en un sistema de recomendación real en un entorno médico con el objetivo de personalizar y optimizar las recomendaciones.

El capítulo 5 describe los casos de prueba generados para poder testear el algoritmo, se analizan e interpretan los resultados obtenidos, se calcula el orden de complejidad del algoritmo de recomendación y se analizan los tiempos de ejecución.

En el capítulo 6 se presentan las conclusiones del proyecto y en base a las pruebas y a los resultados se sugieren posibles cambios y mejoras al algoritmo como trabajo a futuro.

Capítulo 2

Trabajos relacionados y antecedentes

En este capítulo se realiza el estado del arte en sistemas de recomendaciones, técnicas y algoritmos de recomendación, sistemas híbridos de recomendación, redes sociales, crowdsourcing y friendsourcing, lo cual sirve de base para proponer mecanismos que utilicen el contexto social obtenido de las redes sociales para personalizar las recomendaciones. En la sección 2.1 se describen los sistemas de recomendaciones en forma general. En la sección 2.2 se presenta una clasificación de las técnicas de recomendación y en la sección 2.3 se describen con más profundidad los algoritmos correspondientes a las técnicas.

2.1 Sistemas de recomendaciones

Los sistemas de recomendaciones son aplicaciones de software focalizadas en ayudar a los usuarios en la toma de decisiones mientras estos interactúan con grandes volúmenes de información[6]. Estos sistemas brindan a los usuarios recomendaciones de interés teniendo en cuenta las preferencias que han manifestado explícitamente o implícitamente[6]. Por ejemplo, en *Wikipedia.org* se brinda a los usuarios la oportunidad de calificar atributos de calidad de los artículos. Tanto el crecimiento constante en el volumen de información disponible en la web así como la complejidad de la misma hacen de los sistemas de recomendaciones herramientas fundamentales para los usuarios en la búsqueda de información[6].

Los sistemas de recomendaciones surgieron en la década de los noventa, cuando en los grupos de noticias se brindaban servicios de filtrado de noticias permitiendo a los suscriptores acceder a aquellas noticias que podrían llegar a ser de su interés en función de sus preferencias[7]. Los sistemas de recomendaciones utilizan técnicas de descubrimiento de conocimiento para poder generar recomendaciones personalizadas durante una interacción con el usuario[8]. Por ejemplo, en el comercio electrónico (e-commerce), los sitios web dedicados a esta actividad utilizan sistemas de recomendaciones para ayudar a sus clientes a comprar productos que sean de su interés[9]. Los sistemas de recomendaciones aprenden los intereses de los clientes y recomiendan productos que son más valiosos para ellos a partir de los productos disponibles en el sitio web[9]. De esta manera estos sistemas ayudan a dichos sitios a incrementar sus ventas[9]. Algunos ejemplos de estos sitios web son Mercado Libre y Amazon. En el caso de Amazon el 35% de las ventas resultan de recomendaciones brindadas en el propio sitio web[10].

Los sistemas de recomendaciones utilizan: *datos base*, que es la información que el sistema posee antes de que comience el proceso de recomendación; *datos de entrada*, que es la información que los usuarios le brindan al sistema; y *algoritmos de recomendación*, que procesan los datos base y los datos de entrada para generar las recomendaciones[11]. La clásica interacción entre el usuario y el sistema de recomendación es la siguiente: el sistema solicita al usuario que ingrese algún tipo de

información (datos de entrada), ya sea en forma de calificaciones de ítems o simplemente listando sus ítems favoritos, luego el sistema de recomendación procesa dicha información utilizando los datos base y los algoritmos de recomendación, y finalmente el sistema brinda al usuario recomendaciones personalizadas (datos de salida)[12]. Con respecto a la transparencia de los sistemas de recomendaciones, esto se refiere a la comprensión del usuario de porqué una recomendación en particular fue hecha, se observa que éstos actúan en general como "cajas negras", lo que significa que no ofrecen al usuario una visión lógica del sistema ni tampoco brindan justificación alguna acerca de los criterios utilizados para realizar las recomendaciones[12]. En la figura 1 se muestra la visión que tiene un usuario de un sistema de recomendaciones.

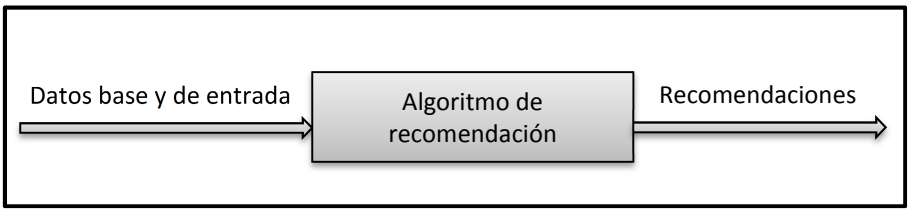

Figura 1: Visión del usuario de un sistema de recomendaciones

2.2 Técnicas de recomendación

Existen diversas clasificaciones de las técnicas de recomendación que son utilizadas por los sistemas de recomendaciones[11]. En esta sección se presenta una posible clasificación. En la sección 2.3 se profundiza en los algoritmos de recomendación correspondientes.

2.2.1 Filtrado colaborativo

En los sistemas de filtrado colaborativo se reconocen similitudes entre los usuarios basados en las puntuaciones de ítems y se generan recomendaciones en base a comparaciones entre usuarios similares[11]. Esta técnica es una de las más familiares y una de las más maduras[11]. A su vez estos sistemas se pueden dividir en dos categorías[11]: *basados en memoria* (memory-based) y *basados en modelo* (model-based). En los sistemas basados en memoria se comparan los usuarios directamente entre sí utilizando correlación u otras medidas[11]. En los sistemas basados en modelo se construye un modelo a partir de los datos históricos de puntuaciones y éste es utilizado para la generación de recomendaciones[11]. En los sistemas de filtrado colaborativo los datos base son las puntuaciones de ítems realizadas por los usuarios, los datos de entrada son las puntuaciones de ítems realizadas por el usuario al que se le quiere brindar las recomendaciones, y el algoritmo de recomendación consiste en identificar usuarios similares y extrapolar las preferencias de los ítems no vistos o no puntuados, a partir de las puntuaciones hechas por dichos usuarios similares[11]. Por ejemplo, los usuarios son representados por vectores de puntuaciones de ítems y por lo tanto se busca similitud entre éstos basados en el coseno o en la correlación de Pearson. Esto se profundiza en la sección 2.3.1. La gran fortaleza de esta técnica es que es completamente independiente de la representación de los ítems que son recomendados. Otra ventaja es que esta técnica funciona bien para ítems complejos

como la música y las películas[11]. Esta técnica presenta la desventaja de que si el usuario al que se le quiere brindar las recomendaciones ha hecho pocas puntuaciones de ítems, se dificulta la comparación con los otros usuarios basándose solamente en las puntuaciones[11]. Otra desventaja es que no es fácil recomendar un ítem que se haya puntuado pocas veces[11]. Este problema que enfrentan los sistemas de recomendaciones se denomina *cold-start problem* (problema del arranque en frío)[13].

2.2.2 Basadas en contenido

En los sistemas de recomendaciones basados en contenido, los ítems de interés son definidos por sus características, como por ejemplo las palabras en un documento de texto[11]. Estos sistemas aprenden el perfil de los intereses de un usuario basándose en las características de los ítems que el usuario ha puntuado[11], a esto se le llama *correlación ítem a ítem* (ítem-to-item correlation)[11]. Para determinar los perfiles se utiliza algún método de aprendizaje, como por ejemplo árboles de decisión o redes neuronales[11]. Los perfiles son modelos a largo plazo y se actualizan a medida que se observan más preferencias del usuario[11]. En los sistemas basados en contenido los datos base son las características de los ítems, los datos de entrada son las puntuaciones de ítems hechas por el usuario al que se le quiere brindar las recomendaciones, y el algoritmo de recomendación consiste en generar el perfil del usuario que se ajuste al comportamiento en las puntuaciones de ítems y luego se utiliza dicho perfil para determinar las preferencias[11]. Estos sistemas tienen el problema de que necesitan muchas puntuaciones de ítems para poder generar perfiles confiables[11]. Otra desventaja es que están limitados por las características de los ítems que recomiendan[11].

2.2.3 Demográficas

En los sistemas de recomendaciones demográficos se categorizan los usuarios a partir de sus atributos personales y se realizan recomendaciones de ítems basándose en clases demográficas[11]. En estos sistemas los datos base son las puntuaciones de ítems hechas por los usuarios e información demográfica de los mismos, los datos de entrada son los atributos demográficos del usuario al que se le quiere brindar las recomendaciones, y el algoritmo de recomendación consiste en identificar aquellos usuarios que son demográficamente similares al usuario activo y se extrapolan las preferencias a partir de las puntuaciones de ítems[11]. A modo de ejemplo, si una mujer busca comprar productos de belleza, el sistema de recomendación tendrá en cuenta su información demográfica (sexo, edad, país de residencia, etc.) para brindar recomendaciones de productos que sean acordes a dicha mujer y evitará recomendar productos que potencialmente no sean de su interés como por ejemplo perfumes para hombre. Una ventaja de esta técnica es que no se requiere de un historial de puntuaciones del usuario como en las técnicas de filtrado colaborativo y en las basadas en contenido[11]. Un problema que presenta esta técnica es que debe reunir información demográfica del usuario[11].

2.2.4 Basadas en utilidad

En los sistemas de recomendaciones basados en utilidad se realizan recomendaciones de ítems basadas en la computación de la utilidad de los ítems para el usuario[11]. El problema es cómo crear la función de utilidad para cada usuario[11]. En estos sistemas los datos base son las características de los ítems, la entrada es una función de utilidad sobre los ítems que describe las preferencias del usuario al que se le quiere brindar las recomendaciones, y el algoritmo de recomendación consiste en aplicar la función de utilidad a los ítems y recomendar aquellos con mayor utilidad para el usuario activo[11]. La función de utilidad puede ser definida por el administrador del sistema, ya que debe poseer conocimientos acerca de los intereses de los usuarios. Una ventaja de esta técnica es que no necesita de puntuaciones de ítems para generar las recomendaciones[11]. Una desventaja que presentan estos sistemas es que no son flexibles en dominios complejos y subjetivos[11]. Otra desventaja es que la habilidad para recomendar es estática, ya que no tienen la capacidad de aprender[11].

2.2.5 Basadas en conocimiento

En los sistemas de recomendaciones basados en conocimiento se realizan recomendaciones de ítems basadas en inferencias acerca de las necesidades y preferencias del usuario[11]. En estos sistemas hay tres tipos de conocimiento[11]: *conocimiento del catálogo*, que es el conocimiento de los ítems que son recomendados y sus características; *conocimiento funcional* que es el conocimiento para mapear las necesidades del usuario y el ítem que podría satisfacer estas necesidades; y *conocimiento del usuario*, que es el conocimiento que el sistema debe tener sobre el usuario para poder brindar buenas recomendaciones. En estos sistemas los datos base son las características de los ítems y el conocimiento de cómo estos ítems satisfacen las necesidades de los usuarios, la entrada es la descripción de las necesidades o intereses del usuario al que se le quiere brindar las recomendaciones, y el algoritmo de recomendación consiste en inferir aquellos ítems que satisfagan las necesidades o sean de interés para el usuario activo[11]. Una ventaja de estos sistemas es que son apropiados para exploración casual[11]. Una desventaja que presentan estos sistemas es que necesitan adquirir el conocimiento[11]. En la siguiente sección se presentan algoritmos de recomendación que permiten adquirir dicho conocimiento. En la tabla 1 se muestra un resumen de las características de las distintas técnicas de recomendación[11].

