ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Análisis de Métricas de Similaridad Usadas en un Algoritmo de Filtro Colaborativo Basado en el Usuario Para Recomendar Materias de Pregrado”

INFORME DE

PROYECTO DE GRADUACIÓN

Previo la obtención de los Títulos de:

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

INGENIERO EN TELEMÁTICA

Presentada por:

JOHANNA MELISSA DEL PINO MANOBANDA

GUSTAVO DANIEL SALAZAR LOOR

GUAYAQUIL – ECUADOR
2011

DEDICATORIA

Dedico este trabajo a mis padres por brindarme su apoyo incondicional y ser un pilar importante en mi vida, a mi hermano como un ejemplo de las cosas por las que vale la pena luchar, y a Juan Carlos por recordarme cada día que puedo lograr lo que me propongo si tengo las fuerzas para ir tras ello.

Johanna Melissa Del Pino Manobanda

A mis amigos, en especial a los miembros del grupo TAWS.

Gustavo Daniel Salazar Loor

AGRADECIMIENTO

Agradecemos a Dios por darnos fuerzas en el camino para alcanzar esta meta. A nuestros padres por apoyar nuestros sueños y creer en nosotros, y a las personas que de alguna manera nos ayudaron a caminar hacia esta nueva etapa de nuestra vida.

DECLARATORIA EXPRESA

“La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la **Escuela Superior Politécnica del Litoral**”

(Reglamento de Graduación de la ESPOL)

_____

JOHANNA MELISSA DEL PINO MANOBANDA

_____

GUSTAVO DANIEL SALAZAR LOOR

TRIBUNAL DE SUSTENTACIÓN

Ing. Jorge Aragundi.

SUB-DECANO FIEC

Ing. Vanessa Cedeño

DIRECTOR DE PROYECTO

Ing. Guido Caicedo.

MIEMBRO PRINCIPAL

RESUMEN

Los sistemas de recomendación son ampliamente utilizados hoy en día gracias a su capacidad de analizar las preferencias de usuarios y sugerir ítems. En el ámbito educativo son aplicados principalmente en la recomendación de objetos de aprendizaje y en la guía de decisión de cursos que deben tomar los estudiantes.

El uso de los recomendadores, en su mayoría, no toma en cuenta el tiempo en el que los usuarios mostraron preferencias por ítems. Consideramos el tiempo como una variable importante para la aplicación de un sistema de recomendación académico donde sea primordial el estudio de las decisiones del estudiante a lo largo de su carrera. Por ejemplo, el semestre donde un estudiante toma una materia.

Este trabajo realiza una adaptación de un recomendador de materias de filtro colaborativo basado en el usuario para lograr realizar recomendaciones basadas en el historial académico de los usuarios para que también se tome en consideración el factor tiempo.

Presentando una nueva manera de verificar similitud entre dos estudiantes, en donde no solo se debe tomar en cuenta la preferencia de un ítem, en este

caso la aprobación de una materia, sino el orden y momento en que se realizó la aprobación para así lograr comparar tendencias completas en la manera de aprobar el flujo de una carrera entre los estudiantes.

Adicionalmente, se expone la medición de la efectividad de dos métricas de similaridad en el sistema de recomendación de materias para estudiantes de la carrera de Ingeniería en Telecomunicaciones ofertada por la FIEC - ESPOL.

Con las pruebas realizadas se pretende determinar cuál es la métrica de similaridad idónea para la adaptación del recomendador basado en el usuario para así obtener recomendaciones más efectivas de materias.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

ABREVIATURAS

URL	Localizador de Recurso Universal

INTRODUCCIÓN

Durante los últimos años, los sistemas de recomendación han sido herramientas que han contribuido a mejorar la experiencia de los usuarios en las aplicaciones que estos utilizan. Numerosas empresas han apostado por el uso de estas herramientas para lograr que sus sitios web posean un ambiente personalizado, en donde el contenido que se les presenta a sus usuarios está estrechamente atado a sus preferencias personales. Algunos de los sitios web en donde se puede encontrar el uso de los recomendadores son Amazon, Netflix, Facebook y Twitter. Los cuales sugieren compras de productos, películas, amistades y personas a seguir, respectivamente. La gran cantidad de información filtrada de algunos usuarios sirve para ayudar a otros, resultando en un entorno colaborativo.

La aplicación de los sistemas de recomendación en el plano comercial es muy conocido, sin embargo su uso en otras áreas como la educativa ha despertado el interés de investigadores que ven en las instituciones educativas una gran fuente de información apta para realizar recomendaciones.

La precisión de los sistemas de recomendación basados en filtro colaborativo, específicamente los basados en el usuario en gran parte se

debe al cálculo que mide la similaridad entre usuarios, por ello se realiza un análisis de dos de ellas en el caso de estudio presentado.

El contenido de este trabajo se distribuye de la siguiente manera: en el capítulo 1 se describe el problema, antecedentes, objetivos, justificación y alcance del presente trabajo. En el capítulo 2 se presentan los fundamentos teóricos de los sistemas de recomendación detallando conceptos de los basados en filtro colaborativo, con mayor énfasis en los basados en el usuario. En el capítulo 3 se describe el análisis de la solución. En el capítulo 4 se muestra el diseño y se detalla la implementación de la solución presentada. Finalmente, las pruebas y resultados son mostrados en el capítulo 5.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN.

1.1. Antecedentes y Descripción del Problema

Antecedentes

En el ámbito educativo los sistemas de recomendaciones han sido ampliamente usados para el beneficio de estudiantes de escuelas secundarias y universidades. Entre las aplicaciones más notorias están la recomendación de objetos de aprendizaje y asesoría académica en la recomendación de materias o cursos de manera personalizada. Una recomendación personalizada requiere de un conocimiento amplio de la información académica del estudiante por parte de los asesores académicos, por ello se han implementado sistemas que pretenden cumplir las mismas funciones de las personas que realizan dicho trabajo. Asimismo, estos sistemas pueden ayudar a

los asesores a tomar una mejor decisión antes de que realicen recomendaciones a sus estudiantes.

Haciendo las veces de un asesor académico, AARCON [1] es un sistema de recomendación basado en casos [2] que recomienda los cursos que debe tomar un estudiante de la Universidad De Paul basándose en el programa académico de la institución, el historial de cursos tomados por el estudiante y la información de otros estudiantes con un historial similar. Una función semejante cumple RARE [3], un sistema de recomendación de cursos para estudiantes de maestría que se basa en reglas de asociación para inferir los cursos que recomendará, incorpora minería de datos y permite a los estudiantes calificar las recomendaciones que se hacen a partir de la experiencia de estudiantes que han finalizado sus estudios.

Además de cumplir funciones de asesores académicos algunos sistemas usan información directa de los mismos asesores, Course Agent, es un sistema implementado en la Escuela de las Ciencias de la Información de la Universidad de Pittsburgh [4], toma esta información junto con evaluaciones de los estudiantes sobre los cursos que han aprobado, la relevancia de que ellos creen que tiene la

materia para sus estudios y la información de un perfil definido por sus objetivos de carrera para recomendarles otros cursos de interés.

Algunos sistemas combinan distintas técnicas de recomendación, como sucede en la Universidad de Dublín, comparando historiales de materias electivas tomadas de sus estudiantes aplican filtros colaborativos y tomando palabras que describen dichas materias aplican un sistema de recomendación basado en contenido. Un estudio similar realizan Castellano y Martínez usando filtros colaborativos para recomendar materias de especialización y electivas en el sistema de educación español de escuelas secundarias [5].

Descripción del Problema

De acuerdo a los proyectos citados en la sección anterior vemos que es posible aplicar los filtros colaborativos para recomendar materias basadas en la información académica que una universidad posee de sus estudiantes. Recomendar todas las materias que debe tomar un estudiante en un término académico no es posible en todas las instituciones educativas porque el flujo de materias de la carrera es fijo y se cumple de forma obligatoria cada semestre. Para el caso de universidades donde la elección de materias se realiza con mayor grado de libertad, dos estudiantes de una carrera no necesariamente

se registrarán en las mismas materias a pesar de cursar el mismo semestre.

Este escenario permite plantear la construcción de un sistema de recomendación que asista a los estudiantes en los procesos de registro a lo largo de su carrera. En donde las recomendaciones se realicen se basen en las aprobaciones que se han dado en una determinada carrera.

De esta manera las elecciones exitosas de registros o dicho de otra manera, las aprobaciones de las materias de estudiantes avanzados pueden servir como experiencias “prestadas” para estudiantes nuevos o menos avanzados en su carrera, al momento de realizar la elección de las materias a registrarse en un semestre.

Dado que lo que se busca es dar recomendaciones para cualquier semestre de la carrera de un estudiante. Al momento de usar las experiencias de otros estudiantes de la carrera, es fundamental analizar el orden en que sus materias fueron aprobadas, de esta manera, las sugerencias para el estudiante que se quiere asistir son personalizadas y basadas en usuarios que tengan su misma tendencia en el orden de aprobación del flujo.

Esta última característica no se encuentra presente en los antecedentes mencionados, ya que la misión de estos sistemas es recomendar cursos optativos, en su escenario el momento de tomar el curso es irrelevante, y el problema se centra en escoger de una lista llena de opciones que curso contribuye mejor a la formación académica en general

1.2. Justificación

Un recomendador de materias en la ESPOL constituye una herramienta de gran interés, ya que en la actualidad no existe un método de recomendación, automático o guiado por humanos, con el que el estudiante cuente previo a un registro, resulta interesante tomar las experiencias de los estudiantes avanzados para que sean aprovechadas por los nuevos.

El presente trabajo nace como una iniciativa que busca explorar la viabilidad de la aplicación de los sistemas de recomendación en nuestra universidad, y hallar un escenario para la implementación de este tipo de sistemas dentro de ella. De esta manera construir un trabajo que sirva de base a futuros proyectos dentro de ESPOL.

Además se desea realizar un análisis del comportamiento de un recomendador en el área de materias de pregrado, en un escenario donde el orden de las preferencias debe ser tomado en cuenta. También se aspira analizar las métricas de similaridad como factores que influyen directamente en la precisión de un recomendador.

1.3. Objetivos

- Diseñar e implementar un algoritmo de recomendación de materias a estudiantes, considerando la historia académica de los mismos.
- Determinar una métrica de similaridad idónea, para el caso de estudio presentado.
- Implementar un nuevo tipo de recomendador en una librería de aprendizaje automático ya existente.
- Implementar una interfaz de presentación de resultados.

1.4. Alcance

En este trabajo se realiza la implementación de un recomendador de materias basado en el perfil académico de los estudiantes, que puede ser ajustado a cualquier carrera de la ESPOL. El recomendador de materias debe formar parte de una librería de aprendizaje inteligente

ya existente. Además debe poder servir de base a futuras implementaciones de aplicaciones de recomendación para los estudiantes de la ESPOL, así como también a la construcción de recomendadores de materias más complejos en donde intervengan otros factores de criterio de elección de materias como: las notas de los estudiantes, valoraciones de los profesores en el CENACAD, etc.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS DE LOS SISTEMAS DE RECOMENDACIÓN.

2.1. Sistemas de Recomendación

Los sistemas de recomendación son una de las técnicas más conocidas de aprendizaje automático y un tipo especial de filtrado de información [6]. Estos sistemas se usan para inferir las preferencias de los usuarios con la finalidad de recomendarles ítems que aún no conocen. Un ítem es cualquier entidad que es objeto de recomendación en el dominio en el que aplicamos el sistema, sean estos libros, películas, documentos, incluso personas.

Fuentes de Información

Para predecir las preferencias de los usuarios los sistemas de recomendación se valen de varias fuentes de información tales como los atributos de los usuarios, los atributos y contenido de los ítems, y la interacción de un usuario con un ítem [6].

La interacción de un usuario con un ítem es la acción que realiza para demostrar interés en él, por ejemplo, cuando un usuario califica un video con un número de estrellas, cuando realiza la compra de un artículo en línea, cuando le da clic al enlace de una foto. Con estas acciones el usuario está demostrando preferencia por el ítem y de cierta manera lo está evaluando.

- Los atributos del usuario tales como ubicación geográfica, edad, sexo, entre otros, son los datos que lo caracterizan y definen un perfil.
- Los atributos como el color, peso, marca, entre otros, caracterizan a los ítems y nos permiten asociarlos con otros que tengan los mismos atributos.
- El contenido de los ítems está muy ligado al de documentos de texto, cuyos atributos son las palabras que en él se hallan y sus valores se determinan con mediciones basadas en la frecuencia con que se repiten.

Procesamiento de la información

Dependiendo de la fuente de información existen diversas técnicas que se aplican para procesarlas y realizar las recomendaciones, distinguiéndose tres clases principales: Filtro colaborativo, basados en contenido, e híbridos.

Las evaluaciones resultantes de la interacción usuario-ítem y la información de las demás fuentes, luego de ser procesadas son mostradas o filtradas previamente para presentarlas como recomendaciones.

2.1.1. Sistemas de Recomendación Basados en Filtro Colaborativo

El proceso de filtrar información usando técnicas que involucran la colaboración de varios agentes es aplicado en los sistemas de recomendaciones para inferir gustos a partir de patrones.

En un ambiente en el que existe una gran cantidad de usuarios, cada uno es una fuente de información que puede ser representada por el conjunto de sus preferencias. Al usar la información de todos para determinar patrones de preferencias se

forma un entorno colaborativo, que servirá para inferir, en base a esos patrones las preferencias de otros usuarios.

Una de las principales características de estos sistemas de recomendaciones es que la única fuente de información necesaria proveniente del usuario y del ítem es la existencia de la interacción entre ambos, independientemente de los atributos o propiedades que los caractericen [7].

Al no depender de atributos particulares de los usuarios ni de los ítems los sistemas de recomendaciones basados en filtro colaborativo pueden ser usados en diversos dominios, es decir su aplicación se puede dar en distintas áreas, el mismo procesamiento de información puede ser aplicado en la recomendación de diversos tipos de ítems, sean estos películas, libros, electrodomésticos, artículos electrónicos, personas.

Formas de Interacción

El usuario tiene diversas formas de interactuar con un ítem y cada una de ellas tiene su representación. Una de las formas de interacción más conocidas y usadas es el *rating* o votación valorada, que se representa con un valor numérico que el usuario

elige dentro de un rango discreto para asignárselo a un ítem y denotar un grado de preferencia por él.

Actualmente las redes sociales más populares como Facebook y Google+ optan por una votación distinta, una forma de interacción con representación binaria o booleana, donde se representa con 1 o verdadero el hecho que a un usuario le guste un ítem, y 0 o falso si no. Un ejemplo común de esta forma de interacción se da cuando se presenta un botón al usuario para que interactúe y le permitan expresar su preferencia por el contenido mostrado. La Figura 2-1 muestra los botones usados por las redes sociales mencionadas.

Figura 2-1 – Botones de Facebook y Google + para denotar preferencia por un ítem

Las aplicaciones que no poseen una interfaz que le permita al usuario evaluar directamente un artículo, interpretan otras interacciones como fuente de información útil, por ejemplo el tiempo en segundos que el usuario permanece viendo el ítem.

La mención de URL's, el número de visitas a una página web, dar clic en un enlace, son otras representaciones de interacción con un ítem en aplicaciones web.

Evaluación explícita e implícita

Con algunas formas de interacción se denota de manera explícita la preferencia hacia un ítem. El usuario expresa clara y determinadamente su interés por él, por ejemplo cuando pulsa un botón que indica que le gusta o asignándole un valor en un rango numérico. Normalmente en los sitios web las evaluaciones explícitas están ligadas a una interfaz gráfica que lo permite.

Cuando no existe un medio que permita a los usuarios denotar interés explícitamente por un ítem, varias acciones del usuario son analizadas y algunas de ellas, como permanecer una cantidad de tiempo en una página web o dar clic en un enlace o ver un artículo repetidas veces, son usadas para extraer información de preferencias de forma implícita.

Al elegir cómo se obtendrá la información sobre las preferencias del usuario en sistema de recomendaciones se debe tomar en cuenta la frecuencia con la que los usuarios cambian sus

preferencias, factor que se relaciona directamente con la frecuencia con la que un usuario evaluará un ítem. Para algunos usuarios evaluar en reiteradas ocasiones no es de su agrado por lo que obtener evaluaciones implícitas sería la decisión adecuada. La Figura 2-2 muestra un esquema de la interacción entre el usuario y el ítem

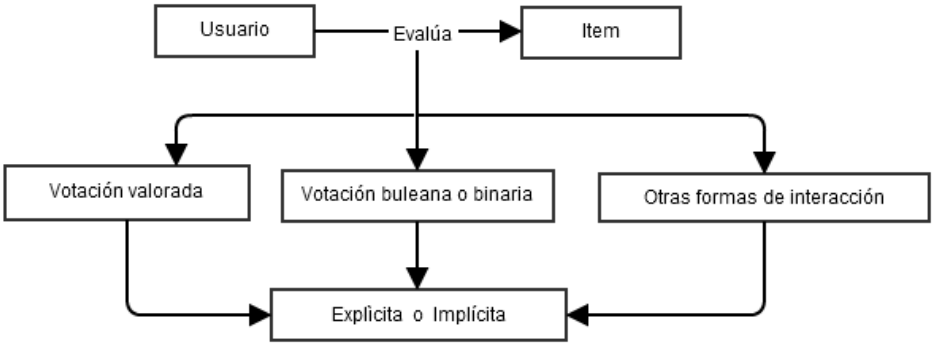


Figura 2-2 – Esquema de interacción entre usuario e ítem

Representación de las evaluaciones

El conjunto de datos obtenido por la interacción de los usuarios puede ser visto como una matriz donde las filas corresponden a los todos usuarios y las columnas a todos los ítems.

Para cada intersección entre usuario e ítem se asigna el valor de la evaluación obtenida por la forma de interacción. Si es votación valorada se asigna el valor numérico. La Tabla I muestra un

ejemplo de la representación de votación valorada visto como una matriz.

Usuarios/Ítems	Ítem 1	Ítem 2	Ítem 3	Ítem 4
Usuario 1	7	-	5	2
Usuario 2	3	3	-	3
Usuario 3	8	4	6	1

Tabla I – Matriz de preferencias de Usuarios vs Ítems usando votación valorada

Si es votación binaria o booleana no es necesario tener un valor en la intersección usuario-ítem ya que de las dos posibles opciones solo es de interés una de ellas, que ocurre cuando el usuario ha evaluado el ítem (1 o verdadero), por lo tanto podemos obviar la valoración; resultando en un conjunto de datos sin valores de preferencias. La Tabla II muestra un ejemplo de la representación de votación booleana o binaria visto como una matriz, en ella se ha colocado un 1 para denotar que existe una preferencia del usuario hacia el ítem, mas no representa en sí un valor dentro de una escala numérica discreta como ocurre con la votación valorada.

Usuarios/Ítems	Ítem 1	Ítem 2	Ítem 3	Ítem 4
Usuario 1	1	-	1	1
Usuario 2	1	1	-	1
Usuario 3	1	1	1	1

Tabla II – Matriz de preferencias de Usuarios vs Ítems usando votación booleana o binaria

En cada aplicación, dependiendo de la forma de interacción que se implementa se pueden interpretar y procesar de distintas maneras las evaluaciones obtenidas, así algunos sistemas que por ejemplo tomen en cuenta el tiempo en segundos que un usuario permanece viendo una página web podrían tomar ese valor directamente o decidir convertirlo en una representación binaria o booleana considerando la interacción usuario-ítem solo si el tiempo supera cierta cantidad de segundos.

