UNIVERSIDAD NACIONAL DE CÓRDOBA

Aprendizaje Activo para mejorar el Arranque en Frío de Sistemas de Recomendación

Luciano Silvi

Directora: Laura Alonso i Alemany

Facultad de Matemática, Astronomía y Física

Abstract

Resumen El principal desafío de los sistemas de recomendación consiste en estudiar el comportamiento del usuario para predecir qué nuevos ítems podrían resultarle relevantes, principalmente cuando el usuario es nuevo en el sistema y no se tiene suficiente información para producir recomendaciones. Este problema es conocido como «arranque en frío».

Para lidiar con esta situación se requiere de un tiempo de aprendizaje en el que el usuario provee una retroalimentación al sistema indicándole así sus preferencias. Pero esto trae aparejado un inconveniente: cuanto mayor demora suponga este proceso, mayor será la pérdida de tiempo tanto para el usuario como para el proveedor del servicio, además de ser una tarea generalmente tediosa.

Es por eso que en este trabajo estudiaremos distintas aproximaciones con el objetivo de intentar reducir el tiempo de aprendizaje del sistema, pero maximizando la utilidad de la información aportada por el usuario. Para ello nos enfocaremos en la aplicación de Aprendizaje Activo, un método de aprendizaje automático que nos ayudará a crear con mayor rapidez un perfil para el usuario en cuestión.

Palabras clave Aprendizaje activo, sistemas de recomendación, clasificación, aprendizaje automático

Abstract *Every time a recommender system has a new user, it does not have enough information to generate recommendations with high precision, this is known as cold start. To deal with this problem, a feedback from the user is needed to know his preferences. This process usually takes a lot of time, which is not desired by neither of them, the user and the service provider.*

Adapting this problem to a classification problem allow us to apply Active Learning techniques that, as we well see, offer some methods to, given the less possible information about a new user, make right predictions with higher precision than the standard solutions applied in this situation.

Keywords *Active learning, recommender systems, classification, machine learning*

Agradecimientos

Agradezco a mi familia, y en especial a mis padres, por el apoyo que me brindaron en estos años.

A los amigos que conocí a lo largo de este trayecto.

A los profesores y la comunidad de FaMAF, cuyo alto nivel educativo y cálido ambiente posibilitó un tránsito cómodo y agradable durante la carrera.

Agradezco especialmente a Laura Alonso Alemany por dirigirme en este trabajo, por su calidad docente y humana, y por su predisposición a ayudar en todo lo posible.

Índice

Capítulo 1

Introducción y Motivación

Los sistemas de recomendación han demostrado ser una respuesta a la abundancia de información existente en el mundo actual, cambiando la forma en que las personas encuentran productos, información, e incluso otras personas. El principal desafío de estos sistemas consiste en estudiar el comportamiento del usuario para predecir qué nuevos ítems podrían resultarle relevantes [10].

Un problema aún presente es el denominado «arranque en frío», el mismo consiste en solucionar el siguiente planteo: dado un nuevo usuario en nuestro sistema, ¿Cómo podemos sugerirle los ítems de mayor interés para él, si aún no contamos con información suficiente sobre sus preferencias? El mismo caso puede trasladarse a un nuevo ítem en el sistema: ¿A qué clase de usuarios debemos recomendarle ese ítem?

Para lidiar con esta situación se requiere de un tiempo de aprendizaje en el que el usuario, a través de diversos mecanismos, provee una retroalimentación al sistema indicándole así sus preferencias. Claramente, mientras más información brinde el usuario, las sugerencias serán realizadas con mejor nivel de precisión por parte del sistema. Pero esto trae aparejado un inconveniente: cuanto mayor demora suponga este proceso, mayor será la pérdida de tiempo tanto para el usuario como para el proveedor del servicio, además de ser una tarea generalmente tediosa.