Técnica de recomendación	Datos base	Datos de entrada	Proceso de recomendación
Filtrado colaborativo	Puntuaciones de ítems realizadas por los usuarios	Puntuaciones de ítems realizadas por el usuario activo	Identificar usuarios similares al usuario activo y extrapolar las preferencias de los ítems a partir de las puntuaciones realizadas por dichos usuarios
Basada en contenido	Características de los ítems	Puntuaciones de ítems realizadas por el usuario activo	Generar el perfil del usuario que representa el comportamiento de las puntuaciones realizadas por el

			usuario activo y utilizar dicho perfil para determinar las preferencias
Demográfica	Información demográfica de los usuarios y sus puntuaciones de ítems	Información demográfica del usuario activo	Identificar usuarios demográficamente similares al usuario activo y extrapolar las preferencias a partir de las puntuaciones de ítems
Basada en utilidad	Características de los ítems	Función de utilidad sobre los ítems que describe las preferencias del usuario activo	Aplicar la función de utilidad proporcionada y determinar las preferencias
Basada en conocimiento	Características de los ítems y conocimiento de cómo estos ítems satisfacen las necesidades de los usuarios	Descripción de las necesidades o intereses del usuario activo	Inferir los ítems que satisfagan las necesidades o sean de interés para el usuario activo

Tabla 1: Características de las técnicas de recomendación

2.3 Algoritmos de filtrado colaborativo

En esta sección se presentan, a modo de ejemplo, algoritmos de filtrado colaborativo que utilizan algunas de las técnicas de recomendación presentadas en la sección anterior. El objetivo de los algoritmos de filtrado colaborativo es sugerir ítems nuevos o predecir la utilidad de un determinado ítem para un usuario en particular basado en las preferencias del usuario y las opiniones de usuarios similares[8]. En un clásico escenario de filtrado colaborativo existe: una lista de usuarios, una lista de ítems y cada usuario tiene una lista de opiniones sobre los ítems[8]. Estas opiniones pueden ser dadas explícitamente mediante un puntaje, generalmente en una escala numérica, o pueden ser obtenidas implícitamente[8]. Al usuario al que se le quiere brindar las recomendaciones se le llama *usuario activo*[8].

En los algoritmos de filtrado colaborativo se definen dos conceptos: la *predicción*, que es un valor numérico que representa el interés predicho por el algoritmo de un determinado ítem para el usuario activo; la *recomendación*, que es una lista que contiene los ítems que serían de mayor interés para el usuario activo[8]. Los algoritmos representan las puntuaciones como una matriz, en la cual cada celda (i, j) contiene el puntaje, en escala numérica, del usuario i-ésimo sobre el ítem j-ésimo[8]. En caso de que el usuario aún no haya puntuado un determinado ítem el puntaje es cero[8]. Los algoritmos de filtrado colaborativo se pueden dividir en dos grandes categorías: *basados en memoria* o también llamados *basados en usuarios* (user-based), y *basados en modelo* o también denominados *basados en ítems* (item-based)[8].

2.3.1 Basados en memoria

Los algoritmos de filtrado colaborativo basados en memoria emplean técnicas estadísticas para encontrar un conjunto de usuarios, llamados *vecinos*, que históricamente han estado de acuerdo con el usuario activo, puntuando de manera similar diferentes ítems[8]. Una vez que la vecindad está formada, se utilizan diferentes algoritmos para combinar las preferencias de los vecinos y poder generar las recomendaciones de ítems para el usuario activo[8].

2.3.2 Basados en modelo

Los algoritmos de filtrado colaborativo basados en modelo construyen un modelo de puntuaciones para brindar recomendaciones de ítems[8]. En este enfoque primero se analizan los ítems que ha puntuado el usuario activo y se computa cuan similares son estos ítems con el ítem en cuestión[8]. Luego se seleccionan los ítems más similares y finalmente se calcula la predicción tomando un promedio ponderado de las puntuaciones del usuario activo sobre estos ítems[8]. Existen distintos métodos para computar la *similitud* y la *predicción*.

A continuación se describen algunos métodos para computar la *similitud* entre dos ítems[8]. Para ello se utiliza la siguiente nomenclatura[8]:

- U es el conjunto de usuarios del sistema
- U'es el conjunto de usuarios que han puntuado al ítem i y al ítem j
- I es el conjunto de ítems del sistema
- $R_{u,i}$ es la puntuación que le dio el usuario u al ítem i
- R_i es la puntuación promedio del ítem i
- $\overline{R_u}$ es la puntuación promedio del usuario u
- $S_{i,j}$ es la similitud entre el ítem i y el ítem j

Similitud basada en el coseno: En este método, los dos ítems son considerados como dos vectores en el espacio dimensional del usuario activo[8]. La similitud entre ellos se calcula computando el coseno del ángulo formado por los dos vectores[8]:

$$S_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i}.\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

donde "." denota el producto vectorial de los dos vectores.

Similitud basada en la correlación de Pearson: En este método, la similitud entre dos ítems se calcula computando la correlación de Pearson entre ambos ítems[8]. Para que se compute adecuadamente la correlación, se debe aislar aquellos casos en los cuales los usuarios han puntuado a ambos ítems[8]:

$$S_{i,j} = \frac{\sum_{u \in U'} (R_{u,i} - R_i)(R_{u,j} - R_j)}{\sqrt{\sum_{u \in U'} (R_{u,i} - R_i)^2} \sqrt{\sum_{u \in U'} (R_{u,j} - R_j)^2}}$$

Similitud basada en el coseno ajustado: El método de cálculo de similitud basada en el coseno tiene la desventaja que no se tiene en cuenta las diferencias en la escala de puntuaciones entre diferentes usuarios, por ejemplo, un usuario que siempre da puntuaciones altas y otro que siempre da puntuaciones bajas[8]. El método de cálculo de similitud basada en el coseno ajustado aplaca esta desventaja[8]:

$$S_{i,j} = \frac{\sum_{u \in U'} (R_{u,i} - \overline{R_u})(R_{u,j} - \overline{R_u})}{\sqrt{\sum_{u \in U'} (R_{u,i} - \overline{R_u})^2} \sqrt{\sum_{u \in U'} (R_{u,j} - \overline{R_u})^2}}$$

A continuación se describen algunos métodos para computar la *predicción* de un ítem para un usuario[8]. Uno de los pasos más importantes en los sistemas de filtrado colaborativo es generar las predicciones, para ello se utiliza la siguiente nomenclatura[8]:

- $P_{u,i}$ es la predicción del ítem i para el usuario u

Suma ponderada: Este método computa la predicción de un ítem para el usuario activo calculando la suma de las puntuaciones brindadas por el usuario sobre aquellos ítems similares al ítem en cuestión[8]. Cada puntuación es ponderada por la correspondiente similitud entre los ítems[8]:

$$P_{u,i} = \frac{\sum_{todos\ los\ recursos\ similares,N} (s_{i,N} * R_{u,N})}{\sum_{todos\ los\ recursos\ similares,N} (|s_{i,N}|)}$$

Regresión: Este método es similar al método de suma ponderada pero en vez de utilizar directamente las puntuaciones de ítems similares, utiliza una aproximación de las puntuaciones basada en un modelo de regresión[8].

2.4 Sistemas híbridos de recomendación

Los sistemas híbridos de recomendación combinan dos o más técnicas de recomendación para obtener lo mejor de cada técnica y minimizar sus desventajas, de esta manera se logra una mejor performance[11]. Comúnmente se combina filtrado colaborativo con alguna otra técnica de recomendación[11]. Se emplean diversos métodos de hibridación:

- **Ponderado** (Weighted): Un sistema híbrido de recomendación ponderado es aquel sistema en que el puntaje para un ítem a recomendar es computado a partir de los resultados de todas las técnicas de recomendación disponibles en el sistema[11]. Por ejemplo, se podría hacer una combinación lineal de los puntajes de recomendación[11]. Una ventaja de estos sistemas es que es fácil ajustar el híbrido para que éste brinde mejores resultados[11].

- **Conmutador** (Switching): Un sistema híbrido de recomendación conmutador es aquel sistema que conmuta entre las diferentes técnicas de recomendación dependiendo de la situación actual[11]. Estos sistemas presentan una complejidad extra al proceso de recomendación, que es la determinación del criterio para conmutar entre las diferentes técnicas[11].

- **Mixto** (Mixed): Un sistema híbrido de recomendación mixto es aquel sistema en que se presentan al mismo tiempo las recomendaciones obtenidas de las diferentes técnicas[11].

- **Combinación de características** (Feature combination): Un sistema híbrido de recomendación con combinación de características es aquel sistema en que las características que se obtienen de las diferentes técnicas de recomendación se combinan en un único algoritmo de recomendación[11].

- **En cascada** (Cascade): Un sistema híbrido de recomendación en cascada es aquel sistema en el cual primero se aplica una técnica de recomendación para generar recomendaciones candidatas y luego se aplica una segunda técnica que refina dichas recomendaciones[11].

- **Aumento de características** (Feature augmentation): Un sistema híbrido de recomendación con aumento de características es aquel sistema en el cual primero se aplica una técnica de recomendación para producir un puntaje o una clasificación de un ítem y luego esta información se incorpora al proceso de recomendación de la siguiente técnica[11]. En otras palabras, la salida de una técnica de recomendación se utiliza como la entrada de otra técnica[11].

- **Nivel-sobre-nivel** (Meta-level): Un sistema híbrido de recomendación nivel-sobre-nivel es aquel sistema en que el modelo generado por una técnica de recomendación es utilizado como la entrada de otra técnica de recomendación[11].

2.5 Redes sociales

Las redes sociales son estructuras compuestas por personas, grupos u organizaciones que se conectan entre sí por medio de una o varias relaciones en un contexto social específico[14]. En otras palabras, las redes sociales son formas de organización de la actividad humana[14]. En los últimos años, el número de usuarios en Internet ha crecido sustancialmente[15], así como la utilización de redes sociales en la web. Actualmente millones de personas utilizan redes sociales en la web[16]. Uno de los casos más notorios es el de *Facebook*, en el cual hay un crecimiento exponencial del número de usuarios activos en la red social desde su lanzamiento en la web, como se puede apreciar en la Figura 2, la cual fue creada utilizando valores obtenidos de la propia página de Facebook[17].

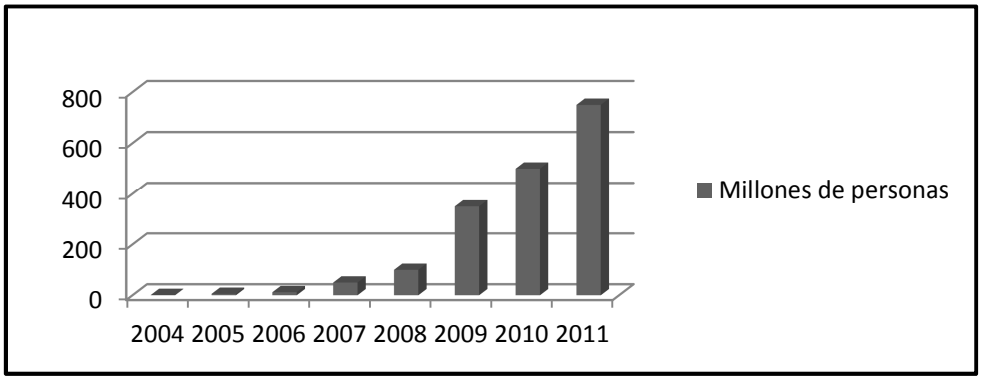

Figura 2: Usuarios activos en Facebook

Hoy en día existe una gran variedad de tipos de redes sociales, una posible clasificación es la siguiente[18]:

- **De interés general**: En esta categoría se encuentran redes sociales populares como *Facebook*, *Twitter* y *MySpace*.

- **Para compartir videos**: Una de las redes sociales más utilizadas es *Youtube*, en la cual los usuarios pueden visualizar, subir y compartir videos.

- **Para compartir fotografías**: En esta categoría se encuentra *Flickr*, que es una de las redes sociales más conocidas para compartir fotografías.

- **Para profesionales**: Una de las redes sociales de esta categoría es *LinkedIn*, que es una red social en la cual los usuarios pueden vincularse profesionalmente.

Por otra parte, los sistemas de recomendaciones necesitan de conocimiento para poder generar recomendaciones que sean de interés para el usuario[8]. Este conocimiento puede ser obtenido de procesos sociales[3], como son los mecanismos de interacción entre personas en una red social. En este proyecto se parte de la hipótesis que la información contenida en el contexto social obtenido de las redes sociales es importante para que los sistemas de recomendaciones puedan generar recomendaciones relevantes. También se asume que es una forma válida de acotar el

problema, debido a que es posible obtener de otras fuentes otro tipo de información que sea de utilidad para generar recomendaciones personalizadas.