La matriz resultante, representa el conjunto de preferencias de todos los usuarios, cada una de sus celdas es un voto $v_{i,j}$ de un usuario i por un ítem j.

Nótese que la estructura formada por usuario, ítem y preferencia es la adecuada para la representación del conjunto de datos producto de las evaluaciones de los usuarios por los ítems, por

consiguiente es la apropiada para crear los modelos de datos de entrada de un sistema de recomendaciones.

Enfoques de algoritmos para filtro colaborativo

Existen dos enfoques para filtro colaborativo, basados en memoria y basados en modelo. Los algoritmos basados en el modelo generan un modelo estadístico que es aprendido para luego hacer predicciones, puede ser visto como el cálculo del valor esperado de un valor de preferencia dado lo que sabemos del usuario. En los cálculos se utilizan redes bayesianas, agrupamiento, regresión lineal, álgebra lineal, métodos probabilísticos [7]

Los algoritmos basados en memoria usan métricas de similaridad basadas en vectores, correlaciones y coeficientes para predecir el valor de preferencia con la que un usuario evaluaría un ítem que desconoce.

Métricas de similaridad

Una métrica de similaridad es una medida que determina el grado de similaridad entre dos objetos. Podemos encontrar métricas que se basan en correlación, en similaridad de vectores, y coeficientes.

Su aplicación en un sistema de recomendación consiste en determinar usuarios o ítems similares entre sí.

De las métricas que se detallan a continuación el coeficiente de correlación de Pearson y el cálculo basado en coseno son útiles para hallar similaridad cuando los valores de preferencia del usuario por los ítems se expresan con votación valorada. En cambio, cuando la preferencia es expresada de forma binaria o booleana, no son útiles, por ello se usan distintas métricas tales como el coeficiente de similaridad Tanimoto y coeficiente Loglikelihood, que no requieren de valores de preferencia.

Similaridad basada en coseno

Esta similaridad está dada por la medición del coseno del ángulo entre dos vectores, obtenido del producto punto entre ambos. Cada usuario es tratado como un vector m-dimensional donde m es la cantidad de ítems que han sido evaluados por ambos usuarios. El valor de la similaridad está dado por la Ecuación 1.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} \qquad \text{(Ec. 1)}$$

Dónde θ es el ángulo que se forma entre los vectores *A* y *B*. $A \cdot B$ es el producto punto entre ambos. $\|A\| \|B\|$ son los módulos de *A* y *B* respectivamente.

Ya que los valores de preferencia de los usuarios son números positivos, el coseno del ángulo solo tomará valores positivos en el rango [0,1]. El máximo valor de similaridad, 1, se da cuando ambos vectores (usuarios) apuntan en la misma dirección, es decir que han evaluado los mismos ítems con los mismos valores, por el contrario mientras el coseno del ángulo más se aproxime a 0 menos similares serán.

Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es un índice que mide el grado de relación entre dos variables siempre y cuando ambas sean cuantitativas, es decir que se expresen mediante cantidades numéricas. Se aplica a variables con relación lineal. El coeficiente de correlación de Pearson entre dos variables *x* y *y* queda expresado en la Ecuación 2.

(Ec. 2)

$$\rho_{x,y} = \frac{\sigma_{x,y}}{\sigma_x \sigma_y}$$

Donde $\sigma_{x,y}$ es la covarianza entre *x* y *y*, σ_x es la desviación estándar de *x*, σ_y es la desviación estándar de *y*.

Si las variables son cuantitativas discretas, tal como sucede con los valores de preferencia de los usuarios de un sistema de recomendación, podemos expresar el coeficiente como se detalla en la Ecuación 3.

$$\rho_{x,y} = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum x^2}\sqrt{\sum y^2}} \qquad \text{(Ec. 3)}$$

Donde *x* es cada valor de la variable *x*, y $\bar{x}$ es su media aritmética. De la misma manera *y* es cada valor discreto de la variable *y*, y $\bar{y}$ es su media aritmética.

El rango del coeficiente es [-1,1] donde obtener valores en el rango (0,1] representa una correlación positiva, si una de las variables aumenta la otra también. Un coeficiente igual a cero significa que no existe una relación lineal entre ambas variables. Obtener valores en el rango [-1,0) representa una correlación inversa lo cual significa que mientras una aumenta la otra disminuye, o viceversa [8].

En un sistema de recomendaciones cada variable y sus valores discretos corresponden normalmente a un usuario y los valores de sus preferencias respectivamente.

Coeficiente de similaridad de Tanimoto

Teniendo en cuenta dos conjuntos A y B que tienen elementos en común, el coeficiente de similaridad de Tanimoto, también conocido como índice Jaccard, se expresa como la relación entre la cantidad de elementos que coinciden en ambos conjuntos y el total de elementos de ambos. El coeficiente se expresa con se muestra en la Ecuación 4.

$$T_c = \frac{K_{11}}{(K_{01} + K_{10} + K_{11})} \qquad \text{(Ec.4)}$$

Donde K_{11} es el número de elementos que ambos conjuntos tienen en común, K_{01} es el número de elementos que pertenecen al segundo conjunto pero no al primero y K_{10} es el número de elementos que pertenecen al primero pero no al segundo [9].

Aplicando este coeficiente a un sistema de recomendaciones cada conjunto corresponde a un usuario cuyo contenido son los ítems por los que tiene preferencia. El coeficiente toma valores en un rango [0,1] donde 0 significa que no hay similaridad porque ambos

usuarios no tienen ítems preferidos en común, y 1 representa similaridad total porque ambos usuarios han preferido los mismos ítems. La Figura 2-3 muestra la representación de dos usuarios, cada uno como conjunto de ítems que prefirieron. Usando ambos conjuntos el coeficiente de Tanimoto es igual a la Ecuación 5.

Figura 2-3 – Conjunto de preferencias de dos usuarios

$$T_c = \frac{Items\ que\ A\ y\ B\ prefieren}{Items\ preferidos\ solo\ por\ A + Items\ preferidos\ solo\ por\ B + \ Items\ que\ A\ y\ B\ prefieren}$$

(Ec. 5)

Coeficiente Loglikelihood

El coeficiente de Loglikelihood expresa cuántas veces es más probable que los datos estén en un modelo que en otro. Se calcula analizando cuentas de ocurrencia de eventos [10]. Teniendo dos eventos, las cuentas de ocurrencia que se conoce de ellos son:

k11 : El número de veces los eventos ocurren juntos.

k12 : El número de veces que el segundo evento ocurrió sin el primer evento.

k21 : El número de veces que el primer evento ocurrió sin el segundo evento.

k22 : El número de veces que algo más ocurrió, no ocurrieron ni el primero ni el segundo evento

Tomando las cuentas mencionadas el coeficiente de Loglikelihood, LLR, entre dos eventos está dado por la Ecuación 6.

$$LLR = 2 \times K \times \left(H_c(k_{11}, k_{12}, k_{21}, k_{22}) - \left(H_c(k_{11}, k_{12}) + H_c(k_{21}, k_{22})\right) - \left(H_c(k_{11}, k_{21}) + H_c(k_{12}, k_{22})\right)\right)$$

(Ec. 6)

Donde K es la suma total de ocurrencias y Hc es la entropía de varias cuentas de ocurrencia de evento. La entropía de conjunto se calcula por medio de la Ecuación 7.

$$H_c(C) = H\left(\sum_{x \in C} x\right) - \sum_{x \in C} H(x) \quad \text{(Ec. 7)}$$

Donde C es el conjunto de al menos dos valores de cuentas de ocurrencia mencionadas; y H es la entropía para un solo valor numérico, determinada por la Ecuación 8.

$$H(x) = \frac{x}{K} \times \log\frac{x}{K} \qquad \text{(Ec. 8)}$$

En un sistema de recomendaciones cada cuenta se interpreta de la siguiente manera tomando en cuenta dos usuarios para los que queremos calcular similaridad:

k11 : El número de ítems dos usuarios prefieren.

k12 : El número de ítems que el segundo usuario prefirió y el primero no.

k21 : El número de ítems que el primer usuario prefirió y el segundo no.

k22 : El número de ítems que ninguno de los dos usuarios prefirió.

Procesamiento de valores de preferencias para sistemas basados en memoria

Procesamos las preferencias de los usuarios para predecir qué tan importantes son para ellos los ítems que desconocen. La predicción del valor de preferencia o voto se realiza con funciones de agregación, que toman una colección de valores y devuelven como resultado un único valor [11]. Dado un usuario al que se le quieren hacer recomendaciones, al cual llamamos usuario objetivo,

y un ítem que él desconoce, para predecir el valor que el usuario objetivo le asignaría al evaluarlo se procesan los valores de preferencias de otros usuarios hacia ese ítem. El caso más básico aplicado al procesamiento de preferencias es un promedio de sus valores, que se lo calcula por medio de la Ecuación 6.

$$v_{i,j} = \frac{1}{N} \sum_{i' \in U'} v_{i',j} \qquad \text{(Ec. 9)}$$

Donde $v_{i,j}$ es el voto desconocido de un usuario objetivo *i* por un ítem *j*, *U'* es el conjunto de usuarios que evaluaron el ítem *j* y *N* es el número de usuarios en *U'*. Otra forma de predecir el voto es usando una suma ponderada, como se muestra en la Ecuación 7.

$$v_{i,j} = k \sum_{i' \in U'} w(i,i') \times v_{i',j} \qquad \text{(Ec. 10)}$$

Esta fórmula presenta un detalle adicional que la pone en ventaja sobre la primera, una ponderación. La ponderación está representada por *w(i,i')* y normalmente es el valor resultante de alguna métrica de similaridad usada entre el usuario *i'* y el usuario objetivo *i*. La ponderación le da un peso al voto de cada usuario, dándole mayor o menor relevancia. Esta fórmula permite usar cualquier métrica de similaridad gracias al factor *k* que la normaliza. El valor de *k* se lo obtiene mediante la Ecuación 8.

$$k = \frac{1}{\sum_{i' \in U'} |w(i, i')|} \qquad \text{(Ec. 11)}$$

Sin embargo esta fórmula no toma en cuenta que distintos usuarios pueden usar el rango de valoraciones de manera diferente para evaluar un ítem.

En un sistema que proporciona una escala entre uno y diez para dar valoraciones a sus ítems, dos usuarios A y B pueden decidir usar distintos valores que para denotar su máximo grado de preferencia por un ítem. Si para A ese valor es 10 y para B es 8 y si ambos son igual de similares que el usuario objetivo, el voto de A tendrá mayor relevancia que el de B al usar la suma ponderada.

Para superar este inconveniente la suma ponderada se ajusta usando desviaciones respecto al promedio de valores de preferencia del usuario *i'* , en vez de los valores absolutos de los votos, como se lo muestra en la Ecuación 9.

$$v_{i,j} = \overline{v}_i + k \sum_{i' \in U'} w(i, i') \times \left(v_{i',j} - \overline{v}_{i'}\right) \qquad \text{(Ec. 12)}$$

Donde $\overline{v}_{i'}$ y $\overline{v}_{i}$ son los promedios de los valores de preferencias de los usuarios *i* e *i'* respectivamente. El promedio de los valores de preferencias $\overline{v}$ de un usuario está dado por la Ecuación 10.

$$\overline{v} = \frac{1}{|A|} \sum_{j \in A} v_j \qquad \text{(Ec. 13)}$$

Donde *A* es el conjunto de ítems que el usuario evaluó, v_j es el valor de preferencia del usuario por un ítem *j.*

Procesamiento de preferencias sin valores de preferencias

Para el caso booleano donde el modelo de datos no tiene valores de preferencias no se deben aplicar estas operaciones, se usan otros cálculos para determinar qué tan importante es un ítem para el usuario. El resultado puede ser visto como un valor de preferencia. Una de las opciones es la suma de las ponderaciones, se la muestra en la Ecuación 11.

$$v_{i,j} = \sum_{i' \in U'} w(i, i') \qquad \text{(Ec. 14)}$$

Una observación hacia este cálculo es que el valor de preferencia será el mismo para ítems diferentes que han sido evaluados por los mismos usuarios de *U'.*

Enfoques de los sistemas basados en memoria

Uno de los enfoques, conocido como "*basado en el usuario"*, recomienda al usuario los ítems que evaluaron otros usuarios con gustos similares.

Otro de los enfoques, "*basado en el ítem"*, se sustenta en el hecho que a las personas les suele gustar cosas similares a las que ya les gusta, por lo tanto recomienda a un usuario los ítems similares a los que ha preferido antes.

Problemas de los sistemas de recomendación de filtro colaborativo

Existen varias situaciones que al presentarse en un recomendador de filtro colaborativo pueden representar problemas al momento de realizar las recomendaciones. A continuación se nombran y explican los diferentes tipos de problemas.

El problema del nuevo usuario

Se presenta en los basados en memoria y basados en modelo, cuando un usuario es nuevo en el sistema es difícil recomendarle ítems ya que no hay evaluaciones pasadas conocidas de él.

El problema del nuevo ítem

Un ítem nuevo no ha sido evaluado por nadie por lo que no se sabe a quién le gusta o no.

Escasez de valores de preferencias.

Se da cuando el número preferencias conocidos de un usuario es muy reducido, la escasez de esta información dificulta la predicción de preferencias precisas.

Gustos inusuales

Ocurre cuando un usuario tiene gustos inusuales, lo que hace complicado hallar usuarios similares a él. Para contrarrestar este problema se puede usar los datos de perfil de usuario para hallar sus similares, comparando su localización, género, edad y demás información que los caracteriza.

Para contrarrestar los problemas del nuevo usuario y nuevo ítem, se suelen usar ambos enfoques, memoria y modelo en el mismo sistema.

2.1.2. Recomendador Basado en el Usuario

Un sistema de recomendación basado en el usuario se sustenta en el hecho que a las personas les suele gustar las mismas cosas que a otras personas con preferencias similares y se enfoca en la similaridad que hay entre varios usuarios. El objetivo es estimar el valor de preferencia del usuario objetivo por un ítem que él no conoce, pero los otros usuarios sí. El usuario objetivo es aquel al que se le hacen recomendaciones. Las fases del recomendador basado en el usuario son las siguientes:

a) *Hallar qué tan similares son todos los usuarios al usuario objetivo.*

Para hallar la similaridad entre dos usuarios se utiliza el conjunto de preferencias de cada uno. Tomando ambos conjuntos de preferencias se hacen cálculos usando una métrica de similaridad. El proceso se realiza entre el usuario objetivo y cada uno de los demás usuarios. El resultado de esta

fase es una lista de valores dados por la métrica de similaridad que determinan qué tan similar es el usuario objetivo a cualquier otro usuario.

b) *Formar una vecindad.*

Una vecindad es un conjunto de usuarios que comparten gustos similares y que puede estar restringido por uno de los siguientes factores: la cantidad de individuos que lo forman o un valor mínimo de similaridad.

El primer factor determina que una vecindad está formada por los *n* vecinos más cercanos al usuario objetivo, es decir los *n* usuarios cuya similaridad con el usuario objetivo es la más alta.

El segundo factor determina que una vecindad está restringida por un umbral de similaridad, es decir que los usuarios que forman la vecindad son aquellos cuya similaridad con el usuario objetivo es mayor o igual a un valor.

De esta manera quedan definidos dos tipos de vecindades, la de *N cercanos* y la de *umbral.*

c) *Tomar los ítems desconocidos para el usuario objetivo y que han sido evaluados por los usuarios de la vecindad.*

Los posibles ítems de interés para el usuario objetivo son aquellos que aún desconoce. El sistema de recomendaciones solo podrá recomendarlos si conoce información sobre ellos, dicha información proviene de las preferencias de otros usuarios, específicamente de los que pertenecen a la vecindad.

d) *Para cada uno de los ítems que el usuario objetivo no conoce haga la predicción del valor de preferencia con el que el usuario objetivo lo evaluaría*.

La predicción del valor de preferencia se realiza con funciones de agregación antes descritas para procesamiento de preferencias, que toman una colección de valores y devuelven como resultado un único valor. Para el caso booleano se determina qué tan importante es un ítem para el usuario objetivo sin usar valores de preferencia.

A continuación en la Figura 2-4 se muestra el pseudocódigo del algoritmo de un sistema de recomendación de filtro colaborativo basado en el usuario:

```
1.Inicio.
2.Para cada usuario que no es el usuario objetivo.
        3.Calcular una similaridad S con el usuario objetivo y forme una
          vecindad
4.Para cada item i que un usuario de la vecindad ha evaluado pero el usuario
  objetivo no
        5.Para cada usuario v de la vecindad que evaluó  i
            6.Realice  la predicción del valor de preferencia del usuario
              objetivo hacia i usando el valor de preferencia de v hacia i.
7.Fin.
```

Figura 2-4 – Algoritmo de un recomendador de filtro colaborativo basado en el usuario

En las líneas 2 y 3 se realiza la primera fase y segunda fase del algoritmo basado en el usuario, se calcula la vecindad del usuario objetivo usando la similaridad hallada entre él y los demás usuarios. De la línea 4 hasta la 6 se realiza la tercera y cuarta fase, se hace el cálculo de la predicción de preferencias del usuario objetivo.

La Figura 2-5 muestra un esquema general de los sistemas de recomendaciones, detallando la subdivisión de los basados en filtro colaborativo.

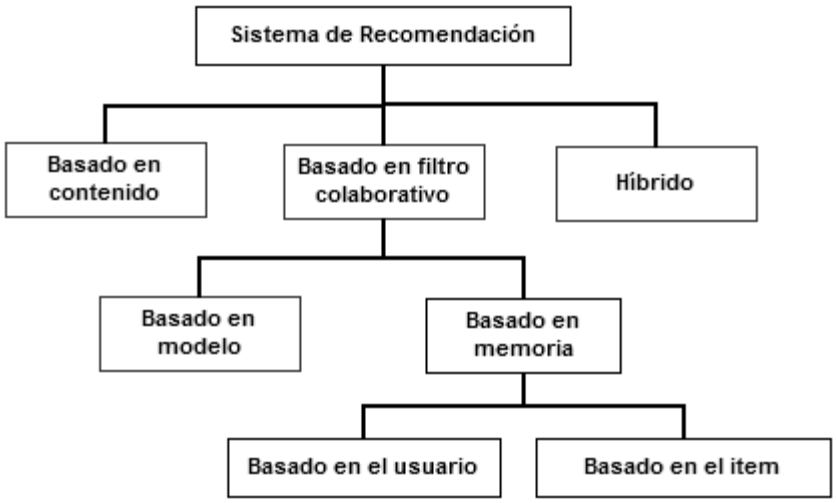


Figura 2-5 – Esquema de los sistemas de recomendación

CAPÍTULO 3

3. ANÁLISIS DE LA SOLUCIÓN.

La ESPOL posee registros de todas las materias aprobadas que los estudiantes obtienen a lo largo de su carrera. Estos datos resultan ser aprovechables para obtener información acerca de experiencias exitosas de registros de materias. Estas experiencias pueden ayudar a establecer una guía para otros estudiantes, al momento de elegir una combinación idónea de materias durante un registro ya que pueden ser interpretadas como preferencias implícitas.