Es por eso que en este trabajo estudiaremos distintas aproximaciones con el objetivo de intentar reducir el tiempo de aprendizaje del sistema, pero maximizando la utilidad de la información aportada por el usuario. Para ello comenzaremos aplicando técnicas clásicas en esta área como punto de partida, para luego enfocarnos en la aplicación de Aprendizaje Activo, un método de aprendizaje automático que nos ayudará a crear con mayor rapidez un perfil para el usuario en cuestión.

1.1. Estructura de la Tesis

Comenzaremos analizando el trabajo relevante en el área de sistemas de recomendación que incluye referencias a las técnicas con las cuales experimentaremos posteriormente (capítulo 2). Luego comentaremos los aspectos de la arquitectura de estos sistemas (capítulo 3), explicando los diferentes procesos aplicables como clustering, suavizado y Aprendizaje Activo, en el cual haremos énfasis. En la sección de Experimentos (capítulo 4) detallaremos los distintos estudios realizados con sus resultados, interpretaciones y posibles mejoras.

El sistema sobre el cual trabajaremos será un sistema de recomendación de películas a usuarios, utilizando para su construcción un conjunto de datos existente libremente (Movielens [23]), que aporta una cantidad significativa de elementos, usuarios y calificaciones para simular un sistema en producción.

Capítulo 2

Trabajo relevante

2.1. Sistemas de Recomendación

Los sistemas de recomendación consisten en técnicas y soluciones de software cuyo objetivo es proveer sugerencias de ítems a ser «consumidos» por un usuario. Las sugerencias se obtienen luego de diversos procesos de toma de decisiones, como qué ítems comprar, qué música escuchar, qué videos ver o qué noticias leer [1] .

A lo largo de este trabajo usaremos la denominación «ítem» para referirnos en general a cada elemento a recomendar en el sistema, aunque usualmente cada sistema suele concentrarse en una determinada clase de elementos (música, libros, películas, etc.).

Los sistemas de recomendación se han vuelto muy populares en los últimos años y cuentan con gran variedad de aplicaciones, como mencionamos previamente: desde películas, libros, o productos en comercios electrónicos hasta restaurantes, noticias, personas en redes sociales o resultados de un buscador.

Si bien distintos mecanismos son utilizados para generar el listado de sugerencias para un determinado usuario, a continuación desarrollaremos dos de los principales.

2.1.1. Filtrado Colaborativo (*Collaborative Filtering*)

Utilizando Filtrado Colaborativo basado en usuarios, las sugerencias se realizan analizando el comportamiento pasado de un usuario (productos elegidos, calificaciones, etc.) junto al comportamiento y decisiones de usuarios similares. La retroalimentación al sistema se puede llevar a cabo tanto implícita como explícitamente. En el primer caso se analizarán ítems vistos, búsquedas realizadas o compras, mientras que en el segundo se solicitarán calificaciones, rankings o un listado de preferencias.

En sistemas de grandes dimensiones, la búsqueda de usuarios similares puede requerir mucho tiempo o capacidad de cómputo, por lo que han surgido alternativas como el Filtrado Colaborativo basado en ítems, donde se analizan similitudes entre elementos del sistema utilizando las calificaciones otorgadas por los usuarios. Este cálculo puede computarse cada cierto tiempo para actualizar las similitudes, además de poder realizarse de manera *offline*.

Existe una serie de problemas que suelen afrontar los sistemas que utilizan Filtrado Colaborativo, destacándose los siguientes:

- *Arranque en frío (cold start)*: La falta de información ante un usuario o ítem nuevo dificulta la tarea de encontrar resultados relevantes, pues no hay historial anterior con el cual comparar.
- *Escalabilidad*: Parte de las metodologías y algoritmos utilizados con frecuencia en esta clase de sistemas se vuelven lentos y difíciles de aplicar cuando el sistema alcanza grandes dimensiones. En estos casos es necesario adaptar las técnicas para lograr la rapidez de respuesta necesaria. Las grandes compañías usualmente optan por distribuir el procesamiento aumentando para ello los recursos de hardware.
- *Dispersidad*: Los ítems en el sistema generalmente superan a la cantidad de usuarios, provocando que incluso los ítems más populares sólo hayan sido calificados por un pequeño grupo de usuarios, lo que puede dificultar la tarea de encontrar semejanzas entre usuarios. Este problema puede ser evitado utilizando Filtrado Colaborativo basado en ítems, donde, en lugar de buscar similitudes entre usuarios, se lo hace entre ítems, puesto que hay más cantidad de calificaciones por ítem que por usuario.