2.6 Crowdsourcing

Crowdsourcing es un término utilizado para referirse al reclutamiento en línea de personas, para que éstas completen tareas que son extensas o complejas para ser llevadas a cabo por una sola persona[2]. Crowdsourcing enfatiza el potencial de las comunidades para el desarrollo del conocimiento[18]. En las aplicaciones que utilizan crowdsourcing, una gran cantidad de personas realizan individualmente pequeñas contribuciones agregando medidas o soluciones al valor global de los contenidos[2]. Estas aplicaciones pueden tener éxito si logran atraer a una gran comunidad de colaboradores. Algunos de los ejemplos más conocidos son:

- **Wikipedia**: Es un sitio web en el cual los usuarios crean nuevos contenidos y amplían los contenidos ya existentes, agregando información y corrigiendo errores.

- **Youtube**: Es un sitio web en el cual los usuarios suben, comparten y califican videos.

2.7 Friendsourcing

Friendsourcing es una forma de crowdsourcing que recolecta información adecuada, disponible sólo para un confiable y eventualmente reducido grupo de personas que están conectadas socialmente en una red[2]. Friendsourcing recupera información social en un contexto social para producir resultados de interés[2]. Crowdsourcing y friendsourcing parecen sinónimos o representar lo mismo, pero en realidad son enfoques distintos[18].

Friendsourcing está basado en el potencial de un eventualmente reducido grupo de amigos que tienen intereses similares y que están socialmente conectados en una red[18]. En friendsourcing no hay sabiduría de la multitud pero sí hay sabiduría de un grupo de personas cuidadosamente selecto, lo cual se convierte en una fuente de información de alta calidad[18]. Por ejemplo, en *Facebook*, se brinda a los usuarios recomendaciones para participar de eventos o unirse a determinados grupos de personas.

Friendsourcing también presenta desventajas, cuando se tienen pocas conexiones sociales o cuando los amigos no han hecho muchas valuaciones de recursos las recomendaciones pueden no ser relevantes para el usuario.

Capítulo 3

Especificación del diseño del algoritmo de recomendación

En este capítulo se define el problema a resolver y los requerimientos funcionales y no funcionales relevados en la etapa de análisis. Luego se especifica el diseño del algoritmo de recomendación basado en friendsourcing que utiliza el contexto social obtenido de las redes sociales para realizar recomendaciones personalizadas.

3.1 Problema a resolver

Los mecanismos para utilizar el contexto social personalizando las recomendaciones están enmarcados dentro del servicio de recomendaciones a construir por parte de los integrantes del proyecto *LACCIR QHIR*[19], en el cual, como se mencionó anteriormente, se desarrolla una red social en el entorno médico *Red Unisalud* y un sistema de calificación y recomendación de recursos utilizando el contexto social obtenido de la red social. El servicio de recomendaciones toma como entradas: las calificaciones de los recursos, información de los recursos, el perfil de usuario, el contexto social del usuario y luego mediante algoritmos brinda recomendaciones de recursos ordenados por relevancia[19]. Esto se muestra esquemáticamente en la figura 3[19].

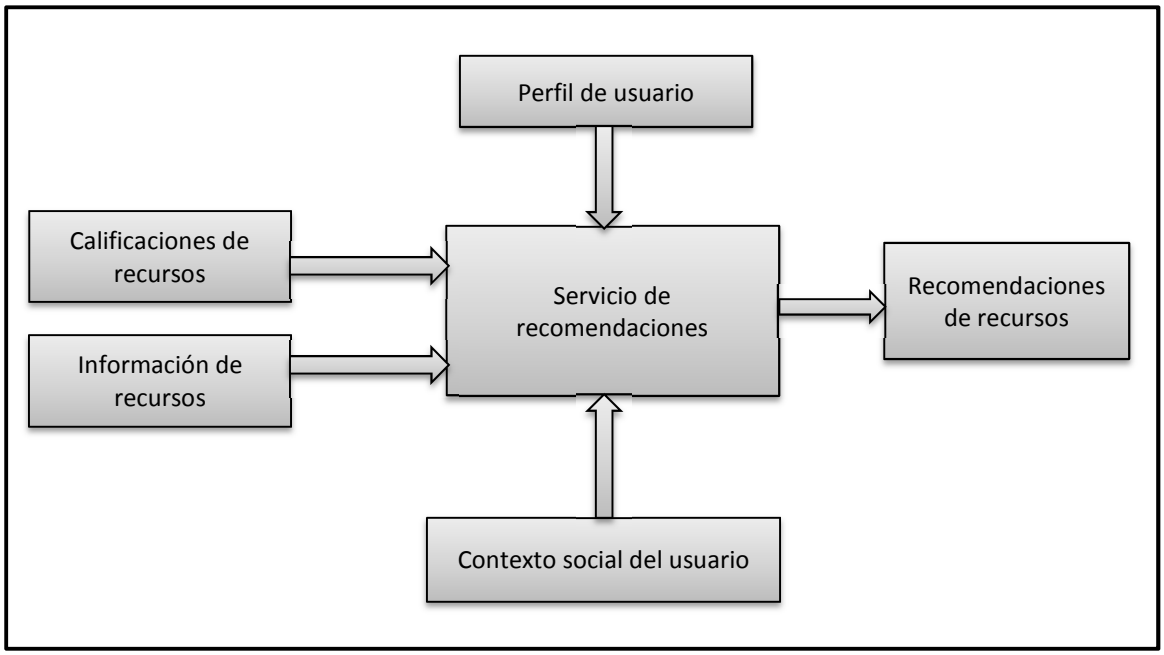

Figura 3: Esquema conceptual de un servicio de recomendaciones.

Para que el servicio funcione se propone usar un algoritmo de recomendación basado en friendsourcing que tome información de una red social en un entorno médico, así como las valuaciones de recursos hechas por los usuarios y dado el identificador de un usuario el algoritmo devuelva una lista de identificadores de recursos recomendados y a su vez que estén ordenados por relevancia. El término *recurso*, se refiere a cualquier objeto (imagen, video, artículo, etc.) que tenga una URI (Uniform Resource Identifier). En un entorno médico, estos recursos pueden ser por ejemplo artículos médicos o videos de prevención de enfermedades. A continuación se describen los requerimientos relevados.

3.2 Requerimientos

El algoritmo de recomendación deberá satisfacer los requerimientos funcionales y no funcionales relevados en las reuniones con los integrantes del proyecto LACCIR QHIR, encargados de definir e implementar la arquitectura para el sistema de calificación y recomendación de recursos.

3.2.1 Funcionales

- **Fuentes de información**: El algoritmo de recomendación debe tener en cuenta las siguientes fuentes de información para realizar las recomendaciones personalizadas: la red social a la que pertenece el usuario al que se le quiere brindar las recomendaciones, las valuaciones de recursos hechas por los usuarios e información del perfil de usuario. En cuanto a las calificaciones de recursos, los usuarios pueden calificar hasta cuatro atributos de calidad que son: trustwhorthy (confiable), objective (objetivo), complete (completo) y well-written (bien escrito).

- **Salida del algoritmo de recomendación**: El algoritmo de recomendación debe devolver una lista de identificadores recursos recomendados para un usuario en particular, y el resultado debe estar ordenado en función de la relevancia para el usuario.

3.2.2 No funcionales

- **Tecnología**: Se requiere que se utilice un software no propietario para el desarrollo del algoritmo de recomendación. Esto es debido a que en caso de que en el futuro se quiera extender el desarrollo del algoritmo, no haya impedimentos en cuanto a licencias. Por lo tanto en las reuniones se sugirió la opción de desarrollar el algoritmo de recomendación en Java o en Ruby.

- **Adaptabilidad**: Se requiere que el algoritmo de recomendación a construir esté preparado a readaptarse a diferentes contextos sociales. Esto implica que se pueda utilizar el algoritmo para brindar recomendaciones a un usuario que pertenezca a distintas redes sociales.

3.3 Diseño del algoritmo

El algoritmo de recomendación está pensado para ser usado en el contexto de una red social, lo cual implica que los usuarios son miembros de una red social y tienen relaciones entre ellos.

3.3.1 Modelado de la red social

Actualmente existe una gran cantidad de redes sociales con características y objetivos particulares. En algunas de ellas se brinda a los usuarios la oportunidad de elegir el tipo de relación que los une. Por ejemplo, en la red social *Facebook*, luego que un usuario agrega a otro como amigo, tiene la opción de elegir de manera más especifica el tipo de relación que lo une con dicho amigo, entre las opciones disponibles se encuentran: *conocidos*, *mejores amigos* u otros tipos de relaciones. Por simplicidad, el algoritmo de recomendación propuesto no tiene en cuenta dicha posibilidad y sólo considera un tipo de relación, la *amistad*. Diferentes tipos de relaciones pueden ser consideradas en una versión general del algoritmo, pero como primer enfoque todas las relaciones son consideradas del mismo tipo. Por otra parte, si bien se considera a la amistad como el único tipo de relación entre usuarios de una red social, no todas las relaciones tienen el mismo valor o peso a la hora de brindar opiniones o recomendaciones. Esto significa que no tiene el mismo valor la opinión brindada por un amigo directo que la opinión brindada por el amigo de un amigo, con lo cual se pueden distinguir diferentes niveles de amistad. Es deseable que el algoritmo se pudiera utilizar independientemente del tipo de red social, con lo cual la información contenida en una red social puede ser modelada con un grafo no dirigido, donde sus vértices o nodos son los usuarios miembros de la red y las aristas o arcos son las relaciones de amistad existentes entre ellos. Esto se puede apreciar en la figura 4.

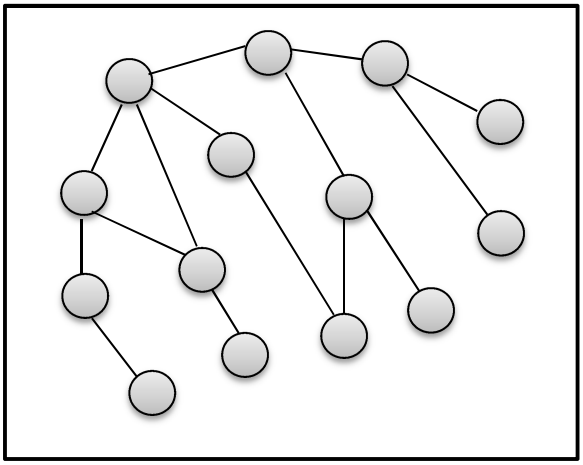

Figura 4: Modelado simple de una red social

3.3.1.1 Perfil de usuario

Cada usuario en la red social debe tener un identificador para que las recomendaciones brindadas por el algoritmo puedan ser personalizadas. En este proyecto, el algoritmo de recomendación se va a utilizar en el contexto de una sola red social, *Red Unisalud*, que es la red desarrollada en el marco del proyecto LACCIR. En el caso de que se quisiera utilizar más de una red social, cada usuario debería tener, aparte del identificador dentro cada red social, un identificador a nivel de todas las redes sociales, esto se puede lograr teniendo un mapeo entre los identificadores de las redes sociales y los identificadores globales. Cada perfil de usuario está compuesto por el identificador dentro de la red social y por información del usuario.

3.3.1.2 Amigos de usuario

Cada usuario en la red social tiene un conjunto de amigos. De cada uno de ellos se registra el identificador del amigo y el tipo de vínculo entre ellos. Si bien este último campo no es considerado por el algoritmo (como se explicó anteriormente), éste puede ser considerado en futuras versiones.

3.3.2 Modelado de las valuaciones de recursos

El sistema de calificación desarrollado en el proyecto LACCIR QHIR brinda a los usuarios de la red social Unisalud la oportunidad de calificar los atributos de calidad de los recursos. Los recursos en un entorno médico pueden ser artículos médicos, videos de prevención de enfermedades, etc. En base a los requerimientos, se conoce que el conjunto de recursos a evaluar es un conjunto finito y no variable. Es necesario que cada recurso tenga un identificador que permita distinguirlo entre los demás recursos. A su vez, cada recurso posee cuatro atributos de calidad que pueden ser valuados por los usuarios, estos son:

- **Trustwhorthy** (confiable)
- **Objective** (objetivo)
- **Complete** (completo)
- **Well-written** (bien escrito)

Estos atributos de calidad fueron definidos por los encargados de desarrollar el sistema de calificación de recursos. Cada usuario puede valuar más de una vez el mismo recurso, pero el sistema de valuación no guarda el histórico de dichas valuaciones y sólo mantiene el registro de la última vez que el usuario puntúo los atributos de dicho recurso. La valuación de cada atributo de un recurso consiste en un valor entero entre 0 y 10, siendo 0 la peor calificación y 10 la mejor. Para cada usuario, se registran las valuaciones de los recursos indicando el identificador de cada recurso junto con las puntuaciones de los atributos de calidad.