Nuestro trabajo parte de la lógica de recomendación basada en el usuario que establece que: si los gustos de un usuario determinado son parecidos a los gustos de un grupo de usuarios, es posible deducir un grupo de ítems idóneos para el usuario si se le recomienda aquellos ítems que eligieron los usuarios del grupo similar. Adaptando

la misma idea al caso de recomendación de materias de pregrado, podemos inferir que es posible recomendar materias a un estudiante buscando estudiantes que hayan tenido un historial académico similar y que es muy probable que aquellas materias que ellos eligieron en el semestre que se está recomendando serán las materias idóneas para el estudiante en cuestión.

Debido a que las recomendaciones se basarán en grupos de usuarios similares con respecto a el historial académico, se define que el tipo de recomendador a construir será uno de filtro colaborativo basado en el usuario, donde el grado de similaridad de los usuarios se definirá por el grado de similitud en su historial académico, es decir, el orden en que sus materias han sido aprobadas hasta el momento. En la Figura 3-1 se señala el tipo de recomendador usado en este trabajo en el esquema de los tipos de recomendadores antes mostrado.

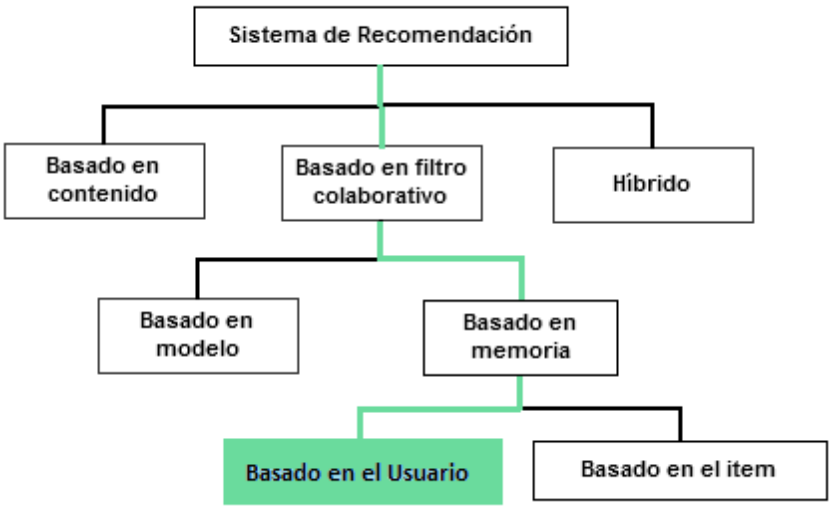


Figura 3-1 – Tipo de recomendador seleccionado

También resulta necesario realizar un análisis de las métricas de similaridad usadas en la implementación de la solución, ya que escoger de manera acertada estas métricas así como el tipo de vecindad a usarse en el recomendador, permiten obtener valores de acierto mejores en las recomendaciones, este aspecto se lo detallara en el capítulo 4.

3.1. Requerimientos.

Tomando en consideración que la solución propuesta se centra en la construcción de un recomendador de filtro colaborativo basado en el usuario, se definen las características principales que este debe cumplir en las subsecciones a continuación.

3.1.1. Requerimientos Funcionales.

- El recomendador recibe el número de materias que debe recomendar.
- El recomendador permite especificar el semestre para el cual se desea obtener la recomendación.
- El recomendador recomienda materias que el estudiante no haya cursado antes del semestre del que se pide recomendación.
- Las materias recomendadas deben cumplir con los pre-requisito y co-requisitos del flujo de la carrera del estudiante al cual se le realiza la recomendación.
- Las recomendaciones son basadas en las experiencias de aquellos estudiantes que hayan recorrido el flujo de su carrera de manera similar al del estudiante que recibe la recomendación.

3.1.2. Requisitos No Funcionales.

Escalabilidad

El recomendador debe ser construido de tal manera que se puedan añadir nuevos criterios de valoración en las recomendaciones.

Flexibilidad

El recomendador puede adaptarse a cualquier carrera de la ESPOL, y las condiciones que cada una de ellas contienen en su malla curricular.

Confiabilidad

El recomendador debe dar como resultado materias relevantes a la carrera, semestre e historial académico del estudiante que recibe la recomendación.

3.2. Análisis de las Capacidades del Recomendador Basado en el Usuario

Haciendo referencia al capítulo 2 en donde se explica el funcionamiento de un recomendador basado en el usuario, describiremos los retos que se presentan al basar las recomendaciones en el historial académico de los usuarios y los procesos que deben ser añadidos a este recomendador para realizar

la adaptación al escenario descrito en el problema.

Se debe tener en claro que al usarse un recomendador basado en el usuario en su estado genérico para realizar las recomendaciones de las materias, tendremos como resultado recomendaciones no muy personalizadas. El motivo principal es la manera que en los datos son representados en el recomendador puesto que al crearse el modelo de datos con la información de los estudiantes y sus materias aprobadas, solo se establecerán relaciones que representen que un estudiante aprobó un curso, el saber en qué semestre de la carrera esto sucedió no será posible.

Las vecindades tomarán como usuarios similares a aquellos estudiantes que hayan aprobado hasta el momento las mismas materias, sin importar si estas fueron tomadas y aprobadas en los mismos semestres. Esto ocasiona que las vecindades tengan grupos grandes que no necesariamente juntarán usuarios similares.

Por ejemplo, si se desea realizar una recomendación a un estudiante que tenga exactamente la mitad de su malla curricular aprobada, sus recomendaciones se basaran en todos aquellos estudiantes que tengan más de la mitad de la malla curricular aprobada sin ningún otro

tipo de distinción. Se podría decir que para el recomendador todos aquellos estudiantes que le lleven ventaja en el número de semestre que se encuentre cursando el estudiante objetivo serán similares a él. Debido a esto las materias recomendadas no necesariamente reflejaran la manera en que el estudiante que pide la recomendación ha decidido recorrer su malla. En la Figura 3-2 se puede observar que para el ejemplo mencionado, la vecindad para efectuar la recomendación contendrá a todos los demás estudiantes de la carrera que se encuentren más avanzados en el flujo.

Figura 3-2 – Agrupación de estudiantes en un recomendador basado en el usuario

Para resolver este inconveniente, se recorre semestre a semestre el historial del estudiante que ha pedido la recomendación. Por cada semestre se identifican las personas que hayan aprobado las mismas materias que él en ese semestre. En otras palabras, se obtendrá una vecindad para cada uno de los semestres que el estudiante objetivo haya cursado, como se ilustra en la Figura 3-3.

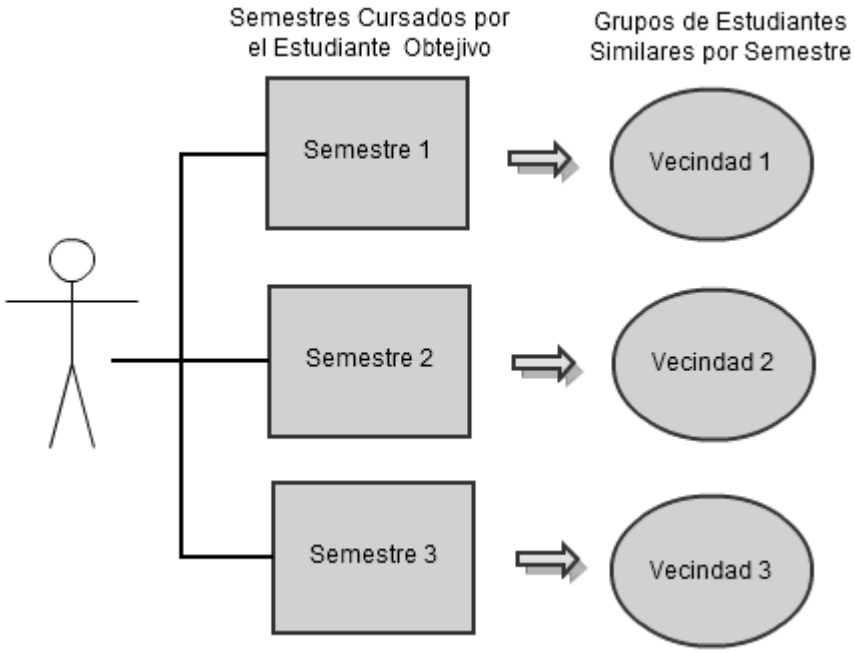


Figura 3-3 – Esquema del recorrido del historial académico del estudiante

Por medio de este conjunto de vecindades se obtienen a los estudiantes que posean un perfil académico similar al del estudiante que recibe la recomendación. El grado de similaridad que deban

cumplir los perfiles académicos dependerá de la rigidez que se establezca en el recomendador. Como casos extremos de niveles de similaridad tenemos que el mas especifico define como estudiantes similares a aquellos que hayan pertenecido a todas las vecindades obtenidas por semestre, mientras que el grado de similaridad mas general permite definir como similares a aquellos estudiantes que hayan aparecido en al menos una vecindad.

De esta manera las recomendaciones se basan en aquellos estudiantes que cumplan con el grado de similaridad requerido y que hayan cursado el semestre del que se está pidiendo recomendación. Las materias recomendadas son aquellas que han sido tomadas por el grupo de estudiantes similares en el semestre del que se está pidiendo recomendación, además las materias deben pasar por las validaciones que verifiquen los pre-requisitos y los co-requisitos.

CAPÍTULO 4

4. DISEÑO E IMPLEMENTACIÓN DEL RECOMENDADOR DE MATERIAS.

En este capítulo se describe la metodología usada para la implementación del recomendador de materias de pregrado basado en el historial académico de los estudiantes. Posteriormente se detalla el plan de pruebas usado.

4.1. Modelo del Recomendador de Materias.

La adaptación del recomendador basado en el usuario provocó un cambio tanto en su estructura, como en los procedimientos normales que obtenían las recomendaciones. Los cambios realizados se explican en las etapas descritas en la siguiente sección.

4.1.1. Modelo Lógico del Recomendador de Materias

El procedimiento para obtener las recomendaciones se centra en la obtención de los estudiantes que posean un historial académico similar al del estudiante que pide la recomendación. Esto se traduce dentro del recomendador en la creación de un nuevo tipo de vecindad a la cual llamaremos *Vecindad por Historial.*

La construcción de la Vecindad por Historial involucra manejar de manera independiente las aprobaciones que se han dado en cada uno de los semestres cursados a lo largo de la carrera de un estudiante. Para explicar mejor esta idea partamos del caso en que un estudiante pida una recomendación para un *semestre N*, en adelante a este estudiante lo llamaremos *Estudiante E.*

Modelo de Datos$_N$

El recomendador empezará su proceso de recomendación recibiendo el *Modelo de Datos$_N$*, que contiene todas las aprobaciones realizadas en el *semestre N* de una carrera. El *Estudiante E* no existe dentro de este modelo de datos, ya que el aún no ha cursado este semestre, pero las materias que serán

recomendadas provienen de allí, ya que este modelo representa las decisiones de registro que los demás estudiantes tomaron al cursar su *semestre N*. Este modelo de datos será procesado en las últimas fases de la recomendación.

Modelos de Datos$_m$, sus vecindades$_m$ y similaridades$_m$

El siguiente paso consiste en crear los modelos de datos que representen los N-1 semestres que el *Estudiante E* ha cursado hasta el momento. En este paso se analiza el historial académico de este estudiante para encontrar sus usuarios similares por semestre. Para su efecto, se recorren los N-1 modelos de datos. Por cada *Modelo de Datos$_m$* que pertenezca a ese conjunto, se obtiene la *Vecindad$_m$* y la *Similaridad$_m$* que los estudiantes dentro de ese modelo de datos poseen con respecto al *Estudiante E.* Además, se obtiene la lista de materias que el *Estudiante E* aprobó en el semestre que representa ese modelo de datos. A continuación se presenta la Figura 4-1 con el pseudocódigo que ilustra los pasos descritos.

```
1. Inicio.
2. Variables m, materiasAprobadas, modeloDatos[N-1], vecindad [N-1], similaridad [N-1].
3. Para m=1 hasta N-1
    4. crear ModeloDatos[m] del semestre m.
    5. similaridad[m]=obtener similaridad de ModeloDatos[m].
    6. vecindad[m]=obtener vecindad de ModeloDatos[m].
    7. añadir a materiasAprobadas las materias que se encuentras relacionadas al
       Estudiante E en el ModeloDatos[m]./*las materias que el Estudiante E aprobó en el semetre m*/
8. Fin.
```

Figura 4-1 – Cálculo de los modelos de datos, sus vecindades y sus similaridades

En la línea 5 y 6 del pseudocódigo, la $Vecindad_m$ y la $Similaridad_m$ son obtenidas mediante los métodos inherentes del recomendador genérico basado en el usuario, por lo que los estudiantes que pertenezcan a la $Vecindad_m$ y los valores de similaridad dentro de $Similaridad_m$ dependen directamente de la métrica de similaridad (*Loglikelihood, Tanimoto*) usada en el recomendador, así como del tipo de vecindad (*N cercanos, umbral*) configurada en el mismo.

Lista Materias Aprobadas

En la línea 7 del pseudocódigo se añaden las materias que el *Estudiante E* aprobó en cada semestre a la lista *Materias Aprobadas*. Esta lista tiene como función proporcionar un acceso rápido al historial académico del *Estudiante E*.

Tabla de Frecuencia

Una vez obtenido el conjunto de *Vecindades Vm*, se construye la *Tabla de Frecuencia*. Esta tabla, representada por un HashMap, tiene como objetivo ordenar a todos los estudiantes que pertenecen al conjunto de *Vecindades Vm* y otorgar a cada uno de ellos un número con valores entre 1 y N-1. Este número simboliza la cantidad de *Vecindades Vm* a las que pertenece el estudiante. La estructura de la *Tabla de Frecuencia* se detalla en la Tabla III.

Key	**Value**
ID del Estudiante	Número de vecindades Vm a las que pertenece el estudiante

Tabla III – Estructura de la Tabla de Frecuencia

En la Figura 4-2 se muestra un ejemplo gráfico donde el *Estudiante 1* pertenece a las vecindades: *Vecindad 1*, *Vecindad 2* y *Vecindad 3*. El registro de este *Estudiante 1* en la *Tabla de Frecuencia* tiene un *value* igual a 3.

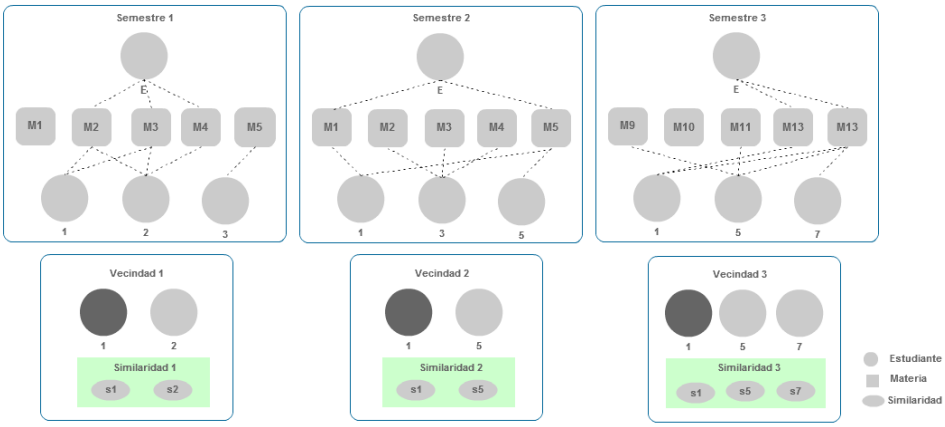


Figura 4-2 – Concurrencia de un estudiante en varias vecindades

Vecindad por Historial

De la *Tabla de Frecuencia* se escogen los estudiantes que conforman la *Vecindad por Historial*. La manera de filtrar los estudiantes varía según el tipo de vecindad que se usa en el recomendador, es decir, teniendo dos opciones de tipo de vecindad, *N cercanos y umbral*, los procesos de selección de estudiantes serán los siguientes:

a) Selección para vecindades de tipo N cercanos

Como se explicó en el capítulo 2, este tipo de vecindad se crea con un tamaño fijo de estudiantes definido en el recomendador. Por ejemplo, si se define un valor de n igual a 27, la vecindad se formara por los 27 estudiantes más similares al *Estudiante*

E. Esta misma lógica se aplica en la selección de estudiantes de la *Tabla de Frecuencia*. Los n estudiantes que tengan los mayores valores en la tabla son los que conforman la *Vecindad por Historial*.

b) Selección para vecindades de tipo umbral

Haciendo referencia al capítulo 2, los estudiantes que pertenecen a este tipo de vecindad deben tener un valor de similaridad mayor igual al valor de umbral definido. Aplicar esta lógica de selección a la *Tabla de Frecuencia* presenta un reto, ya que en esta tabla los estudiantes adquieren importancia según el valor de frecuencia que estos posean y no en base a su similaridad con el Estudiante E.

Es por esto que es imperativo realizar una conversión de un valor de similaridad a un valor de frecuencia para poder filtrar los estudiantes de la Tabla de Frecuencia a la Vecindad por Historial. Dicha conversión se describe en la siguiente Ecuación 15:

Valor de Frecuencia = (Umbral*FrecuenciaMaxima) (Ec.15)

Donde Valor de Frecuencia es el valor de Frecuencia traducido o adaptado del Valor de Similaridad. Umbral es el valor asignado de Umbral por el usuario y FrecuenciaMaxima es la máxima frecuencia obtenida en todas las vecindades de todos los estudiantes. La similaridad puede tomar un entre uno y de cero, siendo uno el valor más restringido y cero el menos restringido.

La creación de la Tabla de Frecuencia y su filtrado representan una fase importante dentro de este proceso de recomendación. En esta etapa es donde se realiza la comparación de los perfiles académicos de los estudiantes, para posteriormente obtener el grupo de estudiantes que posee una tendencia parecida al *Estudiante E* al aprobar su flujo.

Similaridad Promedio

El siguiente paso consiste el calcular un valor de similaridad promedio para cada uno de los estudiantes que pertenecen a la *Vecindad por Historial.* Para esto usaremos el conjunto de $similaridades_m$ que se obtuvo al inicio del proceso.

El método para calcular la *similaridad promedio* de un estudiante consiste en sumar los valores de similaridad que este obtuvo en cada una de las *similaridades*$_m$, luego esta suma se divide para N-1, que como se definió anteriormente es el número de semestres que el Estudiante E ha cursado. A continuación en la Figura 4-3 se presenta el pseudocódigo del proceso explicado.

```
1. Inicio.
2. Variables m, acumulador=0, similaridadPromedio,similaridad [N-1].
3. Para m=1 hasta N-1
    4. Si estudiante tiene un valor de similaridad en  similaridad[m] entonces
        5. acumulador=obtener similaridad del estudiante con respecto al Estudiante E en
           similaridad[m].
6. similaridadPromedio=acumulador/(N-1).
7. Fin.
```

Figura 4-3 – Pseudocódigo de cálculo de similaridad promedio

Obtención de Materias Candidatas

Para obtener las materias candidatas a recomendación se enlistan todas las materias que pertenecen al *Modelo de Datos*$_N$. Estas materias son aquellas que los estudiantes han aprobado en el semestre N de su carrera, que es el mismo semestre del que se está pidiendo recomendación.

De las materias candidatas se remueven las materias que se encuentran en la lista *materias aprobadas.* Posteriormente se calcula un *valor de preferencia* a cada una de las materias candidatas.

Valor de Preferencia

El método para calcular el *valor de preferencia* a una materia candidata, consiste en detectar a todos los estudiantes de la *Vecindad por Historial* que hayan aprobado esa materia en el *Semestre N*, esta información se la obtiene del *Modelo de Datos*$_N$*.* El *valor de preferencia* es la suma de las *similaridades promedio* de los estudiantes detectados.