  La serie de algoritmos SlopeOne [2] hace uso de esta ventaja para maximizar la precisión de las sugerencias realizadas.

Algunas técnicas utilizadas en la implementación de Filtrado Colaborativo consisten en aplicar k-nearest neighbor (k-NN), correlación de Pearson y factorización de matrices (para lograr reducciones de dimensionalidad). Vale recordar que, como mencionamos con anterioridad, estas implementaciones deben ser modificadas o adaptadas al trabajar a gran escala, puesto que, de lo contrario, la ralentización de las ejecuciones vuelve al sistema inviable.

2.1.2. Basado en Contenido (*Content-Based*)

En este método, la obtención de nuevas sugerencias para el usuario se basa en el análisis de propiedades y características de los ítems, obteniendo así aquellos que compartan

mayor cantidad de propiedades con los contenidos en el historial del usuario y en su perfil. El mismo es modelado indicando características y atributos de los ítems en el sistema junto a un valor (o peso) que indica el nivel de importancia para el usuario de determinada característica.

En estas soluciones se utilizan desde aproximaciones simples como tomar valores promedios de ítems calificados en el pasado, hasta implementaciones más complejas utilizando clasificadores Bayesianos, clustering, árboles de decisión y redes neuronales para intentar estimar la probabilidad de que a cierto usuario le resulte relevante un determinado ítem.

La principal desventaja notable en esta clase de implementaciones es que, debido a su naturaleza, no se incorpora la posibilidad de sugerir elementos que, a pesar de no concordar con las preferencias específicas del usuario, pueden interesarle.

2.1.3. Sistemas Híbridos

En el mercado actual, con el objetivo de optimizar resultados, usualmente se combinan diferentes métodos en los denominados sistemas híbridos, reduciendo así algunas de sus desventajas. Para ello existen varias soluciones disponibles: aplicar Filtrado Colaborativo y sobre ese resultado aplicar Content-Based (o viceversa), o combinar diversas aproximaciones bajo un mismo modelo.

Netflix [3], popular sitio web que ofrece series y películas, es un ejemplo de utilización de un sistema híbrido, ofreciendo recomendaciones basadas en las películas vistas y búsquedas realizadas por usuarios similares (Filtrado Colaborativo), sugiriendo además contenidos que comparten características con películas calificadas anteriormente por el usuario (Content-Based).

En estos sistemas híbridos comerciales se añaden además heurísticas propias de cada proveedor, tanto por razones logísticas como de marketing (por ejemplo, promocionar un determinado producto). Esto posibilita, por ejemplo, integrar un nuevo producto al sistema al ser sugerido y calificado por la mayor cantidad de usuarios posible.

2.1.4. Conferencias y desafíos

Con frecuencia se desarrollan competencias por parte de empresas con el objetivo de mejorar los resultados de sus algoritmos de recomendación. El más conocido en los últimos años ha sido el *Netflix Prize* [4], el cual consistió en un premio de un millón de dólares para el equipo que lograra reducir el error cuadrático del algoritmo de Filtrado Colaborativo de Netflix.

Además, cada año se realizan las Conferencias sobre Sistemas de Recomendación organizada por la ACM [5] (RecSys) [6], donde se presentan y comparten las últimas novedades e investigaciones en el área.

2.2. Aprendizaje activo

El Aprendizaje Activo es un tipo de aprendizaje automático semi supervisado cuya característica clave se basa en la elección de elementos representativos de un conjunto sin clasificar, solicitarle al usuario su clasificación y luego realizar el entrenamiento para mejorar los resultados. Este proceso es iterativo, esto es, se llevan a cabo varios ciclos de aprendizaje donde los elementos representativos deben ser nuevamente seleccionados a medida que el clasificador realiza su entrenamiento, permitiendo así cubrir aquellos puntos que colaboren con la mejora del aprendizaje en cada momento mientras dure el proceso.