3.3.3 Modelado de la importancia de los atributos de calidad de los recursos

Si bien cada usuario tiene la oportunidad de calificar distintos atributos de calidad de un recurso, a la hora de establecer una puntuación global del recurso por parte de un usuario, cada atributo puede tener un peso (weight) distinto a los demás en dicha puntuación. Por ejemplo, si los recursos disponibles son videos médicos, el atributo *well-written* (bien escrito) puede carecer de importancia, con lo cual sería de interés poder desestimar las puntuaciones en dicho atributo. Esto se puede modelar asignando a cada atributo de calidad un peso o un valor de ponderación.

Cada peso se define como un real que debe cumplir:

$$0.0 \leq weight_{\text{trustwhorthy}}, weight_{\text{objective}}, weight_{\text{complete}}, weight_{\text{well-written}} \leq 1.0$$

$$weight_{\text{trustwhorthy}} + weight_{\text{objective}} + weight_{\text{complete}} + weight_{\text{well-written}} = 1.0$$

3.3.4 Datos base y datos de entrada

Los sistemas de recomendaciones utilizan datos base y datos de entrada para realizar las recomendaciones, entonces es necesario definir primero dichas fuentes de información. Se define que los datos base son la información básica acerca de la red social, es decir información de los usuarios miembros de la red y sus perfiles de usuario. El primer dato de entrada es el identificador del usuario al que se le quiere brindar recomendaciones de recursos personalizadas, llamado *usuario activo*. El segundo dato de entrada es el conjunto de valuaciones de recursos hechas por los amigos del usuario activo. El tercer dato de entrada es información acerca de la importancia de los atributos de calidad de un recurso. El último dato de entrada es un valor que indica la cantidad máxima de recursos a recomendar. En la tabla 2 se muestra un resumen de los datos base y de entrada que considera el algoritmo de recomendación propuesto.

Datos base	Datos de entrada
Usuarios de la red social y sus perfiles	▪ Identificador del usuario activo. ▪ Conjunto de valuaciones hechas por los amigos del usuario activo. ▪ Importancia de los atributos de calidad de un recurso. ▪ Cantidad máxima de recursos a recomendar.

Tabla 2: Datos base y de entrada del algoritmo de recomendación

3.3.5 Proceso de recomendación

En esta sección se describe la manera en que el algoritmo procesa los datos base y los de entrada para obtener las recomendaciones personalizadas. En crowdsourcing se toma el conocimiento de las masas para hacer las recomendaciones, en este proyecto el algoritmo a desarrollar está basado en friendsourcing, con lo cual toma información adecuada de un confiable y reducido grupo de personas conectadas en la red social. Uno de los puntos críticos de este algoritmo es la selección del grupo de personas que se van a utilizar para generar las recomendaciones personalizadas. Como se mencionó anteriormente, la red social puede ser modelada como un grafo no dirigido, donde los nodos son los usuarios de la red y las aristas son las relaciones de amistad entre ellos. El primer paso del algoritmo es obtener para el usuario al que se le quiere brindar las recomendaciones (usuario activo), el conjunto de amigos del primer nivel de dicho grafo, es decir sus amigos directos.

En la figura 5, el usuario activo está marcado en azul y los amigos del primer nivel en verde. La elección de este grupo de personas se debe a que mientras más cercanos estén al usuario activo, más confiables son sus opiniones y sugerencias. En un corto tiempo de funcionamiento del sistema, es probable que no haya una cantidad suficiente de amigos por lo cual, si es necesario se obtendrían los amigos del siguiente nivel (que serían los amigos de sus amigos), y así sucesivamente.

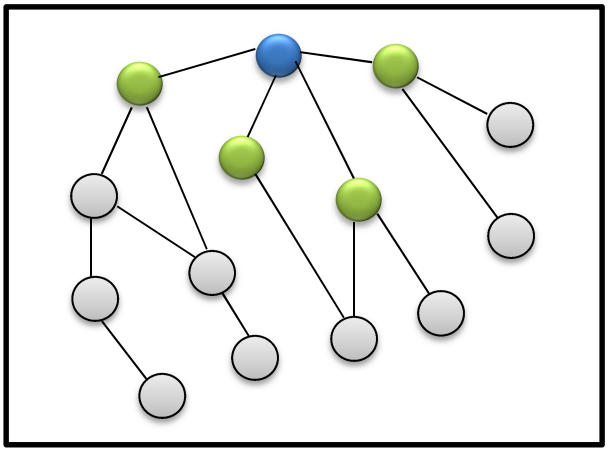

Figura 5: Obtención de los amigos del primer nivel

Luego que se obtiene dicho grupo de personas, es de interés seleccionar aquellas que son más similares al usuario activo. Esto se debe a que el usuario activo confiaría más en aquellas opiniones de amigos que son más similares a él. Hay que considerar el hecho de que en este proyecto no se tiene en cuenta el perfil de usuario como fuente de información para encontrar usuarios similares, debido a que el perfil de usuario podría variar dependiendo la red social y uno de los objetivos de este proyecto es proveer un algoritmo que se pueda adaptar a cualquier red social con lo cual se busca generalidad.

A continuación se describen en detalle los tres tipos de similitud que considera el algoritmo de recomendación:

- Similitud en las relaciones de amistad.
- Similitud en las valuaciones de recursos.
- Similitud de la actividad en las valuaciones de recursos.

Similitud en las relaciones de amistad: Este factor se refiere a la similitud en la relaciones de amistad considerando la cantidad de amigos en común. A partir de la cantidad de amigos que tenga un amigo en la red, mientras más amigos en común con dicho amigo, más similares son entre sí.

Dados:

- u_{active} al usuario activo, es decir al usuario al que se le quiere brindar las recomendaciones.

- u_{friend} a un usuario amigo del usuario activo.

- $cantAmigos_{u_{friend}}$ es la cantidad de amigos del usuario amigo del usuario activo.

- $cantAmigosComun$ es la cantidad de amigos en común.

Se define el *puntaje de similitud en las relaciones de amistad* $(sim_{friendship})$ entre el usuario activo y un usuario amigo, como un valor real que cumple lo siguiente:

$$\begin{cases} sim_{friendship}(u_{active}, u_{friend}) = 0.0, & si\ cantAmigos_{u_{friend}} \leq 1.0 \\ sim_{friendship}(u_{active}, u_{friend}) = \dfrac{cantAmigosComun}{cantAmigos_{u_{friend}} - 1}, & en\ otro\ caso \end{cases}$$

$$\text{donde } 0.0 \leq sim_{friendship}(u_{active}, u_{friend}) \leq 1.0$$

Similitud en las valuaciones de recursos: Este factor se refiere a la similitud en las valuaciones de recursos, esto significa que si un usuario valúa los mismos recursos de la misma manera (positivamente o negativamente) que el usuario activo, entonces existe una mayor probabilidad que valúen otros recursos de manera similar, esto implica que tienen gustos o preferencias similares.

Dados:

- u_{active} al usuario activo, es decir al usuario al que se le quiere brindar las recomendaciones.

- u_{friend} a un usuario amigo del usuario activo.

- $valuation\text{Trustwhorthy}_{i,u_{active}}$ es la valuación del atributo de calidad *trustwhorthy* del recurso *i* por parte del usuario activo.

- $valuation\text{Trustwhorthy}_{i,u_{friend}}$ es la valuación del atributo de calidad *trustwhorthy* del recurso *i* por parte del usuario amigo del usuario activo.

- $valuation\text{Objective}_{i,u_{active}}$ es la valuación del atributo de calidad *objective* del recurso *i* por parte del usuario activo.

- $valuation\text{Objective}_{i,u_{friend}}$ es la valuación del atributo de calidad *objective* del recurso *i* por parte del usuario amigo del usuario activo.

- $valuation\text{Complete}_{i,u_{active}}$ es la valuación del atributo de calidad *complete* del recurso *i* por parte del usuario activo.

- $valuation\text{Complete}_{i,u_{friend}}$ es la valuación del atributo de calidad *complete* del recurso *i* por parte del usuario amigo del usuario activo.

- $valuation\text{Wellwritten}_{i,u_{active}}$ es la valuación del atributo de calidad *well-written* del recurso *i* por parte del usuario activo.

- $valuation\text{Wellwritten}_{i,u_{friend}}$ es la valuación del atributo de calidad *well-written* del recurso *i* por parte del usuario amigo del usuario activo.

- $cantRecursosComun$ es la cantidad de recursos valuados en común.

Se define el *puntaje en la similitud en las valuaciones de recursos* ($sim_{valuations}$) entre el usuario activo y un usuario amigo, como un valor real que cumple lo siguiente:

$$\begin{cases} sim_{valuations}(u_{active}, u_{friend}) = 0.0, & si\ cantRecursosComun = 0 \\ sim_{valuations}(u_{active}, u_{friend}) = \dfrac{\sum_{i=1}^{cantRecursosComun}\left(\dfrac{\sum_{j=1}^{4}|dif(i,j)|}{4}\right)}{cantRecursosComun}, & en\ otro\ caso \end{cases}$$

$$\text{donde } 0.0 \leq sim_{valuations}(u_{active}, u_{friend}) \leq 1.0$$

$dif(i,j)$ es la diferencia de las valuaciones del atributo *j* del recurso *i* hechas por el usuario activo y el usuario amigo del usuario activo, teniendo en cuenta que el recurso *i* es un recurso valuado por ambos y que el atributo de calidad *j* corresponde a un atributo de los cuatro definidos anteriormente (trustwhorthy, objective, complete, well-written).

Similitud de la actividad en las valuaciones de recursos: Este factor se refiere a la similitud de la actividad en las valuaciones de recursos, esto significa que un amigo que previamente haya valuado un largo conjunto de recursos es más confiable porque su comportamiento es mejor descrito a través de sus puntuaciones que otros amigos con menor actividad. Usuarios con una gran actividad en las valuaciones de recursos probablemente han visto más recursos con lo cual brinda valuaciones más confiables.

Dados:

- u_{active} al usuario activo, es decir al usuario al que se le quiere brindar las recomendaciones.
- u_{friend} a un usuario amigo del usuario activo.
- $cantRecursos_{u_{active}}$ es la cantidad de recursos valuados por el usuario activo.
- $cantRecursos_{u_{friend}}$ es la cantidad de recursos valuados por el usuario amigo del usuario activo.

Se define el *puntaje de la actividad en las valuaciones de recursos* ($sim_{activity}$) entre el usuario activo y un usuario amigo, como un valor real que cumple lo siguiente:

$$\begin{cases} sim_{activity}(u_{active}, u_{friend}) = 0.0, & si((cantRecursos_{u_{active}} = 0) \vee (cantRecursos_{u_{friend}} = 0)) \\ sim_{activity}(u_{active}, u_{friend}) = \dfrac{cantRecursos_{u_{friend}}}{cantRecursos_{u_{active}}}, & si\ cantRecursos_{u_{active}} \geq cantRecursos_{u_{friend}} \\ sim_{activity}(u_{active}, u_{friend}) = \dfrac{cantRecursos_{u_{active}}}{cantRecursos_{u_{friend}}}, & si\ cantRecursos_{u_{active}} < cantRecursos_{u_{friend}} \end{cases}$$

$$\text{donde } 0.0 \leq sim_{activity}(u_{active}, u_{friend}) \leq 1.0$$

Una vez que se definieron los tres factores, se puede definir la similitud entre dos usuarios de la red social, $sim(u_{active}, u_{friend})$ como un valor real que representa cuan similar es un usuario amigo con respecto al usuario activo y es calculado de la siguiente forma:

$$\begin{aligned} sim(u_{active}, u_{friend}) = \ & \alpha * sim_{friendship}(u_{active}, u_{friend}) + \\ & \beta * sim_{valuations}(u_{active}, u_{friend}) + \\ & \gamma * sim_{activity}(u_{active}, u_{friend}) \end{aligned}$$

$$\begin{aligned} \text{donde} \quad & 0.0 \leq sim(u_{active}, u_{friend}) \leq 1.0 \\ & 0.0 \leq \alpha, \beta, \gamma \leq 1.0 \\ & \alpha + \beta + \gamma = 1.0 \end{aligned}$$

α, β, γ son ponderadores de cada uno de los factores definidos anteriormente. Estos valores representan la importancia de cada factor.