El valor de preferencia depende de dos factores: el número de estudiantes que aprobó la materia y el valor de similaridad que poseen los estudiantes que la aprobaron. La Figura 4-4 muestra el pseudocódigo del proceso del cálculo del valor de preferencia.

```
1. Inicio.
2. Variables materia, estudiante, vecindadHistorial, similaridadesPromedio, modeloDatosN,
            valorPreferencia=0.
3. Hacer mientras vecindadHistorial tenga un proximo estudiante
    4. estudiante = obtener proximo estudiante de vecindadHistorial.
    5. Si modeloDatosN contiene aprobacion de materia por estudiante
               6.valorPreferencia += obtener de similaridadPromedio la similaridad
                promedio del estudiante
7. Fin hacer mientras.
8. Fin.
```

Figura 4-4 – Pseudocódigo del cálculo del valor de preferencia

Recomendación de Materias

Una vez que se haya calculado el *valor de preferencia* a todas las materias candidatas, se ordena la lista *materias candidatas* de manera descendente según el *valor de preferencia* que posean. Las materias que se encuentran más próximas al inicio de la lista serán las recomendadas al *Estudiante E.* El número de materias a recomendar depende de la cantidad de materias que el *Estudiante E* haya solicitado al inicio del proceso.

4.1.2. Estructura del Recomendador de Materias

El procedimiento de recomendación explicado en la sección anterior provoca que la estructura del recomendador de materias

implementado en este trabajo sea diferente a la estructura de un recomendador genérico basado en el usuario.

El recomendador de materias consta de múltiples modelos de datos. Como se mencionó antes, estos modelos de datos representan las aprobaciones que los estudiantes obtuvieron en cada uno de sus semestres cursados. El número de modelo de datos que use el recomendador es igual al número de semestre del que se pide recomendación. Por ejemplo, si se pide recomendación para el quinto semestre, el recomendador tendrá cinco modelos de datos. Un modelo de datos contiene las aprobaciones del quinto semestre de los estudiantes en los que se basara la recomendación y de este modelo de datos se obtienen las materias que se recomiendan. Los demás modelos de datos, representan las aprobaciones que se dieron en los semestres anteriores, es decir, desde el primero hasta el cuarto semestre. Estos modelos de datos sirven para la obtención de los estudiantes similares que alimentan las recomendaciones.

Los demás modelos de datos, representan las aprobaciones que se dieron en los semestres anteriores, es decir, desde el primero hasta el cuarto semestre. Estos modelos de datos sirven para la

obtención de los estudiantes similares que alimentan las recomendaciones.

En la estructura del recomendador genérico basado en el usuario, los usuarios similares y los ítems a recomendar son obtenidos de un único modelo de datos. A continuación, en la Figura 4-5, se muestra la estructura del recomendador genérico basado en el usuario y la estructura del recomendador de materias basado en el historial académico.

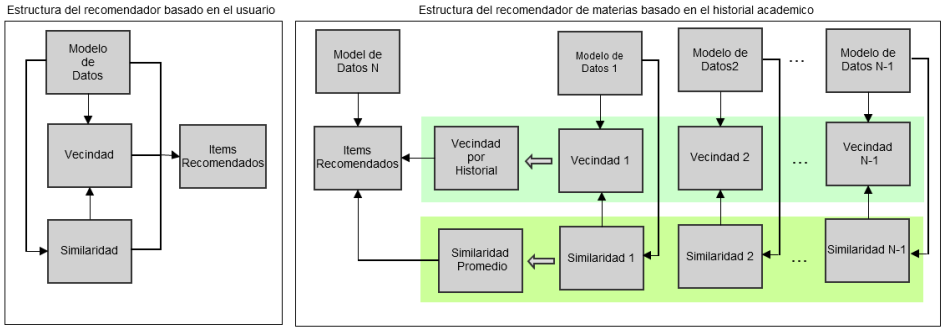


Figura 4-5 – Comparación de estructuras de un recomendador genérico basado en el usuario y del recomendador de materias basado en el historial académico

En la Figura 4-5 también se puede observar que por cada modelo de datos que representa uno de los semestres cursados por el estudiante que recibe la recomendación, se obtiene una vecindad y un grupo de similaridades. Estas vecindades y similaridades sirven para la construcción de la *Vecindad por Historial* y *las*

Similaridades Promedio respectivamente, y son estos dos componentes los que permiten seleccionar las materias a recomendar.

La estructura del recomendador genérico basado en el usuario tiene una vecindad y un grupo de similaridades, y con estos componentes se seleccionan los ítems a recomendar.

El número de componentes usados es un recomendador genérico basado en el usuario son 4: modelo de datos, vecindad, similaridad, ítems recomendados. Mientas que en el recomendador de materias basado en el historial académico, el número de componentes varía según el semestre del que se pide recomendación, la Ecuación 16 explicada a continuación permite calcular el número de componentes:

$$NC= 3 + N +2 (N-1) \qquad \text{(Ec. 16)}$$

Donde NC es número de componentes y N el número del semestre del que se pide recomendación. En esta ecuación, el número 3 representa la *Vecindad por Historial*, *Similaridad Promedio* y los Ítems Recomendados, N representa el número de modelos de

datos usados, y 2(N-1) representa el número de vecindades y similaridades construidas.

4.1.3. Reglas de Validación Aplicadas a las Materias Recomendadas

Cuando un estudiante desea registrarse en una materia, este debe cumplir los pre-requisitos y co-requisitos que la materia posea para que se haga efectivo su registro. Es por esto que el recomendador debe verificar que el estudiante cumpla con los pre-requisitos y co-requisitos de las materias seleccionadas antes de entregarlas como recomendación.

Para la validación de una materia primero se verifican sus pre-requisitos. Este primer proceso de verificación consiste en obtener una lista de pre-requisitos de la materia. Si la lista es vacía entonces la materia puede ser recomendada, sino se procede a verificar que todos los pre-requisitos de la lista ya hayan sido aprobados por el estudiante. En el caso de que todos los pre-requisitos ya hayan sido aprobados se procede a recomendar la materia.

En el caso contrario, se detiene la verificación en el primer pre-requisitos que no haya sido aprobado. Ahora este pre-requisito se convierte en la materia que se recomienda tomar. Por lo tanto a esta nueva materia debe pasar por la validación de sus pre-requisitos. El proceso de validación de pre-requisitos de una materia es recursivo. En la Figura 4-6 se presenta el pseudocódigo del proceso de validación.

```
1. Inicio.
2. Variables materia, requisitos,aprobadas=true.
3. requisitos= obtener requisitos de materia.
4. Si requisitos esta vacio.
    5. Retornar materia.
6. Para cada requisito que pertenezca a la lista requisitos hacer.

       7. Si requisito no ha sido aprobado.
              8. aprobadas=false.
              9. terminar recorrido de la lista de requisitos.
10. Si aprobadas es true.
    11. Retornar materia.
12. Sino.
    13. materia = ultimo requisito verificado.
    14. validar prerequisitos de materia.
14. Fin.
```

Figura 4-6 – Pseudocódigo de la validación de pre-requisitos

La nueva materia a recomendar adquiere el valor de preferencia de la materia que ha sido suplantada. Una vez verificados los pre-requisitos de todas las materias se procede a verificar los co-

requisitos.

La validación de co-requisitos de una materia consiste en verificar si el estudiante ya ha aprobado dicho co-requisito. Si no es el caso se revisa que el co-requisito se encuentre en la lista de materias a recomendar. Si el co-requisito no está, se lo añade a la lista de materias a recomendar con un valor de preferencia igual a cero. En el caso de añadir la materia a la lista de recomendación, esta recibe un valor de preferencia bajo por que no fue producto de la selección propia del recomendador. Su lugar en la lista es un recordatorio al estudiante de que si desea seguir la recomendación recibida debe tomar en cuenta esa materia. Esta solución fue implementada debido a que la presencia de los co-requisitos habitualmente se da en los primeros semestres de la carrera y son escasas.

4.2. Modelamiento de Datos para su Procesamiento.

La forma de interacción entre estudiantes y materias ha sido considerada como binaria o booleana, razón por la cual en nuestro modelo de datos no contamos con valores de preferencia. La

estructura del modelo solo consta de estudiante (usuario) y materia (ítem).

La información de materias aprobadas de los estudiantes fue separada en semestres. Cada año académico en ESPOL consta de tres términos, los dos primeros son regulares y el tercero es durante las vacaciones e irregular. Durante el término irregular los estudiantes pueden decidir tomar o no un máximo de tres materias. Dada la pequeña cantidad de estudiantes que realizan estudios durante sus vacaciones y la poca cantidad de materias que podrían aprobar en él, todas las materias de un estudiante que fueron tomadas en el tercer término son consideradas como aprobadas en el segundo término del mismo año.

Luego la información de materias aprobadas fue agrupada por estudiante, por año y por término y ordenada por las dos últimas del más antiguo al más reciente. La Figura 4-7 muestra la agrupación que se hizo por cada estudiante.

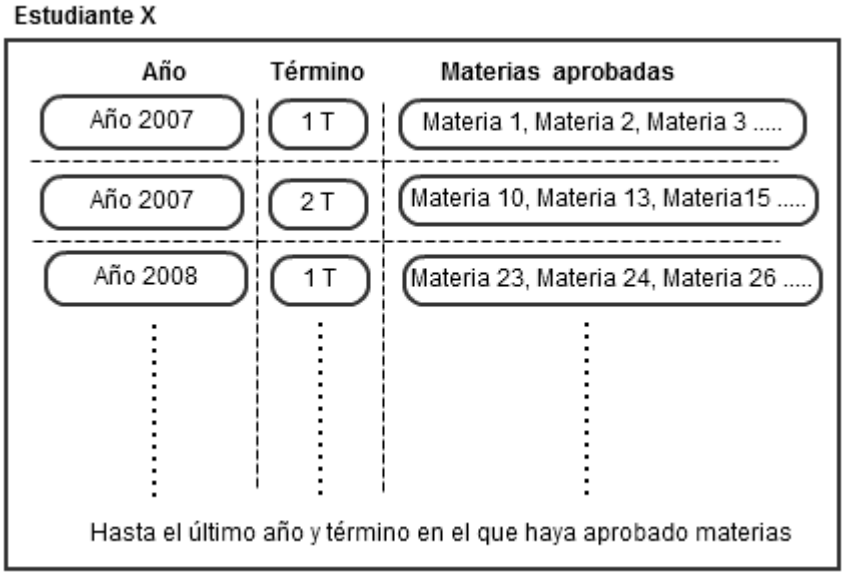


Figura 4-7 – Agrupación de materias aprobadas de un estudiante por año y término ordenados de manera descendente

El orden determina el número del semestre en el que las materias fueron tomadas. De acuerdo a ese orden se separa las materias agrupadas en nuevos grupos. Cada nueva agrupación contiene la información del estudiante y de las materias aprobadas de ese semestre. Los semestres van desde 1 hasta K, siendo K el número de semestre más alto para el cual alguno de los estudiantes aprobó materias.

La Figura 4-8 muestra cómo se formaron los semestres con los grupos de materias aprobadas de cada estudiante. En el gráfico se muestra el ejemplo para dos estudiantes, Y y X.

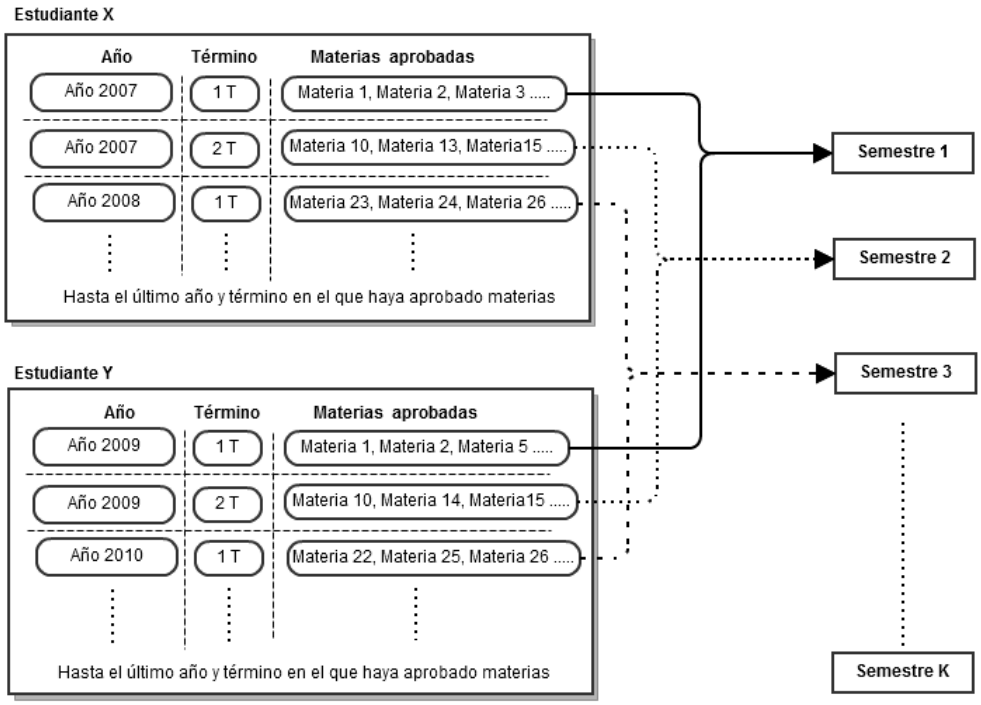


Figura 4-8 – Formación de semestres

Finalmente cada grupo de semestre es adaptado al modelo de datos con la estructura que será procesada por el sistema de recomendaciones. Cada registro o entrada del modelo de datos tendrá al estudiante y la materia que él aprobó en ese semestre. La Figura 4-9 muestra un ejemplo del contenido para el semestre 2 de los estudiantes Y y X.

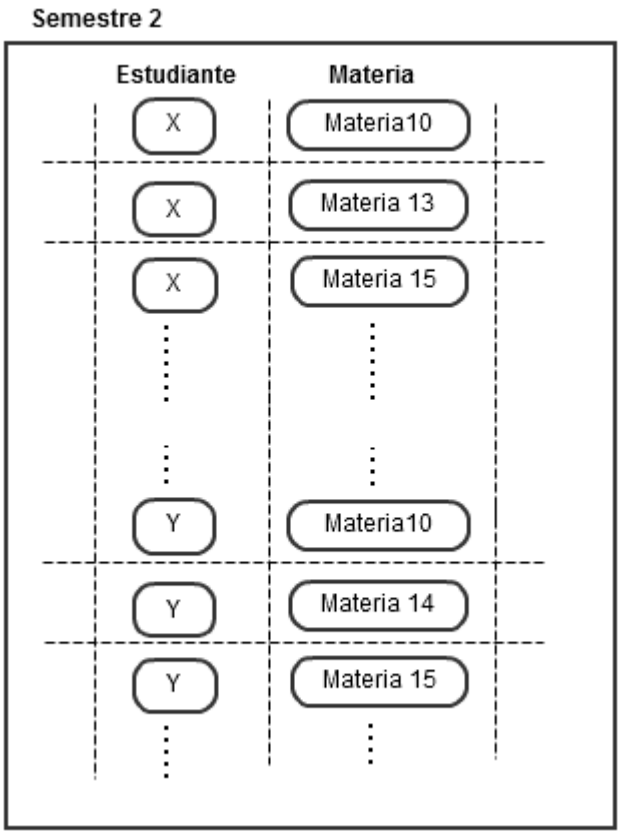


Figura 4-9 – Ejemplo de contenido de semestre

4.3. Plan de Pruebas

Usando el sistema de recomendaciones implementado se hicieron recomendaciones de seis materias que tomarían por semestre los estudiantes de la carrera de Ingeniería en Telecomunicaciones de la FIEC (Facultad de Ingeniería Eléctrica y Computación), ESPOL.

El motivo por el cual se usó esta carrera se debe a que es la que posee el mayor número de estudiantes registrados en la FIEC, esto es

fundamental para el recomendador, ya que la calidad de las recomendaciones está en gran parte determinada por la cantidad y la calidad de los datos usados para realizarlas [12].

Además, la ausencia de especializaciones en la carrera evita que el grupo de estudiantes registrados en ella se dividan en subgrupos, evadiendo así validaciones complejas que introduzcan cambios que pueden afectar los resultados finales de las recomendaciones

La información utilizada como entrada del sistema son los registros de materias aprobadas del grupo de estudiantes mencionado que han ingresado a la carrera desde el primer término del año 2007 hasta el primer término del año 2010. El máximo número de semestres que habrá cursado un estudiante en ese periodo es ocho, razón por la que las predicciones de materias se realizan desde su segundo hasta máximo su octavo semestre. En las pruebas realizadas ocho es el valor que corresponde a K, el número máximo de semestres, mencionado en la sección de modelamiento de datos. Para el primer semestre no se realizan recomendaciones usando la solución propuesta porque no se conoce historial académico previo a ese semestre.

Las recomendaciones se hicieron por semestre y por estudiante. Para determinar qué tan confiable es el sistema de recomendaciones tomamos las materias que un estudiante aprobó en un semestre y las comparamos con las que el sistema predice para ese semestre, es decir comparamos datos reales con datos predichos.

De aquí en adelante para efecto de describir las pruebas realizadas nos referimos como materias aprobadas a aquellas materias que el estudiante aprobó en el semestre del que se piden recomendaciones.

Mientras más materias aprobadas aparecen en las recomendaciones para un estudiante, mejor puntaje obtiene el recomendador. Las materias aprobadas por el estudiante son importantes en las pruebas porque el resultado ideal en ellas es que el sistema recomiende las mismas materias.

La comparación entre materias predichas y aprobadas se realiza usando dos métricas de recuperación de información: *Precision* y *Recall.*

Precision es la proporción de las recomendaciones que son relevantes, es decir, es la proporción de materias recomendadas que

fueron aprobadas por el estudiante. Si a un estudiante se le recomienda cuatro materias, y de ellas tres constan entre las aprobadas, entonces la precisión es de tres cuartos.

Recall es el porcentaje de ítems relevantes que aparecen en las recomendaciones, por lo tanto es la proporción de materias aprobadas que fueron recomendadas. Si un estudiante tiene cinco materias aprobadas y cuatro de ellas fueron recomendadas, entonces el *Recall* es de cuatro quintos.

Para realizar las recomendaciones por semestre y por estudiante se usaron cuatro tipos de configuraciones en el recomendador. Dos variaciones de métrica de similaridad: *Tanimoto, Loglikelihood*; y dos de filtrado de vecindad: *N Cercanos, Umbral*. La manera en la cual se graficaron los resultados fue mediante la variación de los valores de *Umbral y N Cercanos*. Los valores de *Umbral* a usarse en las pruebas son: 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, mientras que los valores de tamaño de vecindad para *N Cercanos* son: 1, 2, 4, 8, 16, 32, 64, 128.

Los resultados muestran el promedio de *Precision* y *Recall* de todos los estudiantes para cada semestre desde el primer término del año

2007 y variando en cada uno de ellos la configuración de métrica de similaridad y filtrado de vecindad.

4.4. Implementación del Recomendador de Materias

Para realizar la implementación del diseño propuesto, se inició la construcción del recomendador de materias tomando como base un recomendador genérico basado en el usuario y realizando adaptaciones a su código. Es por ello que la implementación se desarrolla sobre una librería de aprendizaje automático de código abierto. Adicionalmente, se escogió una base de datos compatible con la librería en donde se cargan los datos necesarios para realizar las recomendaciones. Las herramientas escogidas se presentan a continuación.