A diferencia del Aprendizaje Pasivo, donde los ejemplos a clasificar se seleccionan aleatoriamente, en el aprendizaje activo se selecciona la menor cantidad de ejemplos o características que pueda brindar mayor información implícita al algoritmo de entrenamiento y aprendizaje. En otras palabras, el objetivo es alcanzar una buena precisión con tan pocas instancias etiquetadas como sea posible, minimizando así el costo de obtener nuevas etiquetas (y reduciendo, por lo tanto, la interacción con el usuario).

El Aprendizaje Activo es una motivación en los problemas de aprendizaje automático que manejan grandes cantidades de datos, donde su clasificación y etiquetado manual puede ser muy costoso o llevar mucho tiempo [7].

Un caso particular de este tipo de aprendizaje es el Aprendizaje Activo mediante el Etiquetado de Características («Active Learning by Labeling Features»), donde en lugar de solicitar al usuario que etiquete instancias, se le solicita que etiquete características de las mismas. Este enfoque ha mostrado buenos resultados junto al enfoque clásico para algunas aplicaciones [8]. Estas ideas, junto a otras publicaciones de Burr Settles, como Dualist [9], también serán exploradas en los experimentos del presente trabajo.

2.2.1. Importancia del aprendizaje activo en sistemas de recomendación

Extender un Sistema de Recomendación con Aprendizaje Activo ayuda a mejorar la performance al momento de generar nuevas recomendaciones mientras se reduce la cantidad de información provista por el usuario de manera interactiva.

Muchos sistemas solicitan en cierto momento, por ejemplo al registrarse o ingresar por primera vez, que el usuario califique o indique sus gustos ante productos pre-seleccionados por expertos, que se suponen representativos de un grupo mayor de productos y podría ayudar a dilucidar las preferencias del nuevo usuario. Ahora bien: ¿Tiene este proceso alguna desventaja? Tengamos en cuenta lo siguiente: los productos o ítems pre-seleccionados representan agrados o desagrados para usuarios promedio, mientras que el objetivo de un sistema de recomendación es producir sugerencias con la mayor personalización posible, por lo que suponer que todo usuario podrá quedar definido a partir de características estándares no es una buena aproximación.

Además, dado que las situaciones para aprender los gustos del usuario no son abundantes, es conveniente explotar al máximo la ganancia de información a través de la elección de ejemplos o características representativas, tal como se analiza en [10].

2.2.2. Aprendizaje Activo sobre Instancias

El enfoque más tradicional de Aprendizaje Activo consiste en interactuar con el usuario solicitándole que etiquete ejemplos de instancias para los cuales su clasificación repercuta en una mejora de performance en el sistema. En este trabajo abordaremos este enfoque para luego comparar su rendimiento con el Aprendizaje Activo sobre características.

2.2.3. Aprendizaje Activo sobre Características

Trabajos mencionados previamente, como Dualist [9], sugieren que muchas veces se obtiene una mejor retroalimentación del sistema al solicitarle al usuario no sólo que etiquete y clasifique instancias (como documentos de texto, libros o películas) sino también sus características (como palabras, frases o géneros). Esta aproximación permite ampliar el espacio de búsqueda y podría ayudar a incrementar así el grado de personalización de los resultados. Pondremos a prueba esta hipótesis en la sección de Experimentos.

Capítulo 3

Arquitectura

En este capítulo presentamos las principales decisiones arquitecturales de nuestro sistema. En primer lugar introduciremos la arquitectura clásica de los sistemas de recomendación, para continuar luego con algunos aspectos diferenciadores que exploramos e incorporamos en nuestro sistema: suavizado, clasificación y aprendizaje activo. Terminamos con una descripción general de la arquitectura y un diagrama de flujo de la misma.