La principal ventaja de usar ponderadores es que brinda la oportunidad de desestimar cualquier factor lo cual permite una mayor flexibilidad a la hora de testear el algoritmo y de analizar los resultados.

En un paso posterior, el algoritmo ordena los amigos seleccionados, en base al cálculo del valor de similitud con el usuario activo. Luego se itera sobre dicho conjunto, seleccionando en cada paso de la iteración el amigo más similar al usuario activo u_{friend} y se obtienen los recursos mejores puntuados por dicho usuario. La puntuación de un recurso se calcula como la suma ponderada de las valuaciones de atributos de calidad de los recursos hechas por el usuario u_{friend} de la siguiente forma:

$$\begin{aligned} puntuacion(i, u_{friend}) = \\ = peso_{trustwhorthy} * valuation\text{Trustwhorthy}_{i,u_{friend}} \\ + peso_{objective} * valuation\text{Objective}_{i,u_{friend}} \\ + peso_{complete} * valuation\text{Complete}_{i,u_{friend}} \\ + peso_{well-written} * valuation\text{Wellwritten}_{i,u_{friend}} \end{aligned}$$

donde $weight_{trustwhorthy} + weight_{objective} + weight_{complete} + weight_{well-written} = 1.0$

$$0.0 \leq weight_{trustwhorthy} + weight_{objective} + weight_{complete} + weight_{well-written} \leq 1.0$$

Como se explicó anteriormente, se definen pesos (weights) que definen la importancia de cada atributo de calidad en el cálculo de la puntuación global del recurso. En base al cálculo de esta puntuación se ordenan todos los recursos valuados por el usuario u_{friend}. Luego, el algoritmo toma ordenadamente (por orden de mayor a menor puntuación) cada recurso y hace lo siguiente: si la puntuación calculada es mayor o igual a un cierto valor de aceptación y el usuario activo u_{active} no valuó dicho recurso, entonces el identificador del recurso se agrega a la lista de recursos recomendados. Este valor de aceptación, llamado *umbral de aceptación* impone un mínimo nivel de calidad en los recursos recomendados.

$$\boxed{0.0 \leq umbral_{aceptacion} \leq 1.0}$$

Este proceso se repite hasta que se alcanza la máxima cantidad de recursos recomendados para el usuario activo u_{active}.

3.3.6 Datos de salida

La salida del algoritmo de recomendación es una lista ordenada por relevancia de los identificadores de los recursos recomendados para el usuario activo u_{active}.

Capítulo 4

Implementación del algoritmo de recomendación

Dado que una parte importante del proyecto era proveer un módulo de recomendación que fuera utilizado por un sistema real, en este capítulo se describe la implementación del algoritmo de recomendación especificado en el capítulo anterior. También se explica cómo se realizó la integración del algoritmo con el sistema de calificación y recomendación desarrollado en el marco del proyecto LACCIR QHIR.

4.1 Lenguaje de programación

Para el desarrollo del algoritmo de recomendación se utilizó *Java* como lenguaje de programación, debido a la familiaridad de uso a lo largo de la carrera y a que cumplía con el requerimiento no funcional de ser un software no propietario. En base a las reuniones tenidas en La Plata con los integrantes del proyecto LACCIR QHIR, se acordó que se enviara un proyecto Eclipse con el conjunto de clases en Java que implementaran el algoritmo de recomendación, para facilitar la integración del algoritmo con el sistema de calificación y recomendación que se estaba desarrollando.

4.2 Fuentes de información

El algoritmo de recomendación necesita de algunas fuentes de información para poder generar las recomendaciones personalizadas. Estas fuentes son: la *red social* del usuario al que se le quiere brindar las recomendaciones, las *valuaciones de recursos* que hicieron los usuarios miembros de la red social, y los *pesos de los atributos de calidad de los recursos*. Se diseñó que el algoritmo de recomendación tomara dichas fuentes de información mediante la utilización de archivos xml:

- **Redsocial.xml**
- **Valuaciones.xml**
- **Atributos.xml**

Las principales ventajas de utilizar archivos xml para las fuentes de información son que facilita la adición de nuevas etiquetas en el futuro y que resulta sencillo entender la estructura y procesarla.

4.2.1 Redsocial.xml

Este archivo contiene información de la red social a la que pertenece el usuario, esto es, un conjunto de usuarios donde para cada usuario se conoce su perfil de usuario formado por su identificador dentro de la red social y una lista de atributos, y la lista de sus amigos en la red social.

A modo de ejemplo, supongamos que en una red social el usuario *dgonzalez*, que es un estudiante de *grado*, sólo tiene un vínculo de amistad de tipo *académico* con el usuario *lmendoza* y que el usuario *ividal*, que también es un estudiante de *grado*, tiene tres vínculos de amistad de tipo académico con los usuarios *rguisandez*, *lmendoza* y *msarmiento*.

Un fragmento del archivo **Redsocial.xml** sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
- <RedSocial>
  - <Usuario>
    - <PerfilUsuario>
        <Identificador>dgonzalez</Identificador>
      - <ListaAtributos>
          <Atributo>Grado</Atributo>
        </ListaAtributos>
      - <ListaAmigos>
        - <Amigo>
            <Identificador>lmendoza</Identificador>
            <TipoVinculo>Academico</TipoVinculo>
          </Amigo>
        </ListaAmigos>
      </PerfilUsuario>
    </Usuario>
  - <Usuario>
    - <PerfilUsuario>
        <Identificador>ividal</Identificador>
      - <ListaAtributos>
          <Atributo>Grado</Atributo>
        </ListaAtributos>
      - <ListaAmigos>
        - <Amigo>
            <Identificador>rguisandez</Identificador>
            <TipoVinculo>Academico</TipoVinculo>
          </Amigo>
        - <Amigo>
            <Identificador>lmendoza</Identificador>
            <TipoVinculo>Academico</TipoVinculo>
          </Amigo>
        - <Amigo>
            <Identificador>msarmiento</Identificador>
            <TipoVinculo>Academico</TipoVinculo>
          </Amigo>
        </ListaAmigos>
      </PerfilUsuario>
    </Usuario>
```

4.2.2 Valuaciones.xml

Este archivo contiene información de las valuaciones de recursos hechas por los usuarios miembros de la red social. Para cada usuario se registra la lista de valuaciones de recursos realizadas por dicho usuario. Por cada valuación se registra el identificador del recurso valuado y el puntaje dado para cada atributo de calidad del recurso. El puntaje es un valor entero entre 0 y 10.

A modo de ejemplo, supongamos que el usuario *dgonzalez* calificó los atributos de calidad de dos páginas web.

A la página web cuyo identificador es http://www.unlp.edu.ar le dio las siguientes puntuaciones: trustwhorthy (confiable) = 7, objective (objetivo) = 9, well-written (bien escrito) = 2 y complete (completo) = 1.

A la página web cuyo identificador es http://info.unlp.edu.ar le dio las siguientes puntuaciones: trustwhorthy (confiable) = 9, objective (objetivo) = 1, well-written (bien escrito) = 8 y complete (completo) = 6.

Un fragmento del archivo **Valuaciones.xml** sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
- <EvaluacionesRecursos>
  - <Usuario>
      <Identificador>dgonzalez</Identificador>
    - <ListaEvaluaciones>
      - <Recurso>
          <Identificador>http://www.unlp.edu.ar/</Identificador>
        - <Valuacion>
            <Atributo>Trustwhorthy</Atributo>
            <Puntaje>7</Puntaje>
          </Valuacion>
        - <Valuacion>
            <Atributo>Objective</Atributo>
            <Puntaje>9</Puntaje>
          </Valuacion>
        - <Valuacion>
            <Atributo>Well-written</Atributo>
            <Puntaje>2</Puntaje>
          </Valuacion>
        - <Valuacion>
            <Atributo>Complete</Atributo>
            <Puntaje>1</Puntaje>
          </Valuacion>
        </Recurso>
      - <Recurso>
          <Identificador>http://info.unlp.edu.ar/</Identificador>
        - <Valuacion>
            <Atributo>Trustwhorthy</Atributo>
            <Puntaje>9</Puntaje>
          </Valuacion>
        - <Valuacion>
            <Atributo>Objective</Atributo>
            <Puntaje>1</Puntaje>
          </Valuacion>
        - <Valuacion>
            <Atributo>Well-written</Atributo>
            <Puntaje>8</Puntaje>
          </Valuacion>
        - <Valuacion>
            <Atributo>Complete</Atributo>
            <Puntaje>6</Puntaje>
          </Valuacion>
        </Recurso>
```

4.2.3 Atributos.xml

Este archivo contiene información de los pesos de los atributos de calidad de los recursos. Para cada atributo de calidad se asigna un puntaje real mayor o igual a cero y menor o igual a uno. La suma de los puntajes de todos los atributos de calidad debe ser igual a uno.

A modo de ejemplo, supongamos que los recursos que los usuarios de una red social pueden calificar son artículos médicos y que sólo interesa recomendar artículos bien escritos y completos. A dichos atributos de calidad se le da la misma importancia, asignándole el mismo peso.

Un fragmento del archivo **Atributos.xml** sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
- <ListaAtributos>
  - <Atributo>
      <Nombre>Trustwhorthy</Nombre>
      <Peso>0.00</Peso>
    </Atributo>
  - <Atributo>
      <Nombre>Well-written</Nombre>
      <Peso>0.50</Peso>
    </Atributo>
  - <Atributo>
      <Nombre>Objective</Nombre>
      <Peso>0.00</Peso>
    </Atributo>
  - <Atributo>
      <Nombre>Complete</Nombre>
      <Peso>0.50</Peso>
    </Atributo>
</ListaAtributos>
```

4.3 Proyecto Eclipse

El proyecto Eclipse contiene los siguientes códigos fuentes en Java:

- FriendSourcing.java
- ParserXML.java
- RedSocial.java
- ValuacionesRecursosUsuarios.java
- PesosAtributosCalidad.java
- PerfilUsuario.java
- ListaAmigos.java
- ListaValuaciones.java
- Recomendaciones.java

A continuación se describe cada uno de ellos en detalle.

4.3.1 FriendSourcing.java

Este código fuente contiene el método principal que implementa el algoritmo de recomendación basado en friendsourcing, llamado *algoritmoFriendSourcing*.