4.4.1. Herramientas a Utilizarse

Mahout

Mahout es una librería de autoaprendizaje de código abierto escrita en Java que implementa algoritmos de autoaprendizaje o inteligencia colectiva. Está enfocada en los motores de recomendaciones, agrupamiento y clasificación. Varias de las implementaciones usan librerías que permiten separar la carga de

procesamiento eficientemente entre distintas máquinas ya que está construida sobre el proyecto Hadoop de Apache. Su código está disponible para ser usado y adaptado para los desarrolladores de software que requieran implementar aplicaciones modernas e inteligentes.

Mahout empezó en el 2008 como un subproyecto del proyecto Lucene de Apache, de código abierto, que provee un motor de búsquedas del mismo nombre. Lucene provee implementaciones avanzadas de técnicas de búsqueda, minería de texto y recuperación de la información. En las ciencias computacionales estos conceptos están muy relacionados a los de agrupamiento y clasificación así que algunos desarrolladores de Lucene decidieron separar estas áreas de autoaprendizaje en otro subproyecto, Mahout. Este último luego absorbería otro proyecto de código abierto llamado Taste, un motor de recomendaciones de filtro colaborativo. La principal tarea de Mahout ha sido convertir estos algoritmos para que puedan funcionar con Hadoop. Actualmente Mahout está en su versión 0.5.

MySql

MySql es un motor de bases de datos relacional desarrollado por la

Sun Microsystems que actualmente pertenece a Oracle. Este motor de base de datos es compatible con numerosos lenguajes de programación y puede correr en diferentes plataformas. MySql ofrece licencias de tipo GNU GPL para uso de aplicaciones cuyos fines no sean comerciales pero no es un proyecto totalmente de código abierto, ya que la mayor parte de este se encuentra bajo el poder de una empresa privada.

Mahout fue la librería escogida para desarrollar este trabajo debido a que es una librería de código abierto con una comunidad activa, lo cual es ideal para el desarrollo de este trabajo. Además esta librería es constantemente sometida a mejoras en sus nuevas versiones. Otro punto favorable de Mahout, es que fue construido sobre Hadoop, un framework que soporta aplicaciones distribuidas, lo cual le permite a esta Liberia adaptarse a ambientes distribuidos de ser necesario. MySql es la base de datos que la comunidad de Mahout usa en su mayoría, ya que posee una buena compatibilidad con la librería.

Implementación del recomendador de materias en Mahout

El recomendador de materias basado en el historial académico es de cierta manera un nuevo tipo de recomendador basado en el

usuario. Para su implementación en Mahout se creó una nueva clase de recomendador que hereda de la clase GenericUserBasedRecommender, a esta clase se la denomino CourseRecommender. El Proceso de recomendación de la clase CourseRecommender se lo detalla en el diagrama de interacción de objetos que se muestra en la Figura 4-11. En este diagrama se aprecian las tres clases que intervienen en el proceso: CourseRecommender, TopItems y Estimator.

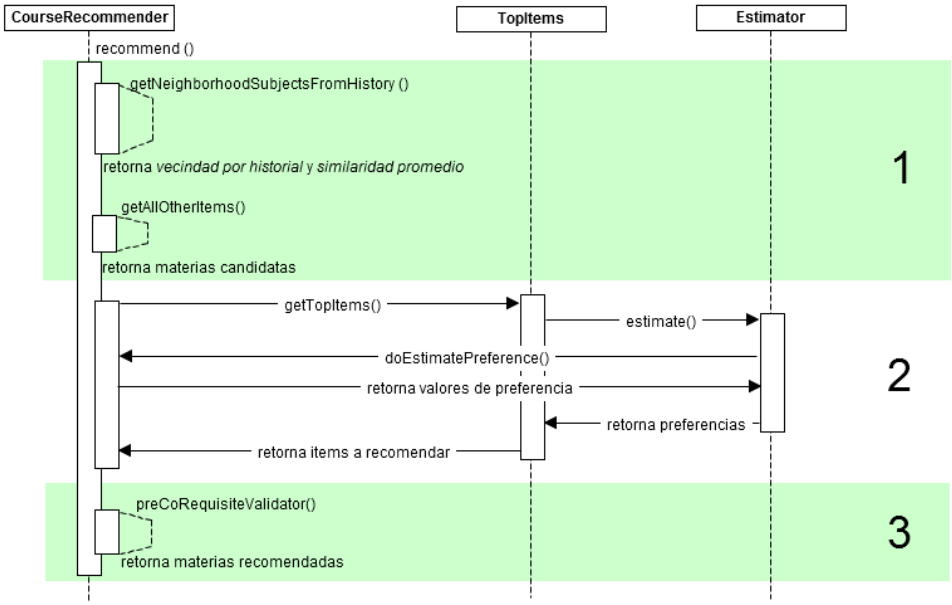


Figura 4-10 – Diagrama de interacción de objetos de la función recommend

En la Tabla IV, se explican las funciones de cada una de las secciones presentes en el diagrama de la Figura 4-11, junto con las etapas que se llevan a cabo en cada función. Las etapas del

proceso de recomendación fueron explicadas en la sección Modelo Lógico del Recomendador de Materias de este mismo capítulo.

Sección	**Función**	**Fases del Diseño Involucradas**
Sección 1	getNeighborhoodSubjectsFromHistory	- Obtención y cálculo de los *Modelos de Datos*$_m$, sus *vecindades*$_m$ y *similaridades*$_m$ - Obtención de la *Lista Materias Aprobadas* del estudiante - Construcción de la *Tabla de Frecuencia* - Construcción de la *Vecindad por Historial* - Calculo de las *Similaridades Promedio*
	getAllOtherItems	- Obtención de *Materias Candidatas*
Sección 2	doEstimatePreference	- Cálculo del *Valor de Preferencia* de las *Materias Candidatas*
Sección 3	preCoRequisiteValidator	- Aplicación de las reglas de validación a las materias recomendadas

Tabla IV – Tabla de clasificación de fases del proceso de recomendación por función que la ejecuta

Las clases TopItems y Estimator presentes en el diagrama de la figura 4-11, son propias de Mahout, por lo que la interacción de ellas en el proceso de recomendación lo impone el modelo del recomendador basado en el usuario de la librería.

La métrica de similaridad y el tipo de vecindad a usarse de manera definitiva en el recomendador serán definidas luego de analizar las pruebas. Por este motivo esta información se la detallara en el capítulo 5.

Problemas presentes en la implementación

Como se mencionó en el capítulo 2, en los recomendadores de filtro colaborativo se presenta el problema del usuario nuevo, por lo que en la implementación del recomendador de materias se realizó un proceso de recomendación para aquellos alumnos que piden recomendación para su primer semestre. A estos alumnos no se les puede obtener una *Vecindad por Historial,* ya que un no poseen un historial académico para realizar comparaciones, es por esto que las recomendaciones para estos alumnos no podrán ser efectuadas por el diseño presentado.

Como solución al problema del usuario nuevo, el recomendador de materias realiza la misma recomendación de materias a todos los

usuarios que consulten el primer semestre. La respuesta a esta consulta entrega como sugerencia las materias que la mayoría de estudiantes de la carrera aprobó en su primer semestre.

Uso del Recomendador

El recomendador puede ser añadido a la librería de manera permanente copiando su clase Java en la carpeta “impl” dentro del paquete Mahout. Para hacer uso del recomendador solo es necesario instanciar su clase en la aplicación Java que requiera su uso y cumplir con los requerimientos de instalación a continuación descritos.

Para este trabajo se construyó una interfaz web en donde el estudiante ingresa su matrícula y el número de semestre del que se requiere recomendación. Esta misma interfaz permite presentar las materias recomendadas.

Requerimientos de instalación del recomendador de materias

Para poder hacer uso del recomendador de materias básicamente se debe cumplir con los requerimientos necesarios para el funcionamiento de Mahout, los cuales comprenden:

- Tener instalado Java JDK 1.6 o una versión mayor
- Tener instalado el administrador de proyectos Maven
- Tener MySql como base de datos instalada
- Instalar y configurar Mahout

Los requerimientos de hardware de Mahout no han sido definidos de manera concreta ya que básicamente solo necesita que soporte Java. Sin embargo basada en la experiencia adquirida en el desarrollo de este trabajo se sugiere tener una máquina con un mínimo de 1 Gb de memoria RAM y una capacidad de disco duro de 15 Gb disponibles. Estos valores permiten una capacidad de almacenamiento y una velocidad de procesamiento razonable. Las características de la máquina deben ser definidos acorde a la cantidad de información que va a ser almacenada y procesada. Cabe recalcar que debido a que los recomendadores manejan grandes cantidades de datos pueden llegar a tener tiempos de procesamiento considerables. Mahout cuenta con la opción de ser ejecutado en ambientes distribuidos gracias a que está construido sobre Hadoop por lo que la idea de montar el recomendador en un ambiente distribuido sería la más acertada.

CAPÍTULO 5

5. PRUEBAS Y RESULTADOS.

5.1. Ejecución de las pruebas

Las pruebas se dividieron en dos grupos según su tipo de vecindad: *N Cercanos* y *Umbral*. Para cada grupo se varió el tamaño de vecindad y el valor de umbral respectivamente. Con cada una de las variaciones se realizaron cálculos usando las dos métricas de similaridad: *Loglikelihood* y *Tanimoto*. Y por cada combinación de métrica, tipo de vecindad y valor para el tipo de vecindad se calculó recomendaciones para todos los estudiantes del semestre a analizarse.

Para la realización de las pruebas se implementó la clase CourseRecommenderTester.java, cuyo código se encuentra en los anexos. Esta clase se encarga de obtener la lista de los estudiantes

que participan en las pruebas para luego pedir la recomendación de materias para cada uno de los estudiantes de la lista, calcularles sus valores de *Precision* y *Recall* y entregar el valor promedio de *Precision* y *Recall* para el semestre.

En esta clase se define el semestre al que se le realizará la prueba, la métrica de similaridad y el tipo de vecindad a usar. Para calcular las recomendaciones con los diferentes valores para los tipos de vecindad de forma paralela la clase CourseRecommenderTester utiliza hilos. A cada hilo se le asigna un valor de tamaño vecindad o de umbral, luego este se encarga de pedir la recomendación y escribir en un archivo los resultados para cada uno de los estudiantes y del semestre.

5.2. Análisis de los resultados

Promedio *Recall* con tipo de vecindad *N Cercanos*

En las Figuras 5-1, 5-2, 5-3, 5-4, 5-5, 5-6, 5-7 se muestran los promedios de *Recall* para los diferentes tamaños de vecindad *N Cercanos.* El eje vertical representa el promedio de *Recall* del semestre, mientras que el eje horizontal representa el tamaño de la vecindad. Como se puede observar todos los gráficos son similares en

la tendencia de sus curvas, y las dos métricas de similaridad: *Loglikelihood* y *Tanimoto*, incrementan su valor promedio conforme se va incrementando el tamaño de la vecindad, esto se debe a que las recomendaciones que usan un tamaño de vecindad pequeño se basan en un número de estudiantes tan reducido, que no se cuenta con una cantidad suficiente de ítems a analizar para la recomendación. Luego la curva crece de manera exponencial conforme la vecindad se agranda. Al pertenecer más estudiantes a *la vecindad por historial*, las materias recomendadas no solo se escogerán por haber sido aprobadas por los estudiantes con un grado de similaridad muy cercano al estudiante objetivo, sino que también se verán influenciadas por la tendencia de la mayoría de los estudiantes al aprobar la materia en el semestre, ya que mientras más estudiantes aprueban la materia el *valor de preferencia* de ésta va a aumentar, inclusive si los estudiantes que la aprobaron tuvieron un valor de similaridad normal o bajo.

Los valores óptimos de promedio de *Recall* para los semestres se alcanzaron con el tamaño de vecindad 64 y la métrica de similaridad *Loglikelihood.* En promedio el valor de *Recall* para todos los semestres fue de 0.83.

Cabe destacar que la métrica de similaridad Tanimoto con un tamaño de vecindad de 32, alcanza valores cercanos a *Loglikelihood* con tamaño 64 para todos los semestres. El promedio de Recall alcanzado por Tanimoto es 0.81, sin embargo para las comparaciones con los resultados de tipo de vecindad Umbral se usara *Loglikelihood* por tener un *Recall* superior.

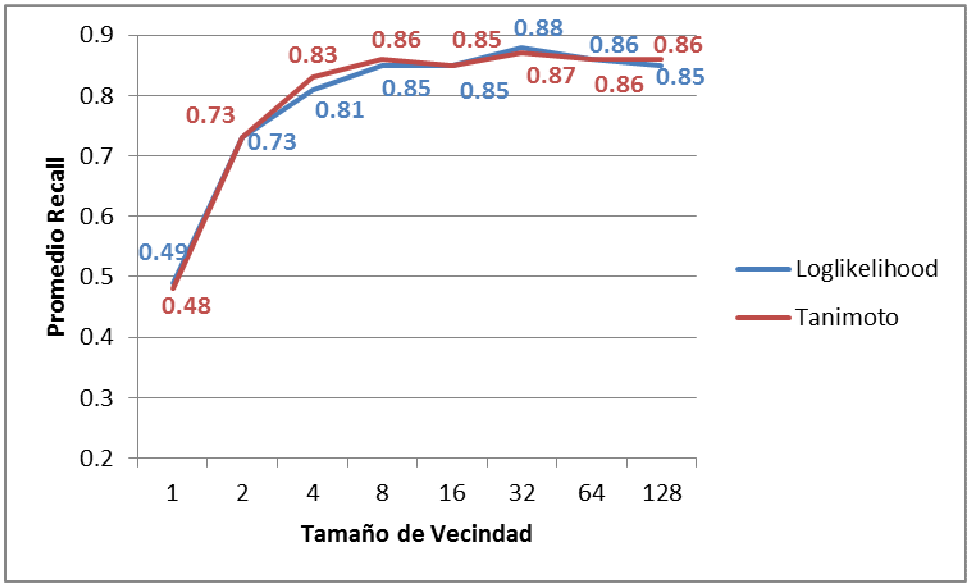


Figura 5-1— Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 2 para las métricas Loglikelihood y Tanimoto

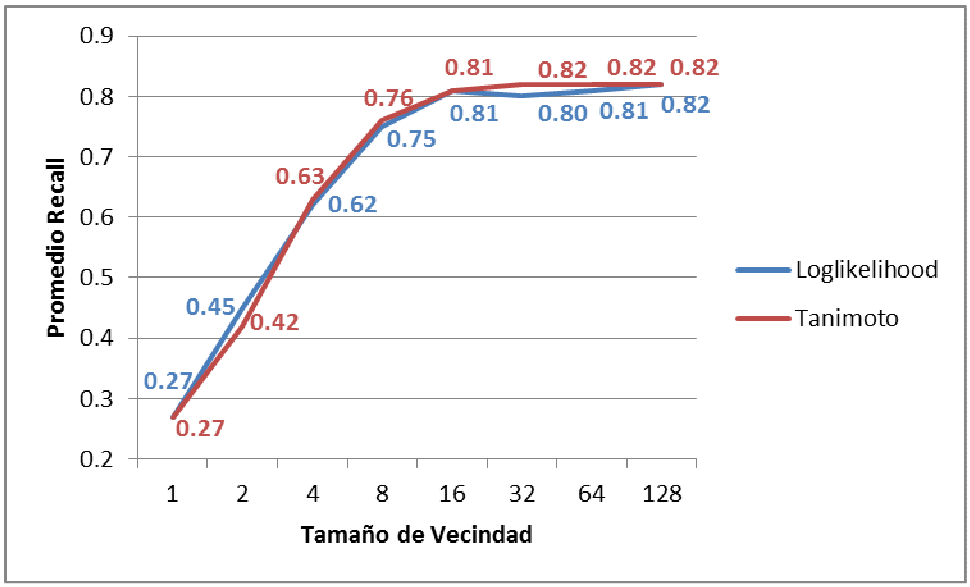


Figura 5-2 — Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 3 para las métricas Loglikelihood y Tanimoto

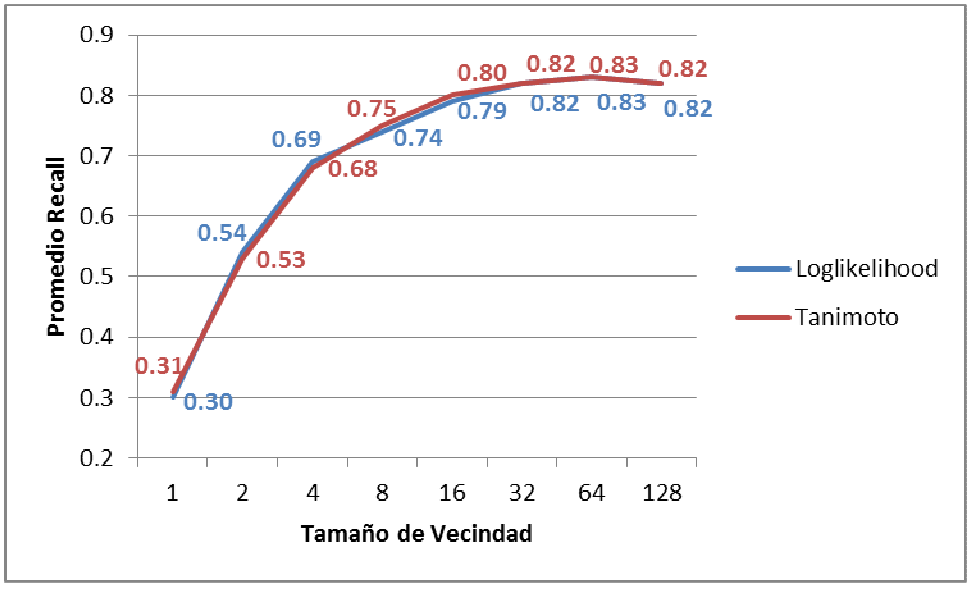


Figura 5-3 — Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 4 para las métricas Loglikelihood y Tanimoto

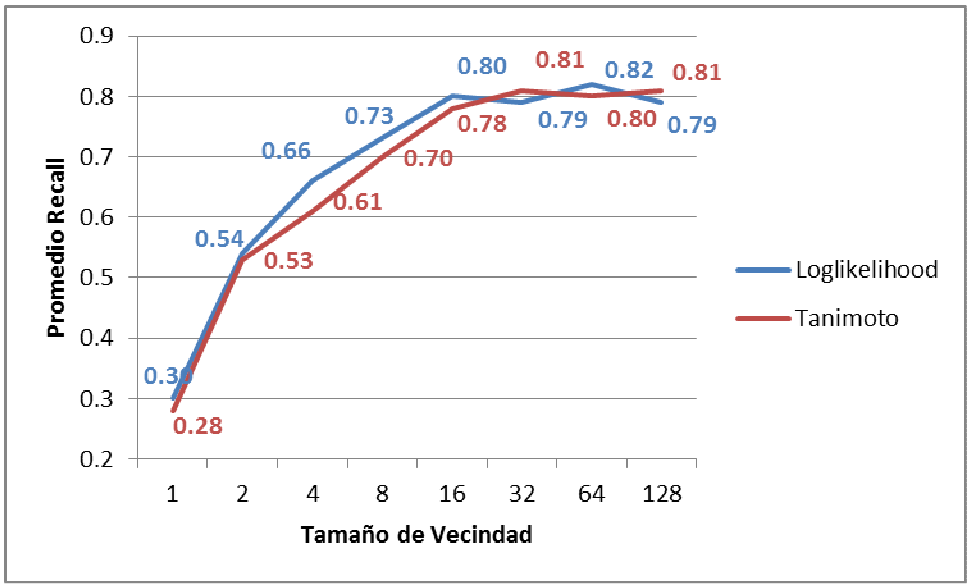


Figura 5-4 — Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 5 para las métricas Loglikelihood y Tanimoto