```
public ArrayList<String> algoritmoFriendSourcing(String idUsuario,
int cantidadRecursosARecomendar, String rutaRedSocialXML,
String rutaValuacionesRecursosXML, String rutaAtributosCalidadXML)
```

donde:

- *idUsuario*, es el string que identifica al usuario al que se le quiere brindar las recomendaciones personalizadas.

- *cantidadRecursosARecomendar*, es el entero que representa la cantidad máxima de recursos recomendados.

- *rutaRedSocialXML*, es la ruta absoluta donde se encuentra el archivo xml que contiene la red social.

- *rutaValuacionesRecursosXML*, es la ruta absoluta donde se encuentra el archivo xml que contiene las valuaciones de los recursos.

- *rutaAtributosCalidadXML*, es la ruta absoluta donde se encuentra el archivo xml que contiene los pesos de los atributos de calidad.

En este método primero se crea una lista vacía de strings que va a contener los identificadores de los recursos recomendados. Luego utilizando la clase *ParserXML* se parsean los archivos xml creando las clases: *RedSocial* que contiene la red social a la cual pertenece el usuario, *ValuacionesRecursosUsuarios* que contiene las valuaciones de recursos hechas por todos los usuarios de la red social, y *PesosAtributosCalidad* que contiene los pesos de los atributos de calidad.

Una vez que se tienen las clases deseadas, se definen cuatro **parámetros internos**:

- **peso1**: Es el peso que pondera el factor “Similitud de la actividad en las valuaciones de recursos”.

- **peso2**: Es el peso que pondera el factor “Similitud en las valuaciones de recursos”.

- **peso3**: Es el peso que pondera el factor “Similitud en las relaciones de amistad”.

- **umbralAceptacion**: Para que un recurso pueda ser recomendado, la puntuación global dada por el usuario debe ser mayor o igual a dicho umbral de aceptación.

Estos parámetros son valores reales y son definidos por el administrador del sistema. Además deben cumplir lo siguiente:

$$0.0 \leq peso_1, peso_2, peso_3, umbralAceptacion \leq 1.0$$

$$\sum_{i=1}^{3} peso_i = 1.0$$

Inicialmente, a los pesos que ponderan los distintos factores de similitud se les pueden asignar el mismo valor, y al umbral de aceptación se le puede asignar el valor 0.5 para exigir un mínimo nivel de calidad de los recursos recomendados. A medida que se van generando las recomendaciones se puede observar su influencia en los resultados y se van actualizando.

Luego se invoca al método *recomendacionesPersonalizadasRecursos* de la clase *Recomendaciones* con dichos parámetros:

```
public ArrayList<String> recomendacionesPersonalizadasRecursos(String
idUsuario, int cantRecursosARecomendar, float umbralAceptacion,
PesosAtributosCalidad pesos, RedSocial red,
ValuacionesRecursosUsuarios val, float peso1, float peso2, float
peso3)
```

Finalmente el método devuelve una lista con los identificadores de recursos recomendados por el algoritmo de recomendación.

4.3.2 ParserXML.java

Este código fuente contiene tres métodos encargados de parsear los archivos xml y construir las clases correspondientes:

```
public RedSocial crearRedSocialXML(String rutaArchivo)
```

Este método toma como entrada la ruta absoluta donde se encuentra el archivo Redsocial.xml y devuelve un objeto de la clase RedSocial.

```
public ValuacionesRecursosUsuarios
crearValuacionesRecursosUsuariosXML(String rutaArchivo)
```

Este método toma como entrada la ruta absoluta donde se encuentra el archivo Valuaciones.xml y devuelve un objeto de la clase ValuacionesRecursosUsuarios.

```
public PesosAtributosCalidad crearPesosAtributosCalidadXML(String
rutaArchivo)
```

Este método toma como entrada la ruta absoluta donde se encuentra el archivo Atributos.xml y devuelve un objeto de la clase PesosAtributosCalidad.

4.3.3 RedSocial.java

En este código fuente se define la clase *RedSocial* que contiene los usuarios miembros de dicha red y las relaciones de amistad entre ellos. También se proveen las siguientes funciones:

```
public int cantAmigosUsuario(String idUsuario)
```

Dado el identificador de un usuario, lo busca en la red social y devuelve la cantidad de amigos de dicho usuario.

```
public int cantAmigosComun(String idUsuario1, String idUsuario2)
```

Dados los identificadores de dos usuarios de la red social, devuelve la cantidad de amigos en común.

```
public float puntajeActividadRedSocial(String idUsuario1, String
idUsuario2)
```

Dado los identificadores de dos usuarios de la red social, devuelve el puntaje de la actividad en la red social. Este valor indica cuan similares son dos usuarios, teniendo en cuenta los amigos en común.

4.3.4 ValuacionesRecursosUsuarios.java

En este código fuente se define la clase *ValuacionesRecursosUsuarios* que contiene las valuaciones de recursos realizadas por los usuarios de la red social. También se proveen las siguientes funciones:

```
public int cantRecursosPuntuados(String idUsuario)
```

Dado el identificador del usuario, devuelve la cantidad de recursos puntuados.

```
public int cantRecursosPuntuadosEnComun(String idUsuario1, String
idUsuario2)
```

Dado los identificadores de dos usuarios de la red social, devuelve la cantidad de recursos puntuados en común.

```
public float puntajeActividadValuacion(String idUsuario1, String
idUsuario2)
```

Dados los identificadores de dos usuarios de la red social, devuelve el puntaje de la actividad en las valuaciones. Este valor indica cuan similares son dos usuarios, teniendo en cuenta la cantidad de recursos puntuados por ambos.

```
public float puntajeValuacionSimilar(String idUsuario1, String
idUsuario2)
```

Dados los identificadores de dos usuarios de la red social, devuelve el puntaje de las valuaciones similares. Este valor indica cuan similares son dos usuarios, teniendo en cuenta el comportamiento de ambos (positivamente o negativamente) en la valuación de los mismos recursos.

```
public boolean puntuoRecursoUsuario(String idUsuario, String
idRecurso)
```

Dado el identificador de un usuario y el identificador de un recurso, devuelve si dicho usuario lo puntuó o no.

```
public HashMap<String, Float>
obtenerRecursosPuntuacionesUsuario(String idUsuario,
PesosAtributosCalidad pesosAtributos, float umbralAceptacion)
```

Dado el identificador de un usuario, los pesos de los atributos de calidad de los recursos y un valor que indica el umbral de aceptación, devuelve un HashMap con parejas (identificador del recurso, puntaje) tal que el usuario haya puntuado dicho recurso y la puntuación global, teniendo en cuenta los pesos de los atributos de calidad, sea mayor o igual al umbral de aceptación.

4.3.5 PesosAtributosCalidad.java

En este código fuente se define la clase *PesosAtributosCalidad* que contiene los pesos de los atributos de calidad de los recursos. Los atributos de calidad son los siguientes: *confiable* (trustwhorthy), *objetivo* (objective), *completo* (complete) y *well-written* (bien escrito).

Se debe cumplir:

$$peso_{trustwhorthy} + peso_{objective} + peso_{complete} + peso_{well-written} = 1.0$$

4.3.6 PerfilUsuario.java

En este código fuente se define la clase *PerfilUsuario* que contiene el identificador del usuario y una lista de atributos del mismo.

4.3.7 ListaAmigos.java

En este código fuente se define la clase *ListaAmigos* que representa la lista de amigos que puede tener un usuario en la red social.

4.3.8 ListaValuaciones.java

En este código fuente se define la clase *ListaValuaciones* que representa las valuaciones de recursos realizadas por los usuarios. Para cada valuación se registra el identificador del recurso valuado y el puntaje que le dio el usuario a cada atributo de calidad del recurso. Si bien un usuario de la red puede realizar varias valuaciones del mismo recurso, el sistema registra la última valuación realizada.

4.3.9 Recomendaciones.java

En este código fuente se definen tres funciones necesarias para la generación de las recomendaciones personalizadas:

```
public float gradoSimilaridad(RedSocial red,
ValuacionesRecursosUsuarios val, String idUsuario1, String
idUsuario2, float peso1, float peso2, float peso3)
```

Dada la red social, las valuaciones de recursos realizadas por los usuarios de dicha red, los pesos de los factores de similitud (puntajeActividadValuacion, puntajeValuacionSimilar, puntajeActividadRedSocial) y los identificadores de dos usuarios, devuelve el grado de similitud entre dichos usuarios calculado como la suma ponderada de dichos factores:

```
float gradoSimilaridad =
(float) peso1 * val.puntajeActividadValuacion(idUsuario1, idUsuario2)+
(float) peso2 * val.puntajeValuacionSimilar(idUsuario1, idUsuario2)+
(float) peso3 * red.puntajeActividadRedSocial(idUsuario1, idUsuario2)

public ArrayList<String> obtenerUsuariosSimilares(String idUsuario,
RedSocial red, ValuacionesRecursosUsuarios val, float peso1, float
peso2, float peso3)
```

Dado el identificador de un usuario, la red social a la cual pertenece dicho usuario, las valuaciones de recursos hechas por los usuarios de la red social y los pesos de los factores de similitud, devuelve una lista ordenada por grado de similitud de los identificadores de los amigos del primer nivel del usuario.

```
public ArrayList<String> recomendacionesPersonalizadasRecursos(String
idUsuario, int cantRecursosARecomendar, float umbralAceptacion,
PesosAtributosCalidad pesos, RedSocial red,
ValuacionesRecursosUsuarios val, float peso1, float peso2, float
peso3)
```

Dado el identificador del usuario al que se le quiere brindar las recomendaciones, un entero que representa la cantidad máxima de recursos recomendados, un real que indica el umbral de aceptación de un recurso, los pesos de los atributos de calidad de los recursos, la red social a la cual pertenece el usuario al que se le quiere brindar las recomendaciones, las valuaciones de recursos hechas por los usuarios de la red, y los ponderadores de los términos de similitud, la función devuelve una lista con los identificadores de los recursos recomendados utilizando el algoritmo descrito en el capítulo anterior.

4.4 Integración del algoritmo

Para realizar la integración del algoritmo en el sistema real de calificación y recomendación, se llevaron a cabo varias reuniones con los integrantes del proyecto LACCIR así como coordinación vía correo electrónico. Inicialmente, las reuniones se realizaron en la Facultad de Ingeniería de la UdelaR. En una primera instancia se acordó que debería proveer un conjunto de clases, un método principal que implemente el algoritmo de recomendación y un servicio web que invoque a dicho método. Luego que se avanzó en el diseño del algoritmo se viajó a La Plata para explicarles el funcionamiento del algoritmo y trabajar en la integración del mismo. Luego de varias reuniones diarias con los integrantes del proyecto LACCIR, en el Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA) de la Facultad de Informática de la Universidad Nacional de La Plata, se acordó que me encargaba de brindar las clases Java correspondientes y que ellos se encargaban de desarrollar el servicio web. Esto se debe a que había pocos días para realizar las primeras pruebas con el algoritmo funcionando según el calendario del proyecto LACCIR. Los integrantes del proyecto LACCIR también desarrollaron generadores de archivos xml necesarios para poder ejecutar el algoritmo.

Para poder obtener las recomendaciones personalizadas, se debe instanciar la clase *FriendSourcing* e invocar al método *algoritmoFriendSourcing* de la siguiente manera:

```
FriendSourcing f = new FriendSourcing();
ArrayList<String> recomendaciones =
f.algoritmoFriendSourcing(idUsuario, cantRecomendaciones,
rutaRedSocialXML, rutaValuacionesRecursosXML, rutaPesosAtributosXML);
```

donde:

- *idUsuario* es el string que identifica al usuario.
- *cantRecomendaciones* es el entero que representa la cantidad máxima de recursos recomendados.
- *rutaRedSocialXML* es la ruta absoluta donde se encuentra el archivo xml que contiene la red social.
- *rutaValuacionesRecursosXML* es la ruta absoluta donde se encuentra el archivo xml que contiene las valuaciones de los recursos.
- *rutaPesosAtributosXML* es la ruta absoluta donde se encuentra el archivo xml que contiene los pesos de los atributos de calidad.

La integración del algoritmo de recomendación en su proveedor de servicios se realizó de manera directa y sin inconvenientes.

Capítulo 5

Testeo del algoritmo y resultados obtenidos

En este capítulo se describen los casos de prueba generados para testear el algoritmo, se explica qué objetivos tiene cada prueba, qué resultados se obtuvieron y cómo se interpretan dichos resultados teniendo en cuenta los objetivos. Con estas pruebas se busca demostrar el esperado y correcto funcionamiento del algoritmo. Luego se calcula el orden de complejidad del algoritmo y finalmente se analizan los tiempos de ejecución.

5.1 Datos utilizados en las pruebas

Para la realización de las distintas pruebas se utilizaron los siguientes datos:

- *Cantidad de usuarios en la red social*: 50
- *Cantidad de recursos disponibles para valuar*: 10
- *Número promedio de conexiones por usuario*: 3.6
- *Número promedio de valuaciones por usuario*: 2.0

Las valuaciones de recursos se hicieron sobre los cuatro atributos de calidad: trustwhorthy (confiable), objective (objetivo), complete (completo) y well-written (bien escrito). En el Anexo se muestran los xml que contienen los datos utilizados para las pruebas.

La figura 6 muestra un grafo que representa la red social utilizada en las pruebas.

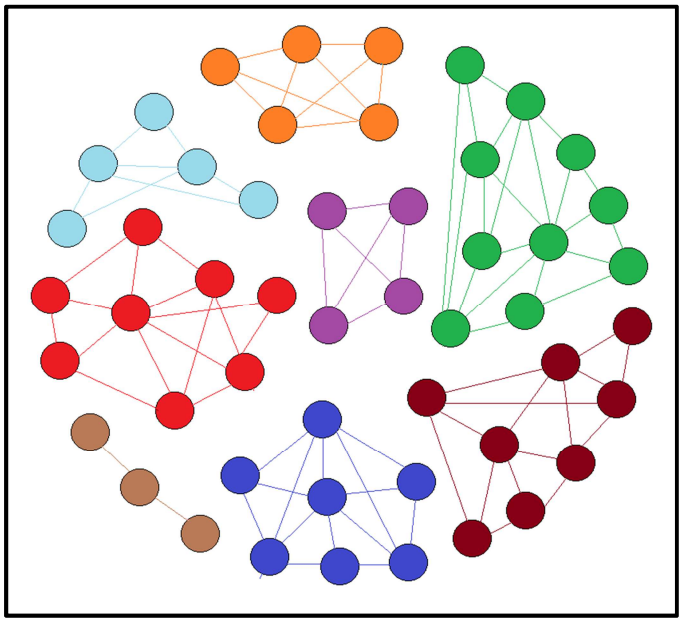

Figura 6: Representación de la red social

5.2 Evaluación de los resultados

Para la evaluación de la eficiencia de los resultados obtenidos por el algoritmo de recomendación se utilizan dos medidas comúnmente utilizadas en la recuperación de información[20]: *precision* y *recall*. *Precision* es la fracción de ítems recuperados que son relevantes[20]. *Recall* es la fracción de ítems relevantes que son recuperados[20].

$$Precision = \frac{\#(Items\ recuperados\ relevantes)}{\#(Items\ recuperados)}$$

$$Recall = \frac{\#(Items\ recuperados\ relevantes)}{\#(Items\ relevantes)}$$

Se considera que un *ítem recuperado* es aquel recurso recomendado por el algoritmo y que un *ítem relevante* es aquel recurso que fue valuado previamente por el usuario al que se le quiere brindar recomendaciones y cuya puntuación supera el umbral de aceptación definido.

5.3 Casos de prueba

A continuación se especifican los casos de prueba que testean cada una de las componentes del algoritmo. Considerar que para estas pruebas, el algoritmo recomienda a un usuario un recurso aunque éste haya sido valuado previamente por el usuario. Para cada caso de prueba se eligen cuatro usuarios aleatoriamente y se brindan las recomendaciones de recursos generadas por el algoritmo. Luego se calcula el promedio y la desviación estándar para las dos medidas de eficiencia mencionadas anteriormente.

5.3.1 Caso de prueba 1

Los parámetros utilizados en el caso de prueba 1 se detallan a continuación:

- *Peso en la similitud de las relaciones de amistad*: 1.0
- *Peso en la similitud de las valuaciones de recursos*: 0.0
- *Peso en la similitud de la actividad en las valuaciones*: 0.0
- *Peso del atributo de calidad trustwhorthy*: 0.25
- *Peso del atributo de calidad objective*: 0.25
- *Peso del atributo de calidad complete*: 0.25
- *Peso del atributo de calidad well-written*: 0.25
- *Umbral de aceptación de recursos*: 0.5
- *Cantidad de recursos recomendados*: 5

Con este caso de prueba se busca testear la componente del algoritmo que utiliza la similitud en las relaciones de amistad y analizar la eficiencia de los resultados obtenidos.

En la tabla 3 se muestran los resultados del primer caso de prueba.

	Precision	Recall
Promedio (Mean)	0.650	0.718
Desviación estándar (Standard deviation)	0.167	0.282

Tabla 3: Resultados del primer caso de prueba

Los resultados obtenidos son buenos, lo que indica que el factor de similitud en las relaciones de amistad es un factor a tener en cuenta a la hora de seleccionar usuarios similares. El algoritmo puede no brindar recomendaciones relevantes cuando existen pocos usuarios en la red o cuando existen pocos vínculos de amistad entre ellos.

5.3.2 Caso de prueba 2

Los parámetros utilizados en el caso de prueba 2 se detallan a continuación:

- *Peso en la similitud de las relaciones de amistad*: 0.0
- *Peso en la similitud de las valuaciones de recursos*: 1.0
- *Peso en la similitud de la actividad en las valuaciones*: 0.0
- *Peso del atributo de calidad trustwhorthy*: 0.25
- *Peso del atributo de calidad objective*: 0.25
- *Peso del atributo de calidad complete*: 0.25
- *Peso del atributo de calidad well-written*: 0.25
- *Umbral de aceptación de recursos*: 0.5
- *Cantidad de recursos recomendados*: 5

Con este caso de prueba se busca probar la componente del algoritmo que utiliza la similitud en las valuaciones de recursos y analizar la eficiencia de los resultados obtenidos.

En la tabla 4 se muestran los resultados del segundo caso de prueba.

	Precision	Recall
Promedio (Mean)	0.700	0.719
Desviación estándar (Standard deviation)	0.224	0.180

Tabla 4: Resultados del segundo caso de prueba

Los resultados obtenidos son apenas mejores que los anteriores pero siguen siendo buenos resultados. Esto implica que las valuaciones de usuarios similares son útiles para poder seleccionar recursos de interés para el usuario activo. El algoritmo puede no brindar recomendaciones relevantes cuando existen pocas valuaciones de recursos, lo cual afecta al intentar obtener conocimiento acerca del comportamiento de un usuario de la red.

5.3.3 Caso de prueba 3

Los parámetros utilizados en el caso de prueba 3 se detallan a continuación:

- *Peso en la similitud de las relaciones de amistad*: 0.0
- *Peso en la similitud de las valuaciones de recursos*: 0.0
- *Peso en la similitud de la actividad en las valuaciones*: 1.0
- *Peso del atributo de calidad trustwhorthy*: 0.25
- *Peso del atributo de calidad objective*: 0.25
- *Peso del atributo de calidad complete*: 0.25
- *Peso del atributo de calidad well-written*: 0.25
- *Umbral de aceptación de recursos*: 0.5
- *Cantidad de recursos recomendados*: 5

Con este caso de prueba se busca probar la componente del algoritmo que utiliza la actividad en las valuaciones de recursos y analizar la eficiencia de los resultados obtenidos.

En la tabla 5 se muestran los resultados del tercer caso de prueba.

	Precision	**Recall**
Promedio (Mean)	0.750	0.782
Desviación estándar (Standard deviation)	0.219	0.219

Tabla 5: Resultados del tercer caso de prueba

Los resultados obtenidos son mejores que los obtenidos con las componentes anteriores. Es relevante el hecho de que un amigo de la red haya valuado previamente un largo conjunto de recursos, esto brinda valuaciones más confiables ya que se dispone de un conocimiento más amplio que otros usuarios que hayan puntuado menos recursos. El algoritmo puede no brindar recomendaciones relevantes cuando existe poca actividad en las valuaciones de recursos.

5.4 Resultados obtenidos en el sistema real

Los integrantes del proyecto LACCIR también realizaron pruebas para testear el comportamiento del algoritmo. A los usuarios de la red social les asignaron recursos para valuar. En la tabla 6 se muestran los resultados obtenidos.

Tamaño del conjunto de recursos recomendados	**Precision promedio**	**Recall promedio**
2	0,041	0,082
5	0,037	0,184
10	0,034	0,336

Tabla 6: Resultados obtenidos en el sistema real

Como resultado de las pruebas se observó que el algoritmo realiza recomendaciones de relevancia pero en algunos casos no devolvió resultados de interés, esto se debe principalmente a que el algoritmo busca usuarios similares en base a las valuaciones de recursos y a que se asignó a cada usuario una serie de recursos para valuar. Una manera de mejorar los resultados cuando existen pocas valuaciones de recursos, es utilizar al inicio un tipo de crowdsourcing que considere las valuaciones de recursos hechas por todos los usuarios, se puede tomar por ejemplo, el promedio de dichas valuaciones. También se pueden utilizar los datos del perfil de usuario, como por ejemplo, información demográfica del usuario como sexo, edad, nacionalidad, etc.

5.5 Análisis e interpretación de los resultados de las pruebas

En base a los resultados de las pruebas se observa que el factor de similitud en las relaciones de amistad es un factor a tener en cuenta a la hora de seleccionar usuarios similares. Las valuaciones de usuarios similares también son útiles para poder seleccionar recursos de interés para el usuario activo y es relevante el hecho de que un amigo de la red haya valuado previamente un largo conjunto de recursos, lo que permite brindar valuaciones más confiables ya que se dispone de un conocimiento más amplio que otros usuarios que hayan puntuado menos recursos. El algoritmo funciona de manera correcta y brinda recomendaciones personalizadas de relevancia. Los mejores resultados fueron obtenidos teniendo en cuenta información en la actividad de las valuaciones de recursos de otros usuarios similares (precision = 75 %, recall = 78 %). Por otro lado, en un corto tiempo de funcionamiento del sistema donde existen pocos usuarios, pocas relaciones de amistad y pocas valuaciones (cold-start problem), el algoritmo de recomendación puede brindar resultados que no sean relevantes para el usuario. Para contrarrestar este problema se puede utilizar inicialmente crowdsourcing tomando como fuente de información el promedio de las valuaciones de todos los usuarios de la red social.

5.6 Complejidad y eficiencia

En esta sección se calcula primeramente el orden de complejidad del algoritmo de recomendación y luego se analizan los tiempos de ejecución.

Se considera la siguiente nomenclatura para dicho cálculo:

- u_{active} es el usuario activo, es decir al usuario al que se le quiere brindar las recomendaciones.

- N es la cantidad de amigos del primer nivel del usuario activo, es decir, los amigos directos.

- M es la cantidad promedio de recursos que valuó un amigo del usuario activo.

En primera instancia, el algoritmo de recomendación obtiene los N amigos más similares al usuario activo u_{active}, en base a las tres similitudes descriptas en los capítulos anteriores. Luego para cada uno de estos amigos, se recorren los M recursos valuados por dicho amigo en busca del recurso con mejor puntuación en base a las valuaciones hechas y a los pesos de los atributos de calidad definidos inicialmente. Por lo tanto, el orden de complejidad del algoritmo de recomendación es $\Theta(N * M)$.

Con respecto a la rapidez de los resultados, se calculó el tiempo de ejecución del algoritmo para un usuario elegido aleatoriamente de cada caso de prueba descrito en la sección anterior y luego se calculó el promedio de dichos tiempos de ejecución. En la tabla 7 se muestran los tiempos de ejecución obtenidos.

	Segundos
Tiempo de ejecución (Caso de prueba 1)	0.016
Tiempo de ejecución (Caso de prueba 2)	0.015
Tiempo de ejecución (Caso de prueba 3)	0.014
Tiempo de ejecución promedio = $\overline{t_{ejec}}$	0.015

Tabla 7: Tiempos de ejecución del algoritmo

El tiempo de ejecución promedio es de 0.015 segundos para una red social con 50 usuarios y 10 recursos disponibles para valuar, lo que se considera un tiempo de ejecución promedio aceptable ($\overline{t_{ejec}}$ << 1 segundo).

Capítulo 6

Conclusiones y trabajo a futuro

En este capítulo se presentan las conclusiones del proyecto y finalmente se mencionan posibles cambios y mejoras al algoritmo de recomendación como trabajo a futuro.

En general, los algoritmos de recomendación utilizados en los sitios web no son de público conocimiento debido al impacto que tienen en ciertas actividades desarrolladas en la web, como por ejemplo, el comercio electrónico. Por lo tanto, el conocimiento de dichos algoritmos es muy valioso. Por otro lado, en este proyecto existía el desafío de diseñar un algoritmo de recomendación para un sistema de calificación y recomendación que estaba en plena etapa de diseño y posterior desarrollo. A pesar de todos estos factores, se cumplió con el objetivo de diseñar y desarrollar un algoritmo de recomendación que cumpla con todos los requerimientos establecidos en las distintas reuniones con los integrantes del proyecto LACCIR, se pudo integrar sin inconvenientes a un sistema real de recomendaciones y también se llevaron a cabo pruebas reales. En las pruebas realizadas para testear el algoritmo se obtuvieron resultados eficientes (precision = 75 %, recall = 78 %), mostrando que la información contenida en las redes sociales es sumamente importante para un sistema de recomendación.

A partir de los resultados obtenidos en el sistema real se propuso la idea de desarrollar una variante del algoritmo que puede mejorar la eficiencia de los resultados brindados por el algoritmo, el cual obtiene las mismas entradas que el algoritmo original pero en vez de considerar el factor de similitud en las valuaciones de recursos, tiene que tener en cuenta los artículos puntuados en común, que es una muestra de interés, sin importar cómo lo han valuado. En vez de asignar recursos a los usuarios, éstos elegirían aquellos que sean de su interés para valuar. Luego de obtener una lista ordenada con los usuarios más similares, se obtiene de cada usuario similar, los recursos a recomendar en base a las valuaciones de recursos hechas por el usuario similar. El desarrollo de esta variante del algoritmo se deja como trabajo a futuro. También se puede incorporar en futuras versiones del algoritmo, información del perfil de usuario, como por ejemplo información demográfica, para encontrar similitud entre usuarios.

Anexo

Atributos.xml