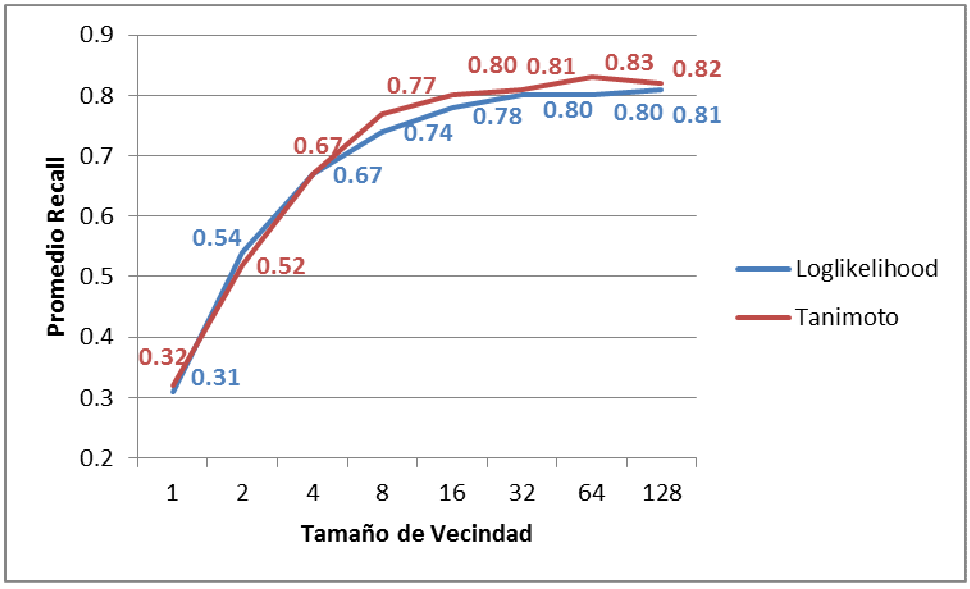


Figura 5-5 — Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 6 para las métricas Loglikelihood y Tanimoto

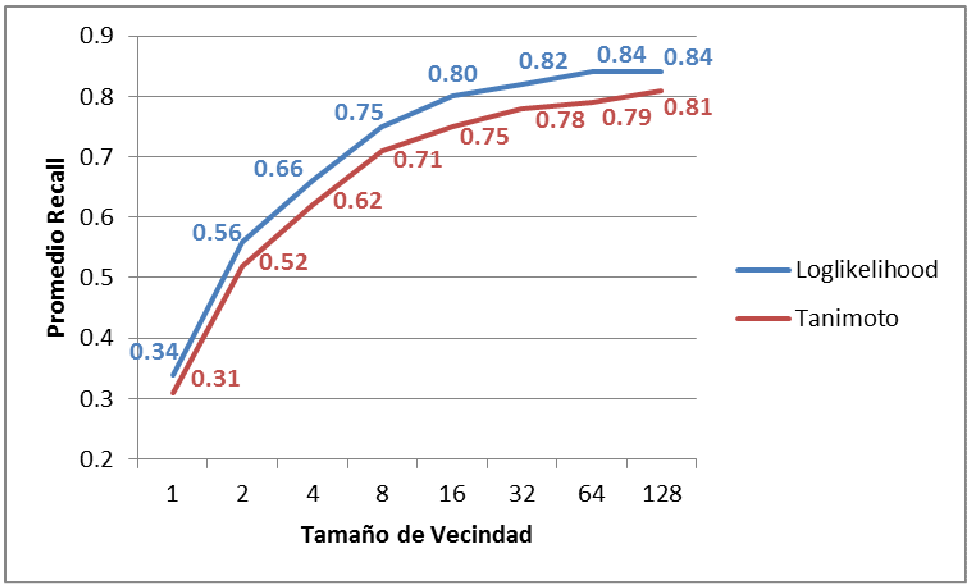


Figura 5-6 — Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 7 para las métricas Loglikelihood y Tanimoto

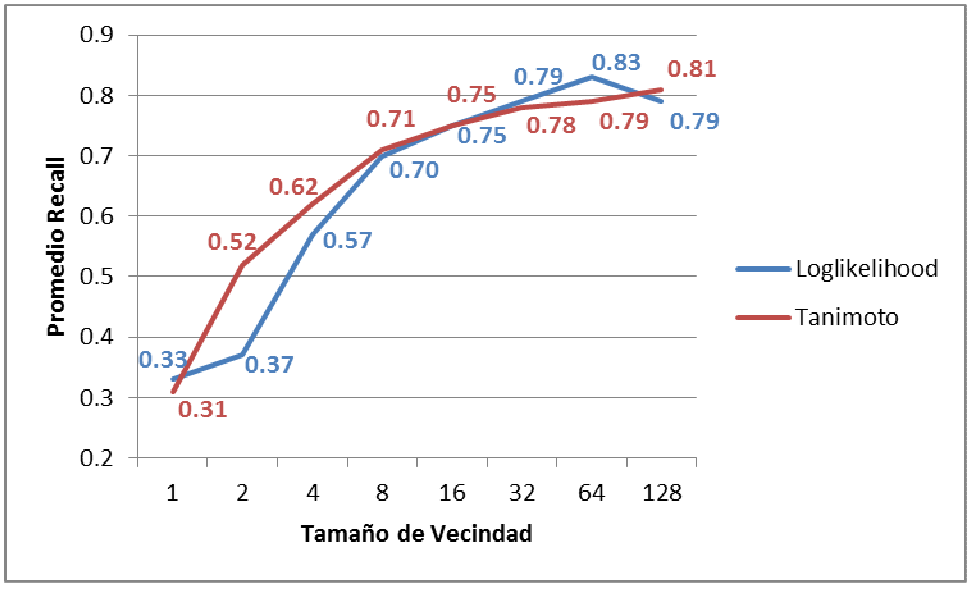


Figura 5-7 — Promedio recall obtenido para los diferentes tamaños de vecindad en el semestre 8 para las métricas Loglikelihood y Tanimoto

Promedio *Recall* con tipo de vecindad *Umbral*

En las Figuras 5-8, 5-9, 5-10, 5-11, 5-12, 5-13, 5-14 se muestran los promedios de *Recall* para los diferentes valores de umbral para las vecindades de tipo *Umbral.* El eje vertical representa el promedio de *Recall* del semestre, mientras que el eje horizontal representa el valor de umbral. Como se observa en las gráficas el valor de promedio de *Recall* decrece conforme se va aumentando el valor de umbral, esto se debe a que el incremento del valor del umbral implica que los usuarios de la *vecindad por historial* deben tener un historial académico muy similar al del estudiante objetivo, es decir que debieron haber aprobado las materias de igual manera para un número mayor de semestres, lo que provoca que el número de estudiantes en la *vecindad por historial* sea cada vez menor.

Se puede observar en las gráficas que a medida que avanza el número de semestre las curvas empiezan a decrecer y deja de ser una línea recta como al inicio. Esto se debe a que a medida que avanzamos en el flujo de una carrera se incrementa el número de caminos y formas de orden en las que se pueden aprobar las materias, teniendo así grupos cada vez más pequeños de usuarios que han recorrido su flujo de la misma manera. Esto ocasiona que la mayoría de estudiantes en la tabla de frecuencia tenga valor de

frecuencia cada vez menores, y al realizarse el filtro por umbral las vecindades contendrán menos estudiantes conforme aumente el semestre y se aumente el valor de umbral. Como se mencionó anteriormente, las recomendaciones basadas en vecindades pequeñas habitualmente no alcanzan valores de acierto altos.

Para el tipo de vecindad Umbral los mejores promedios de *Recall* para la mayoría de semestres se obtuvieron con *Loglikelihood* con valor de umbral 0.4 y Tanimoto con valor 0.3, ambos tuvieron un promedio de Recall de 0.79 para todos los semestres. Sin embargo Tanimoto posee diferencias de valor considerables entre sus valores máximos y mínimos obtenidos en cada semestre, a diferencia de *Loglikelhood* que mantiene valores estables para todos los semestres. Por este motivo para las comparaciones con los valores de las vecindades de tipo N Cercanos se usó a *Loglikelihood* con valor de umbral 0.4 como la combinación óptima.

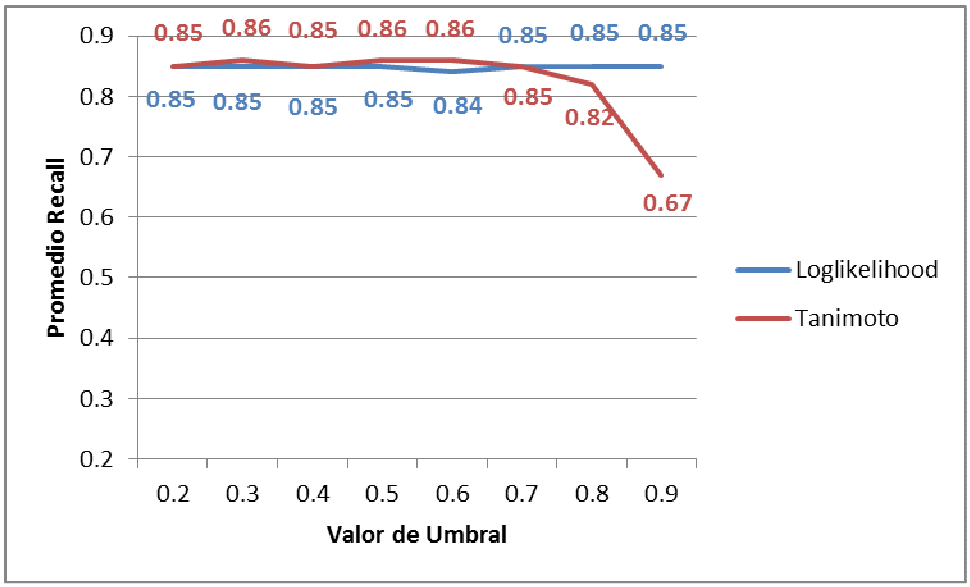


Figura 5-8 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 2 para las métricas Loglikelihood y Tanimoto

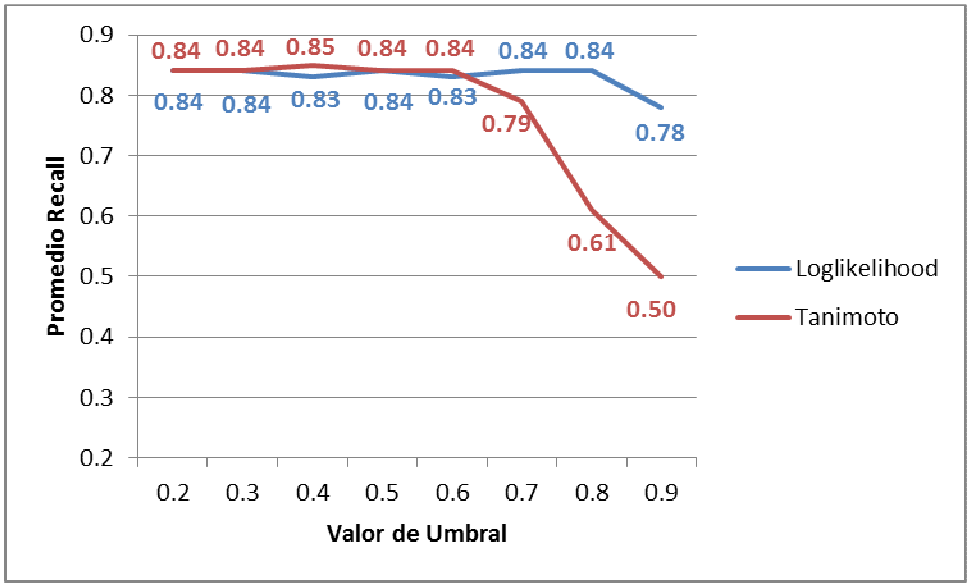


Figura 5-9 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 3 para las métricas Loglikelihood y Tanimoto

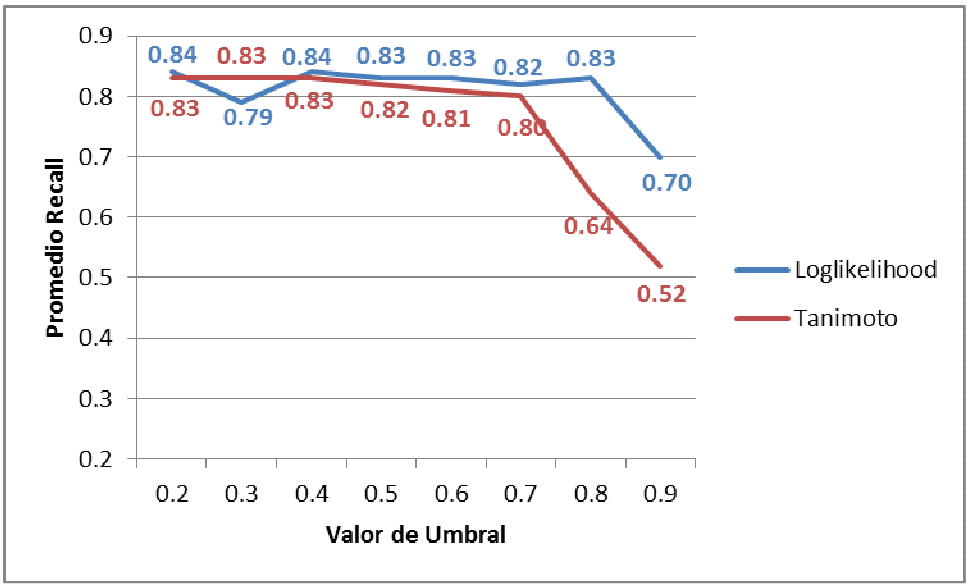


Figura 5- 10 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 4 para las métricas Loglikelihood y Tanimoto

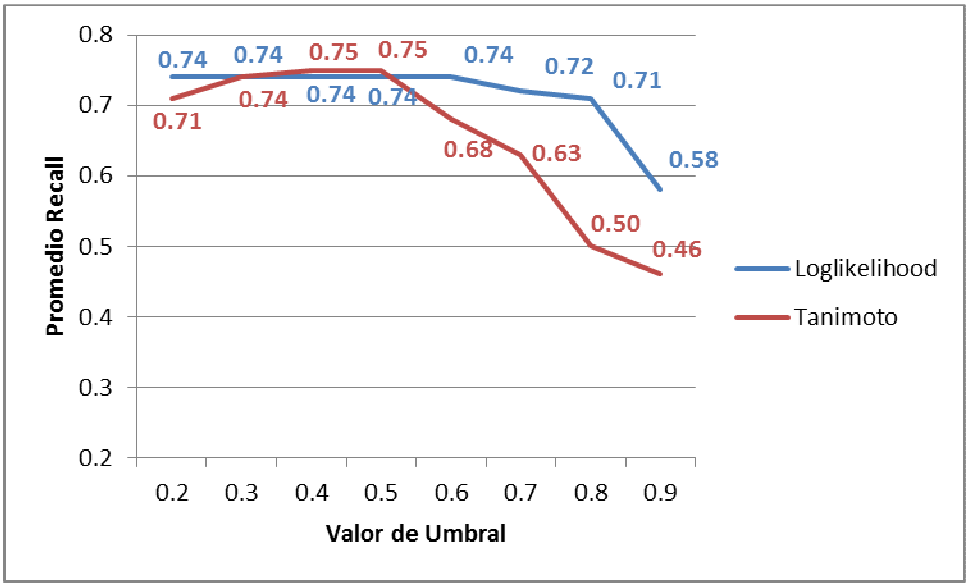


Figura 5-11 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 5 para las métricas Loglikelihood y Tanimoto

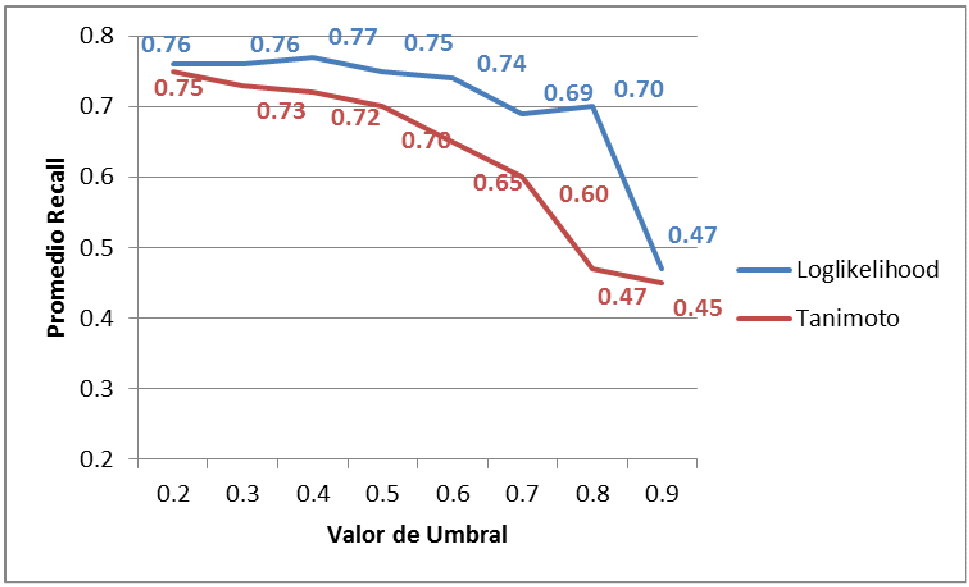


Figura 5-12 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 6 para las métricas Loglikelihood y Tanimoto

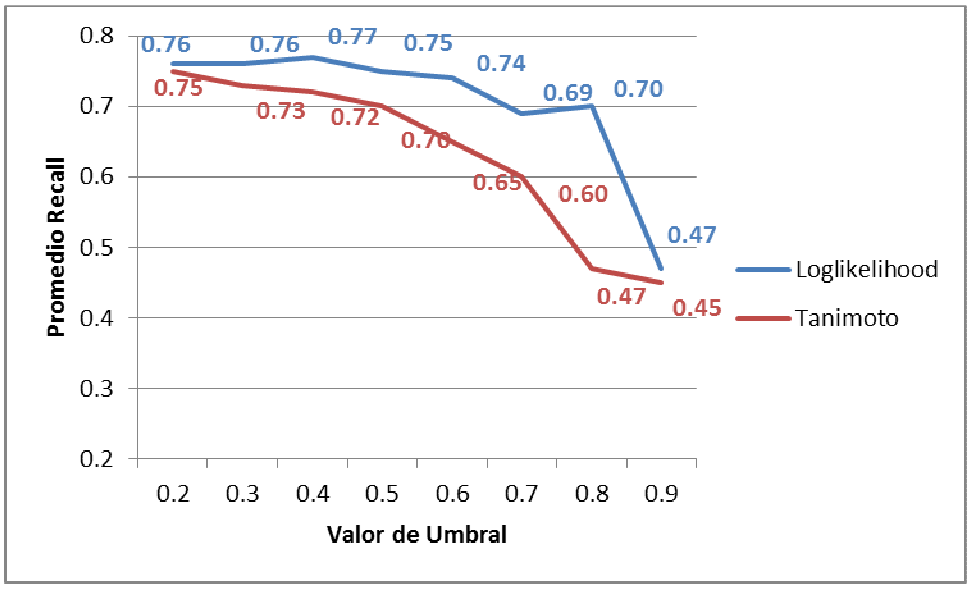


Figura 5-13 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 7 para las métricas Loglikelihood y Tanimoto

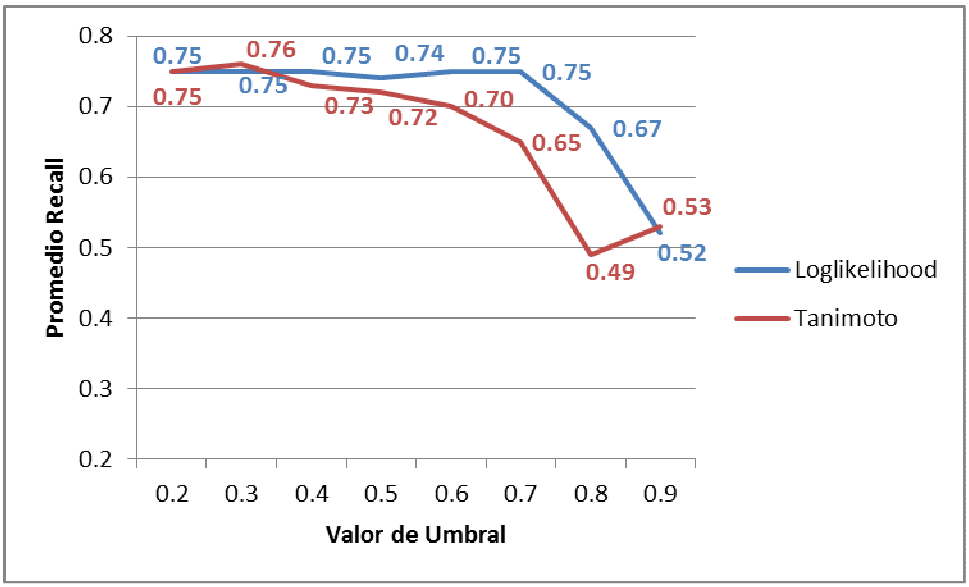


Figura 5-14 — Promedio recall obtenido para los diferentes valores de umbral en el semestre 8 para las métricas Loglikelihood y Tanimoto

Comparación de las los valores obtenidos en los diferentes tipos de vecindad

En los resultados presentados en los gráficos anteriores, se mostró que la métrica de similaridad *Loglikelihood* obtiene mejores resultados que Tanimoto, tanto para las vecindades filtradas por *N Cercanos* como para las filtradas por *Umbral.* En la Figura 5-15 se grafican los valores de *Recall* obtenidos en todos los semestres para las dos combinaciones seleccionadas: *Loglikelihood* valor 64 para *N Cercanos*, *Loglikelihood* valor 0.4 para *Umbral*. Con el tipo de vecindad *N Cercanos* los valores de acierto en las recomendaciones fueron similares para todos los semestres, manteniendo una tendencia

más estable que usando el tipo de vecindad *Umbral,* en donde hubo una mayor variación en los valores de acierto. Se determinó que *Loglikelihood* con una vecindad de tipo N Cercanos de tamaño 64 es la mejor combinación para el recomendador de materias, con un promedio de acierto de 0.83 en las recomendaciones respecto a las materias que los estudiantes aprobaron en la realidad, este valor de acierto se lo obtuvo calculando un promedio de los valores de *Recall* de todos los semestres.

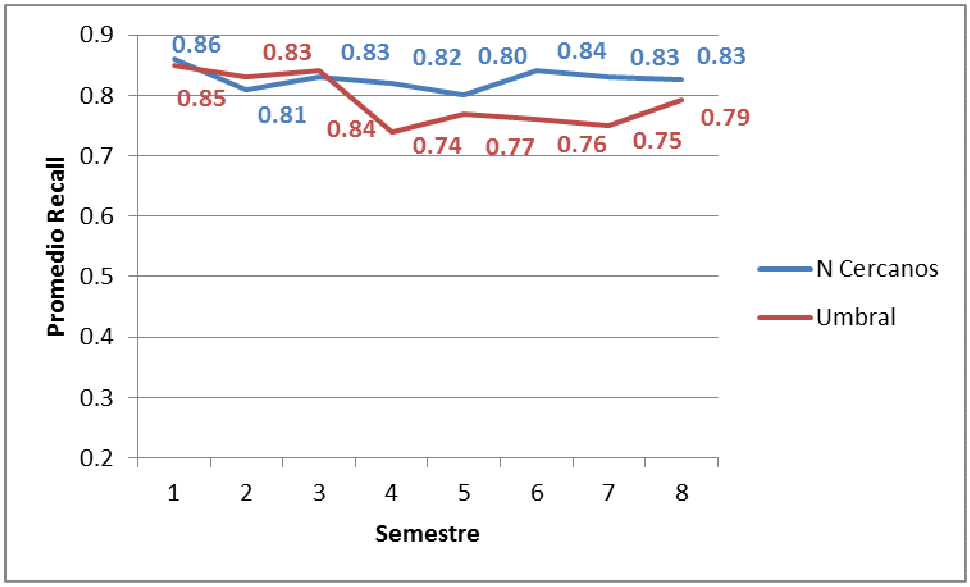


Figura 5- 15 -- Resultados de Loglikelihood con los tipos de vecindad N Cercanos y Umbral para todos los semestres

Valores de *Precision* en *Loglikelihood* con *N Cercanos* de tamaño 64

Los valores de *Precision* permiten identificar grupos de estudiantes por cantidad de acierto recibidos, así como la proporción de estos grupos. En las Figuras 5-16, 5-17, 5-18, 5-19, 5-20, 5-21, 5-22 se muestran estos grupos de estudiantes según la cantidad de aciertos recibida en cada uno de los semestres. El eje x representa la cantidad de materias acertadas y el eje y representa el número de estudiantes que obtuvo esa cantidad de acierto.

Se debe tomar en consideración que los estudiantes en las pruebas no aprobaron la misma cantidad de materias en todos los semestres, y que esto afecta a los valores de *Precision* obtenidos. Por ejemplo si se recomendó 6 materias pero el estudiante solo aprobó 4 materias y las 4 constan entre las 6 recomendadas, el valor de *Precision* es 4/6, a pesar de que el valor de acierto o *Recall* es de 1.0.

De manera general se observa que la mayor parte de los estudiantes obtuvieron aciertos en 3, 4 y 5 materias de las 6 recomendadas.

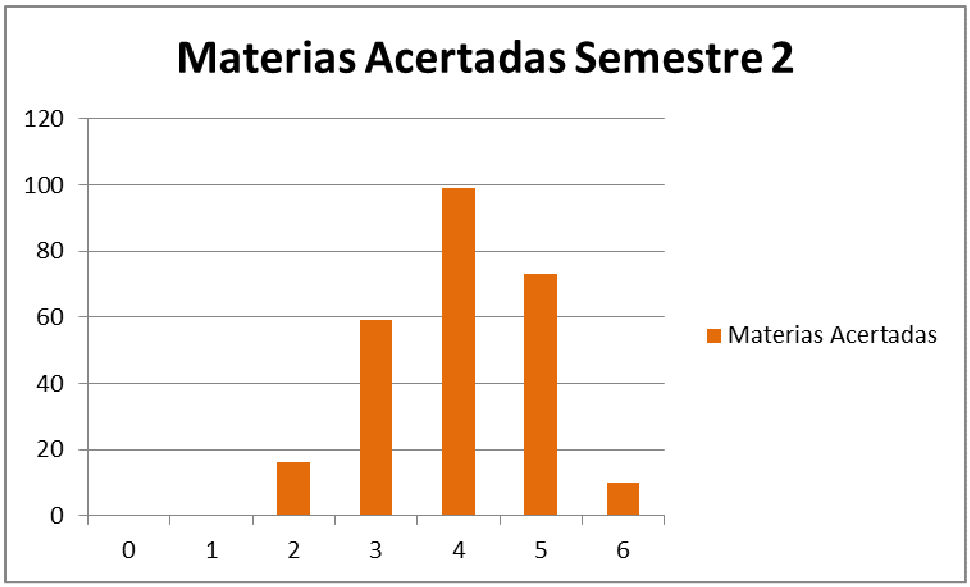


Figura 5-16 -- Valores de *precision* obtenidos para el semestre 2

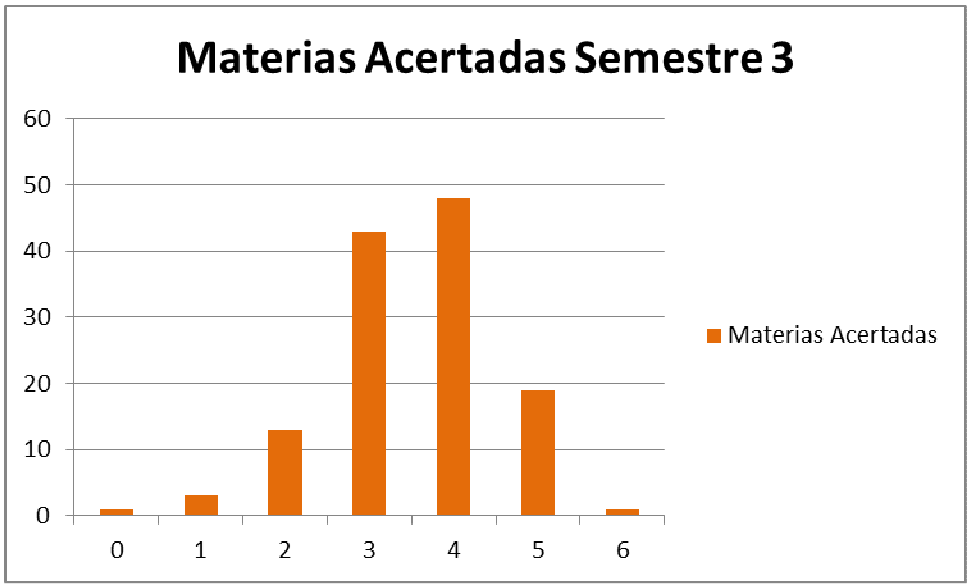


Figura 5-17 -- Valores de *precision* obtenidos para el semestre 3

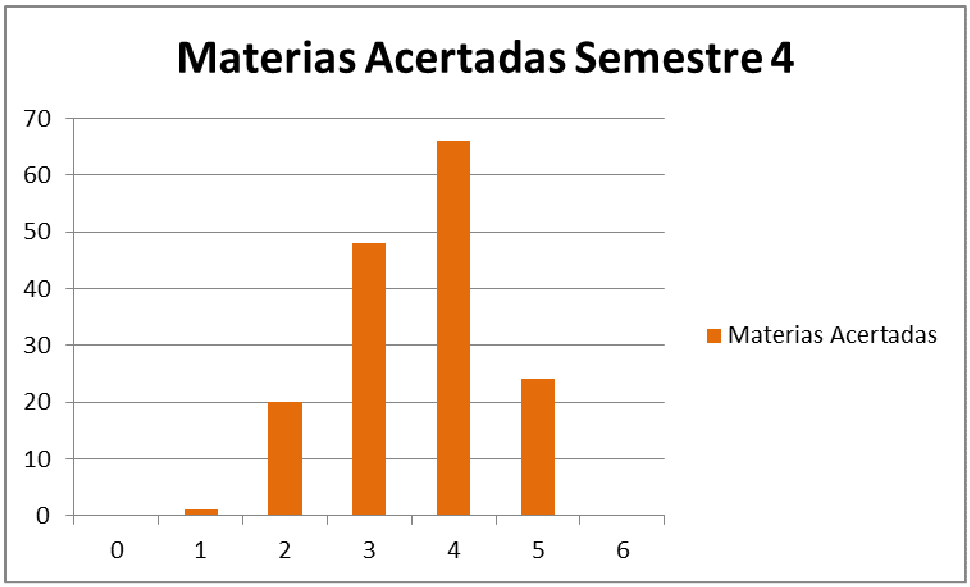


Figura 5-18 -- Valores de *precision* obtenidos para el semestre 4

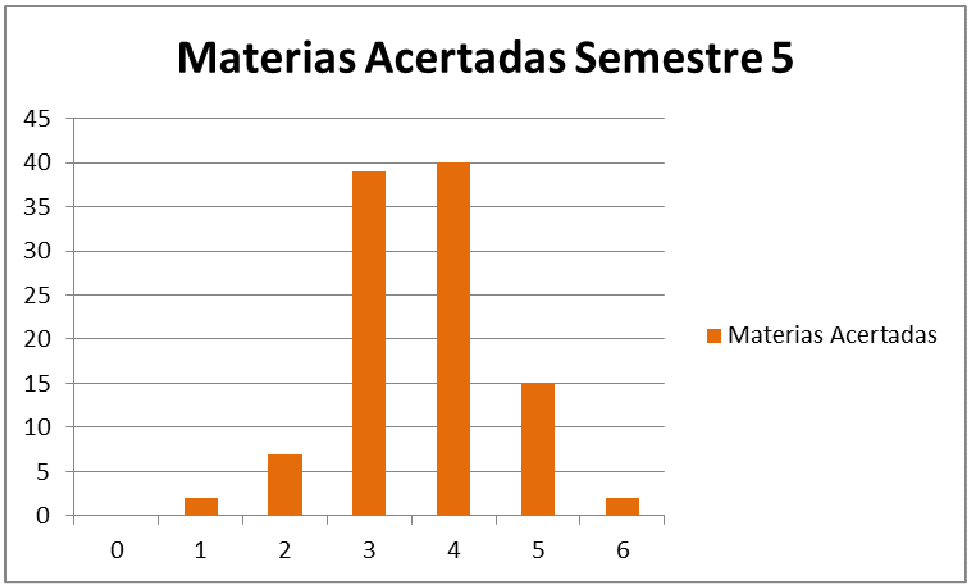


Figura 5-19 -- Valores de *precision* obtenidos para el semestre 5

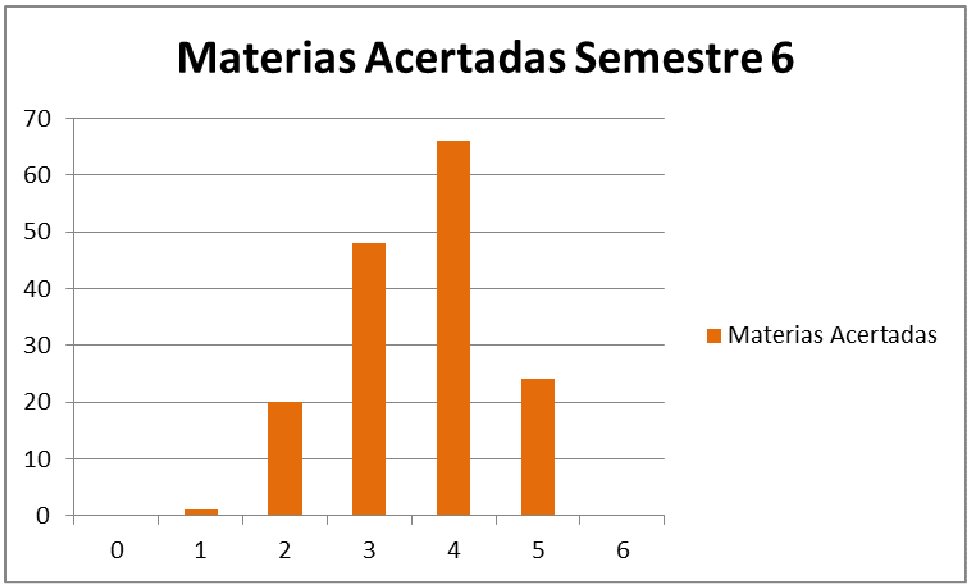


Figura 5-20 -- Valores de *precision* obtenidos para el semestre 6

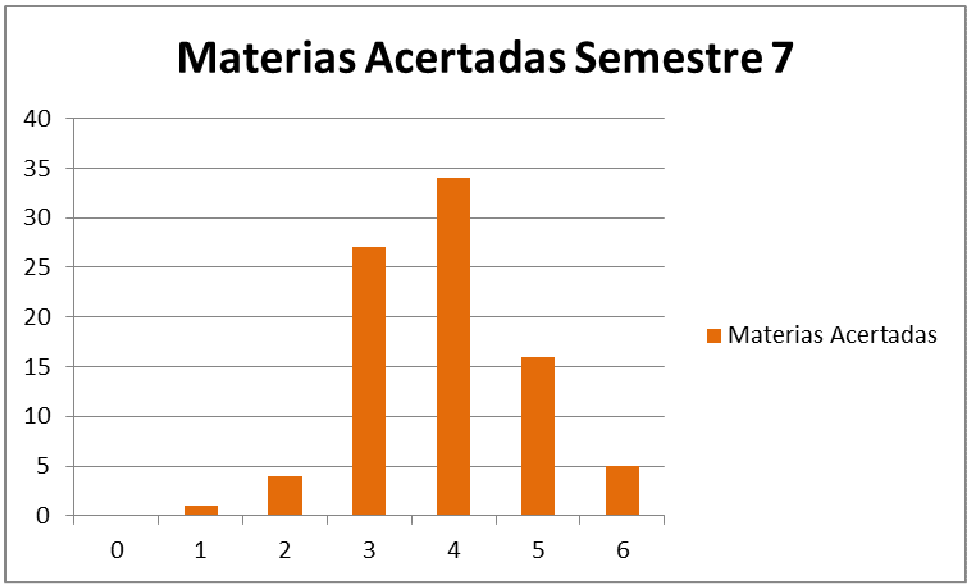


Figura 5-21 -- Valores de *precision* obtenidos para el semestre 7

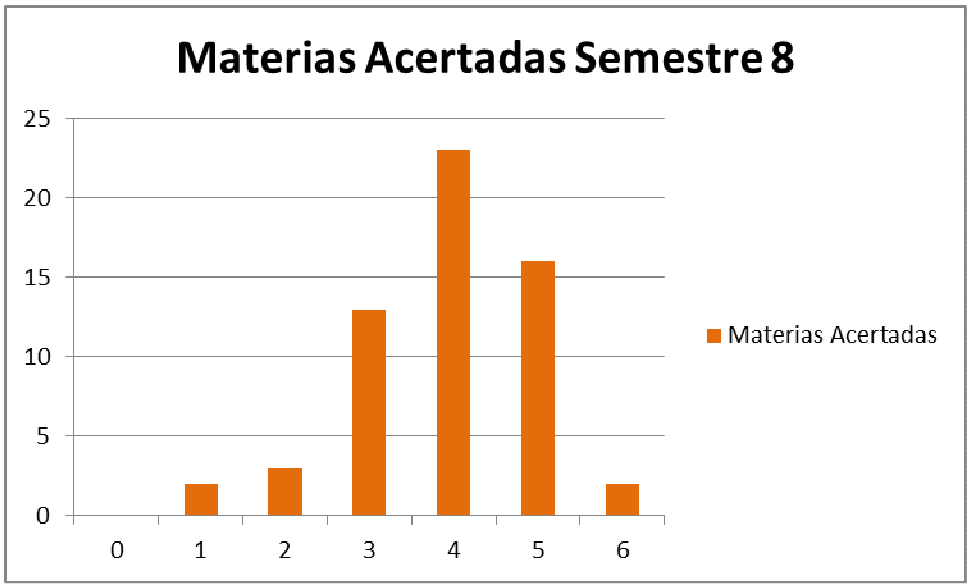


Figura 5-22 -- Valores de *precision* obtenidos para el semestre 8

CONCLUSIONES

Las conclusiones son:

1. La adaptación del recomendador se realizó de manera exitosa. Las pruebas demostraron que se consiguió un valor de acierto de 0.83 en las recomendaciones con respecto a las materias que el estudiante decidió tomar en la realidad. Este valor se lo consiguió usando la métrica de similaridad *Loglikelihood* y en tipo de vecindad *N Cercanos* con un tamaño igual a 64. Los estudiantes que pertenecen a una vecindad de este tamaño para la métrica mencionada tienen en promedio un valor de similaridad de 0.57 con una desviación estándar de 0.11. Sesenta y cuatro representa un tamaño de vecindad óptimo dentro de los valores asignados en las pruebas. Futuros estudios podrán determinar el valor óptimo del tamaño de vecindad para el recomendador.