```
<ListaAtributos>
    <Atributo>
        <Nombre>Trustwhorthy</Nombre>
        <Peso>0.25</Peso>
    </Atributo>
    <Atributo>
        <Nombre>Objective</Nombre>
        <Peso>0.25</Peso>
    </Atributo>
    <Atributo>
        <Nombre>Well-written</Nombre>
        <Peso>0.25</Peso>
    </Atributo>
    <Atributo>
        <Nombre>Complete</Nombre>
        <Peso>0.25</Peso>
    </Atributo>
</ListaAtributos>
```

Redsocial.xml (fragmento)

```
<RedSocial>
    <Usuario>
        <PerfilUsuario>
            <Identificador>verde1</Identificador>
            <ListaAtributos>
                <Atributo>Grado</Atributo>
            </ListaAtributos>
            <ListaAmigos>
                <Amigo>
                    <Identificador>verde2</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde3</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde8</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
            </ListaAmigos>
        </PerfilUsuario>
    </Usuario>
    <Usuario>
        <PerfilUsuario>
            <Identificador>verde2</Identificador>
            <ListaAtributos>
                <Atributo>Grado</Atributo>
            </ListaAtributos>
```

```
            <ListaAmigos>
                  <Amigo>
                        <Identificador>verde1</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde3</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde4</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde6</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde7</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
            </ListaAmigos>
      </PerfilUsuario>
</Usuario>
<Usuario>
      <PerfilUsuario>
            <Identificador>verde3</Identificador>
            <ListaAtributos>
                  <Atributo>Grado</Atributo>
            </ListaAtributos>
            <ListaAmigos>
                  <Amigo>
                        <Identificador>verde1</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde2</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde6</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde7</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde8</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
            </ListaAmigos>
      </PerfilUsuario>
</Usuario>
<Usuario>
      <PerfilUsuario>
            <Identificador>verde4</Identificador>
```

```
            <ListaAtributos>
                  <Atributo>Grado</Atributo>
            </ListaAtributos>
            <ListaAmigos>
                  <Amigo>
                        <Identificador>verde2</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde5</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde6</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
            </ListaAmigos>
      </PerfilUsuario>
</Usuario>
<Usuario>
      <PerfilUsuario>
            <Identificador>verde5</Identificador>
            <ListaAtributos>
                  <Atributo>Grado</Atributo>
            </ListaAtributos>
            <ListaAmigos>
                  <Amigo>
                        <Identificador>verde4</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde6</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde10</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
            </ListaAmigos>
      </PerfilUsuario>
</Usuario>
<Usuario>
      <PerfilUsuario>
            <Identificador>verde6</Identificador>
            <ListaAtributos>
                  <Atributo>Grado</Atributo>
            </ListaAtributos>
            <ListaAmigos>
                  <Amigo>
                        <Identificador>verde2</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
                        <Identificador>verde3</Identificador>
                        <TipoVinculo>Academico</TipoVinculo>
                  </Amigo>
                  <Amigo>
```

```
                    <Identificador>verde4</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde5</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde7</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde8</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde9</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
                <Amigo>
                    <Identificador>verde10</Identificador>
                    <TipoVinculo>Academico</TipoVinculo>
                </Amigo>
            </ListaAmigos>
        </PerfilUsuario>
    </Usuario>
```

Valuaciones.xml (fragmento)

```
<EvaluacionesRecursos>
    <Usuario>
        <Identificador>verde1</Identificador>
        <ListaEvaluaciones>
            <Recurso>
                <Identificador>articulo1</Identificador>
                <Valuacion>
                    <Atributo>Trustwhorthy</Atributo>
                    <Puntaje>9</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Objective</Atributo>
                    <Puntaje>3</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Well-written</Atributo>
                    <Puntaje>6</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Complete</Atributo>
                    <Puntaje>4</Puntaje>
                </Valuacion>
            </Recurso>
            <Recurso>
                <Identificador>articulo2</Identificador>
                <Valuacion>
                    <Atributo>Trustwhorthy</Atributo>
```

```
            <Puntaje>3</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Objective</Atributo>
            <Puntaje>5</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Well-written</Atributo>
            <Puntaje>7</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Complete</Atributo>
            <Puntaje>3</Puntaje>
        </Valuacion>
    </Recurso>
    <Recurso>
        <Identificador>articulo3</Identificador>
        <Valuacion>
            <Atributo>Trustwhorthy</Atributo>
            <Puntaje>9</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Objective</Atributo>
            <Puntaje>2</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Well-written</Atributo>
            <Puntaje>5</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Complete</Atributo>
            <Puntaje>2</Puntaje>
        </Valuacion>
    </Recurso>
    <Recurso>
        <Identificador>articulo4</Identificador>
        <Valuacion>
            <Atributo>Trustwhorthy</Atributo>
            <Puntaje>8</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Objective</Atributo>
            <Puntaje>6</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Well-written</Atributo>
            <Puntaje>6</Puntaje>
        </Valuacion>
        <Valuacion>
            <Atributo>Complete</Atributo>
            <Puntaje>8</Puntaje>
        </Valuacion>
    </Recurso>
    <Recurso>
        <Identificador>articulo5</Identificador>
        <Valuacion>
            <Atributo>Trustwhorthy</Atributo>
```

```
                <Puntaje>7</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Objective</Atributo>
                <Puntaje>0</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Well-written</Atributo>
                <Puntaje>6</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Complete</Atributo>
                <Puntaje>7</Puntaje>
        </Valuacion>
</Recurso>
<Recurso>
        <Identificador>articulo6</Identificador>
        <Valuacion>
                <Atributo>Trustwhorthy</Atributo>
                <Puntaje>5</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Objective</Atributo>
                <Puntaje>4</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Well-written</Atributo>
                <Puntaje>9</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Complete</Atributo>
                <Puntaje>3</Puntaje>
        </Valuacion>
</Recurso>
<Recurso>
        <Identificador>articulo7</Identificador>
        <Valuacion>
                <Atributo>Trustwhorthy</Atributo>
                <Puntaje>5</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Objective</Atributo>
                <Puntaje>1</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Well-written</Atributo>
                <Puntaje>9</Puntaje>
        </Valuacion>
        <Valuacion>
                <Atributo>Complete</Atributo>
                <Puntaje>2</Puntaje>
        </Valuacion>
</Recurso>
<Recurso>
        <Identificador>articulo8</Identificador>
        <Valuacion>
                <Atributo>Trustwhorthy</Atributo>
```

```
                    <Puntaje>3</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Objective</Atributo>
                    <Puntaje>3</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Well-written</Atributo>
                    <Puntaje>9</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Complete</Atributo>
                    <Puntaje>3</Puntaje>
                </Valuacion>
            </Recurso>
            <Recurso>
                <Identificador>articulo9</Identificador>
                <Valuacion>
                    <Atributo>Trustwhorthy</Atributo>
                    <Puntaje>1</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Objective</Atributo>
                    <Puntaje>4</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Well-written</Atributo>
                    <Puntaje>9</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Complete</Atributo>
                    <Puntaje>0</Puntaje>
                </Valuacion>
            </Recurso>
            <Recurso>
                <Identificador>articulo10</Identificador>
                <Valuacion>
                    <Atributo>Trustwhorthy</Atributo>
                    <Puntaje>2</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Objective</Atributo>
                    <Puntaje>4</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Well-written</Atributo>
                    <Puntaje>2</Puntaje>
                </Valuacion>
                <Valuacion>
                    <Atributo>Complete</Atributo>
                    <Puntaje>3</Puntaje>
                </Valuacion>
            </Recurso>
        </ListaEvaluaciones>
</Usuario>
```

Bibliografía

[1] A. Díaz, R. Motz, A. Fernández, J. Valdeni de Lima, D. López. *Quality Web Information Retrieval: Towards Improving Semantic Recommender Systems with Friendsourcing*. VI Congreso Ibero-americano de Telemática (2011).

[2] M. Bernstein, D. Tan, G. Smith, M. Czerwinski, E. Horvitz. *Personalization via Friendsourcing*. ACM Transactions on Computer-Human Interaction (2010).

[3] C. Basu, H. Hirsh, W. Cohen. *Recommendation as Classification: Using Social and Content-Based Information in Recommendation*. Proceedings of the 15th national/10th conference on Artificial intelligence/innovative applications of artificial intelligence (1998).

[4] *Latin American and Caribbean Collaborative ICT Research - Quality Health Information Retrieval*. Accesible en: http://www.laccir.org/web/#/Projects/HealthCare/InformationRetrieval Último acceso: 12/03/2013.

[5] *Universidad del Cauca - Red Social Unisalud*. Accesible en: http://esalud.unicauca.edu.co/redunisalud. Último acceso: 12/03/2013.

[6] 5th ACM Conference on Recommender Systems (2011). Accesible en: http://recsys.acm.org/2011/index.shtml. Último acceso: 12/03/2013.

[7] E. Peis, J. M. Morales del Castillo, J. A. Delgado López. *Sistemas de Recomendación Semánticos: Un análisis del estado de la cuestión*. Hipertext.net (2008).

[8] B. Sarwar, G. Karypis, J. Konstan, J. Riedl. *Item-Based Collaborative Filtering Recommendation Algorithms*. Proceedings of the 10th international conference on World Wide Web (2001).

[9] J. Ben Schafer, J. Konstan, J. Rield. *Recommender Systems in E-Commerce*. Proceedings of the 1st ACM conference on Electronic commerce (1999).

[10] D. Tunkelang. *Recommendations as a Conversation with the User*. 5th ACM Conference on Recommender Systems (2011).

[11] R. Burke. *Hybrid Recommender Systems: Survey and Experiments*. User Modeling and User-Adapted Interaction (2002).

[12] R. Sinha, K. Swearingen. The Role of Transparency in Recommender Systems. ACM Press (2002).

[13] A. I. Schein, A. Popescul, L. H. Ungar, D. M. Pennock. Methods and Metrics for Cold-Start Recommendations. ACM Press (2002).

[14] O. Serrat. *Social Network Analysis*. Knowldege Solutions (2009).

[15] *Internet World Stats - Internet Growth Statistics*. Accesible en: http://www.internetworldstats.com/emarketing.htm#stats. Último acceso: 12/03/2013.

[16] M. Ferner. *Beyond Facebook: 74 Popular Social Networks Worldwide*. Accesible en: http://www.practicalecommerce.com/articles/2701-Beyond-Facebook-74-Popular-Social-Networks-Worldwide. Último acceso: 12/03/2013.

[17] *Facebook - Timeline*. Accesible en: http://www.facebook.com/press/info.php?timeline. Último acceso: 14/09/2011.

[18] M. Wilkowski. *Friendsourcing on Twitter*. Accesible en: http://historiaimedia.org/2009/09/30/friendsourcing-on-twitter-for-academic-purposes. Último acceso: 12/03/2013.

[19] L. Mendoza, J. I. Vidal, A. Díaz, A. Fernández, R. Motz. *Plataforma I+D en sistemas de calificación y recomendación: arquitectura de referencia*. Proceedings of VI Congresso Ibero-americano de Telemática (2011).

[20] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press (2008). Accesible en: http://nlp.stanford.edu/IR-book/. Último acceso: 12/03/2013.