2. La métrica de similaridad *Loglikelihood* demostró darle un comportamiento más certero y uniforme al recomendador en cuanto a sus predicciones. En contraparte *Tanimoto* alcanzo altos valores en algunas de las pruebas realizadas, pero el recomendador en general

no siempre entregó recomendaciones altamente acertadas en todos los semestres.

3. Con respecto al tipo de vecindad, el recomendador no sostuvo buenos valores con el uso de vecindad *Umbral,* esto se dio a que este tipo de filtrado depende directamente de la cantidad de estudiantes que poseen un historial académico similar en comparación al estudiante objetivo. Los mejores valores de acierto se dieron alrededor de los umbrales 0.3 y 0.4. El motivo se debe a que los estudiantes que pasan a la *vecindad por historial* con estos valores de umbral son aquellos que presentaron una similitud alta o media en su historial académico con respecto al estudiante objetivo. Es decir, el grupo de estudiantes no es tan numeroso como para tomar en cuenta materias que no reflejan realmente las tendencias del estudiante, y tampoco es tan reducido como para basar la recomendación con pocas opciones de donde obtener las materias candidatas. Por ejemplo, para el umbral 0,9 el grado de similaridad es muy estricto, por lo que no hay muchos estudiantes en los cuales basar la comparación y se debe tomar en consideración el hecho de que si dos estudiantes han tomado sus materias de manera similar en todos sus semestres pasados no asegura que sus decisiones serán idénticas en el futuro.

4. El éxito del recomendador de materias radica en encontrar patrones de grupos. Debido a esto, el filtrado de vecindad por medio de *N Cercanos* le permitió al recomendador alcanzar los valores de acierto más altos presentes en las pruebas. La curva de los resultados con este tipo de vecindad es creciente, y a pesar de que se inicia con valores bajos debido a los tamaños de vecindad reducidos, alcanza una estabilidad para todos los semestres con el tamaño de vecindad 64. Con este tamaño se ajusta de una manera más eficiente la cantidad de estudiantes con similaridad alta y mediana que formaran parte de la *vecindad por historial.* Con los tamaños de vecindad mayores a 64 empiezan a aparecer caídas en los valores de acierto en algunos semestres, a pesar de que las caídas no son de gran magnitud, se concluye de que con un tamaño de vecindad excesivamente grande ingresaran estudiantes que introduzcan “ruido” en las recomendaciones, ya que con estas grandes vecindades el recomendador empieza a entregar como recomendación las materias que la mayoría de estudiantes aprobó en el semestre, sin mayor distinción.

5. El tiempo que toma realizar una recomendación a un estudiante es de 10 segundos aproximadamente. Este tiempo se dio usando un hardware con 2GB de memoria RAM y un procesador de 1.2 Ghz. De

manera general se concluye que las recomendaciones deben ser calculadas de manera previa a la consulta de los estudiantes, ya que alcanzar buenos tiempos de respuesta en consultas procesadas en tiempo real implica altos costos en hardware.

6. Las recomendaciones entregadas por el sistema son buenas desde el punto vista de que en las pruebas se demostró que las materias recomendadas a los estudiantes fueron elegidas por estos en los registros reales y aprobadas de manera exitosa con un alto grado de acierto. Sin embargo las recomendaciones no necesariamente son las ideales, ya se debe considerar que las decisiones que tomaron los estudiantes en los que se basa la recomendación, no necesariamente involucra una combinación eficiente en la manera de aprobar el flujo de la carrera.

RECOMENDACIONES

Las recomendaciones son:

1. Se propone realizar con un grupo de estudiantes pruebas que midan la efectividad de las recomendaciones en base a decisiones que tomen ellos luego de haber consultado al recomendador. Asimismo se debe encontrar el tamaño de vecindad óptimo dentro del rango [32,128).

2. Como trabajo a futuro se puede considerar usar este trabajo como la base para la implementación de una aplicación al alcance del uso de los estudiantes además de extender el recomendador a otras carreras.

3. Los cálculos de las recomendaciones deben hacerse antes de empezar el periodo de registros de los estudiantes, de este modo al presentar las recomendaciones solo deberá hacerse una consulta de las mismas, y no un cálculo en tiempo real que implica altos costos de hardware.

4. Si en las recomendaciones deben estar presentes reglas de pre-requisitos se recomienda que los datos que se ingresen al recomendador tengan presente estas reglas de pre-requisitos, de esta manera las recomendaciones seguirán el patrón validado de los estudiantes anteriores.

5. Si en algún momento de desea realizar recomendaciones en tiempo real por que es necesario usar información recién ingresada por el estudiante, se debe trasladar el recomendador de materias a un ambiente distribuido para alcanzar mejores tiempos de respuesta.

6. Se debe considerar añadir al proceso de recomendación otros factores de criterio de elección de materias como: las notas de los estudiantes, valoraciones de los profesores en el CENACAD, etc.

ANEXOS

ANEXO A

Función getNeightborsSubjectsFromHistory

```
private  FastMap< Long , Double > [] getNeightborsSubjectsFromHistory(Long
userID, int semester,Double threshold, String stringSimilarity,String
neighborType)throws Exception{
	ArrayList<UserSimilarity> similarities = new
	ArrayList<UserSimilarity>();
	ArrayList<String> filtered_neighbors= new ArrayList<String>();
	HashMap <String,Integer>table_frequency = new
	HashMap<String,Integer>();
	UserSimilarity similarity=null;
	FastMap<Long,Double> closerNeighborSimilarities;
	MysqlDataSource dataSource = new MysqlDataSource();
	dataSource.setServerName("localhost");
	dataSource.setUser("root");
	dataSource.setPassword("linux");
	dataSource.setPort(3306);
	dataSource.setDatabaseName("mahout2007");
	long timeIni = System.currentTimeMillis();
	FastIDSet academicHistory = new FastIDSet();
	FastMap<Long,Double> userAcademicHistory= new FastMap();

	for(int s=1;s<semester;s++){
	JDBCDataModel model = new
	MySQLBooleanPrefJDBCDataModel(dataSource,"semestre"+s,"id_estudiante
	","id_materia");

	//Lleno academicHistory con las materias de cada semestre del
	estudiante recomendado
	FastIDSet f=model.getItemIDsFromUser(userID);
	if(f==null){
	System.out.println("es que es null");
	}else{
	academicHistory.addAll(f);}
	//Se escoge la metrica de similaridad a usar
	if(stringSimilarity.equals("tanimoto")){
	similarity = new TanimotoCoefficientSimilarity(model);
	}else if(stringSimilarity.equals("loglikelihood")){
	similarity = new LogLikelihoodSimilarity(model);
	}
	//se guardan todas las similaridades de ese model entro todos los
	usuarios que pertenecen en el
	similarities.add(similarity);
	//obtenemos la vecindad
	UserNeighborhood neighborhood=null;
	if(neighborType.equals("nearest")){
	neighborhood = new NearestNUserNeighborhood((int)
	Math.floor(threshold), similarity, model);
```

```
}else if(neighborType.equals("threshold")){
neighborhood = new ThresholdUserNeighborhood(threshold, similarity,
model);
}
//obtenemos la vecindad del estudiante para el semestre s
long[] neighbors = neighborhood.getUserNeighborhood(userID);
//Llenamos la tabla de frecuencia
for (int i=0; i< neighbors.length; i++){
addToFrequencyTable(table_frequency,neighbors[i]);
}
}
//Sacar el mayor numero de ocurrencias y que ese sea el frecuency
grade
Iterator i=table_frequency.values().iterator();
Integer frequency=0;
Integer frequencyValue=0;
while(i.hasNext()){
Integer tmp=(Integer)i.next();
if(tmp>frequency){
frequency=tmp;
}
}
//Aqui se crea la Vecindad por Historial
if(neighborType.equals("nearest")){
filtered_neighbors=filterOfNeighborByNumberPersons(table_frequency,
Integer.parseInt(threshold+""),frequency);
}else if(neighborType.equals("threshold")){
frequencyValue=(int)Math.round(threshold*(frequency));
filtered_neighbors=filterOfNeighborByFrequency(table_frequency,
frequencyValue);
}
//Se crea la similaridad promedio
closerNeighborSimilarities =
similarityOfNeighbors(similarities,userID,filtered_neighbors,
semester, table_frequency);
closerNeighborSimilarities.rehash();
//Pasar el fastIdSet to Fast Map
long [] userSubjects=academicHistory.toArray();
for(int k=0;k<userSubjects.length;k++){
long tmp=userSubjects[k];
userAcademicHistory.put(tmp,Double.parseDouble(tmp+""));
}
userAcademicHistory.rehash();
FastMap [] fastMapSet = new FastMap[2];
fastMapSet[0]=closerNeighborSimilarities;
fastMapSet[1]=userAcademicHistory;
return fastMapSet;
}
```

ANEXO B

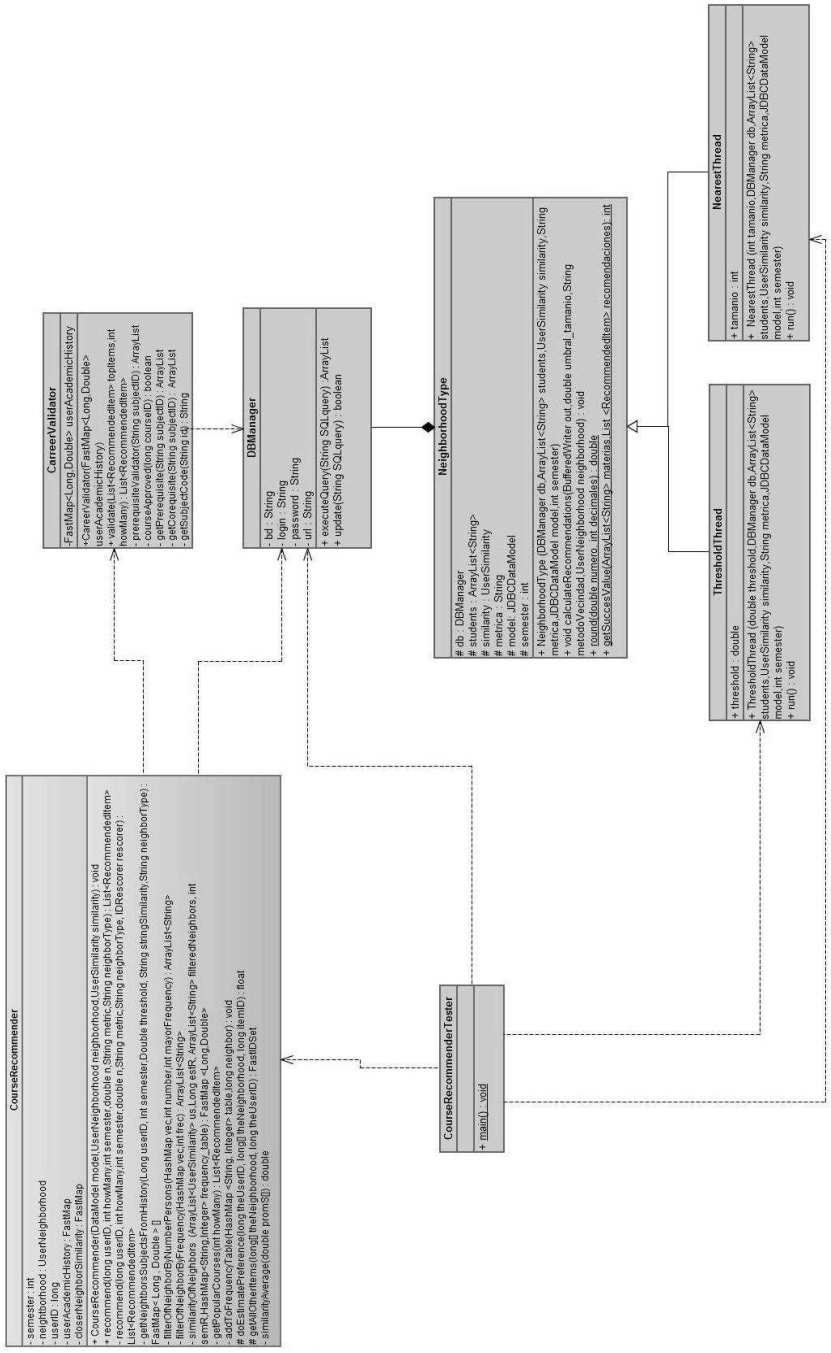
Clase CourseRecommenderTester

```
/*CLASE: CourseRecommenderTester
 *DESCRIPCION: Clase encargada de realizar las pruebas al
 * CourseRecommender y crear archivos con los resultados de las pruebas
 */
public class CourseRecommenderTester {
  public static void main(String[] args) throws Exception {
   //Semestre para el que se haran los calculos
    int semester=6;
    //Metrica de similaridad
    String metrica="loglikelihood";
    //Metodo de obtencion de vecinos
    String neighboorType="nearest";
    DBManager db = new DBManager();
    //Cargar la lista de estudiantes
    ArrayList<String> students=db.executeQuery("select distinct
    id_estudiante,count(id_estudiante)as r from semestre"+semester+" group
    by id_estudiante having r>3");

   //Cargar datamodel
   MysqlDataSource dataSource = new MysqlDataSource();
   dataSource.setServerName("localhost");
   dataSource.setUser("root");
   dataSource.setPassword("linux");
   dataSource.setPort(3306);
   dataSource.setDatabaseName("mahout2007");

   long timeIni = System.currentTimeMillis();

   //Crear modelo de datos de una base de MySql
   JDBCDataModel model = new
   MySQLBooleanPrefJDBCDataModel(dataSource,"semestre"+semester,"id_estudi
   ante","id_materia");

   //Eleccion de la metrica de similaridad
   UserSimilarity similarity=null;
   if(metrica.equals("tanimoto")){
   similarity = new TanimotoCoefficientSimilarity(model);
   }else if(metrica.equals("loglikelihood")){
   similarity = new LogLikelihoodSimilarity(model);
   }

   //Eleccion del metodo para hallar vecinos
   if(neighboorType.equals("nearest")){
   int n=0;
   for (int i=0;i<8;i++){
```

```
n=(int)Math.pow(2,i);
new Thread (new NearestThread (n, db, students, similarity, metrica,
model, semester)).start();
}
}else if(neighboorType.equals("threshold")){
double k=0.2;
while(k<=0.9){
new Thread (new ThresholdThread (round(k,2), db, students, similarity,
metrica, model, semester)).start();
k=k+0.1;
}
}
}
public static double round(double numero, int decimales){
return Math.round(numero*Math.pow(10,decimales))/Math.pow(10,
decimales);
}
}
```

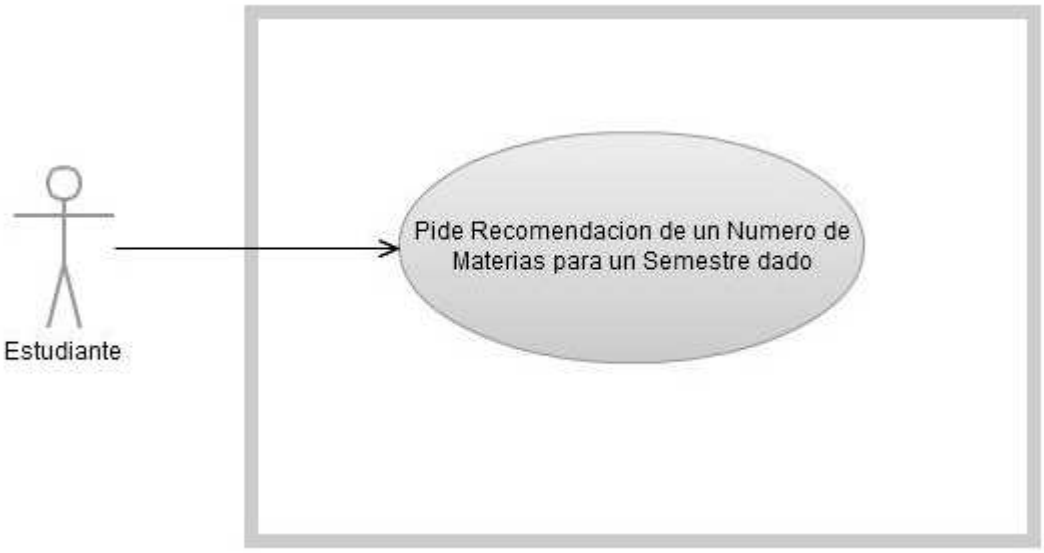
ANEXO C

Diagrama de Clases



ANEXO D

Casos de Uso



ANEXO E

Interfaz gráfica

BIBLIOGRAFÍA

[1] O' Mahonny, Michael P. , Smith, Barry. , A Recommender System for On-line Course Enrolment, Proceedings of the 2007 ACM conference on Recommender systems, 2007, 133-136.

[2] Smyth, B., Case-based recommendation, The adaptive web, Springer, 2007, 342-376.

[3] Bendakir N. , Aimeur E., Using association rules for course recommendation, In Proceedings of the AAAI Workshop on Educational Data Mining, 2006, 31–40

[4] Farzan R, Brusilovsky P., Social navigation support in a course recommender system. In Proceedings of the 4th International Conference on Adaptive Hypermedia and Apadtive Web-based Systems, 2006, 91–100

[5] Castellano E.J., Martínez L., A web-decision support system based on collaborative Filtering for academic orientation. case study of the spanish secondary school, Journal of Universal Computer Science, 2009.

[6] Satman, Alag. Collective Intelligence, Manning Publications Co, 2008.

[7] Breese John S., Heckerman David, Kadie Carl, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, Microsoft Research,Microsoft Corporation, 1988.

[8]Weisstein, Eric W., Correlation Coefficient, http://mathworld.wolfram.com/CorrelationCoefficient.html, MathWorld, Wolfram Research, fecha de consulta mayo 2011.

[9] Real, R., Vargas, J.M., The probabilistic basis of Jaccard's index of similarity. Systematic Biology, 2006, 380-385.

[10]Dunning, T. Surprise and Coincidence, http://tdunning.blogspot.com/2008/03/surprise-and-coincidence.html, 2008.

[11] Adomavicius, G., Tuzhilin, A.

Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 17, IEEE Computer Society, 2005.

[12] Owen S., Anil R., Dunning T., Friedman E., Mahout in Action, Manning Publications Co., 2009, 4-65.