3.1. Arquitectura clásica de Sistemas de Recomendación

Una implementación común para un sistema de recomendación consiste en, dada una matriz de preferencias, donde las filas representan a usuarios y las columnas a ítems, aplicar fórmulas de correlación como el Coeficiente de Pearson [11], o algoritmos como K Nearest Neighbor [12] o SlopeOne [2] para obtener sugerencias de ítems para un usuario en particular en base a los elementos consumidos previamente o encontrar los usuarios más similares y a partir de ellos realizar recomendaciones.

El sistema que utilizaremos como punto de partida en este trabajo utiliza la Correlación de Pearson para encontrar usuarios similares. El mismo está determinado por la siguiente fórmula:

$$r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{n s_x s_y}$$

donde x e y son dos vectores de tamaño n, s es la desviación estándar, y $\bar{x}$, $\bar{y}$ son los promedios de los valores de cada vector.

La misma es utilizada para encontrar usuarios similares en el sistema, y a partir de allí generar recomendaciones haciendo uso del procedimiento que detallaremos en párrafos posteriores.

Este coeficiente computa la correlación estadística entre dos vectores (que en nuestro caso representan calificaciones de usuarios), obteniendo así su semejanza, esto es, si ambos usuarios tienden a calificar de forma parecida los mismos ítems, su correlación (y por lo tanto su cercanía en cuanto a preferencias) aumentará [13].

Ilustraremos el funcionamiento de este sistema con un ejemplo. Supongamos que deseamos sugerirle ítems al usuario "43", para ello realizaremos lo siguiente:

1. Buscamos los usuarios similares a 43, llamémosle SIM al conjunto de usuarios similares.
2. Seleccionamos aquellos ítems puntuados por usuarios en SIM que no han sido puntuados por 43.
3. Estimamos el puntaje que otorgará el usuario 43 para cada ítem sumando los puntajes otorgados por los usuarios en SIM pesados por su coeficiente de similitud.
4. Los puntajes estimados para cada ítem son normalizados con la suma de todas las similitudes, obteniendo así los puntajes finales estimados para 43 de los elementos a sugerir.

El ejemplo ilustrado se encuentra representado en el cuadro 3.1, donde la última fila muestra los puntajes estimados por el sistema para el usuario 43 para los ítems mencionados en la tabla.

Usuarios	Similitud	Star Wars	Star Wars x Similitud	LOTR	LOTR x Similitud	The Hobbit	The Hobbit X Similitud
36	1.0	5	5.0	5	5.0	3	3.0
105	0.83	4	3.32			4	3.32
824	0.8	4	3.2	5	4.0	4	3.2
179	0.79	4	3.16	4	3.16		
812	0.79	3	2.37	5	3.95	5	3.95
Total			**17.05**		**16.11**		**13.47**
Total Similitud			4.21		3.38		3.42
Total/Total Similitud			**4.04**		**4.76**		**3.93**

Cuadro 3.1: Ejemplo de predicción de calificaciones.

El sistema descripto se basa en un ejemplo presente en el libro Programming Collective Intelligence [14]. Sobre este sistema realizaremos diversos experimentos, ejecutando pruebas con diferentes algoritmos y técnicas para contar con una variedad de resultados a comparar y analizar.

La aproximación clásica a encontrar los usuarios semejantes a un usuario objetivo es el algoritmo *K nearest neighbors* (K-nn) [12] mediante el cual, dado un conjunto de elementos representados mediante vectores en un plano multidimensional, se puede analizar un nuevo elemento de la siguiente manera:

1. Identificar los K elementos más cercanos al elemento en cuestión. La determinación del valor de K merece un estudio aparte, pues si es muy alto se ayuda a reducir el ruido pero pueden surgir resultados demasiado generales, mientras que si es muy bajo puede llegarse a conclusiones apresuradas e imprecisas.
2. Asignar al elemento bajo análisis las características con mayor número de ocurrencias entre los elementos cercanos. Dado que trabajamos sobre sistemas de recomendación, el resultado consistiría en recomendarle al usuario los ítems que más ocurren en los usuarios cercanos (o los que tienen mejores calificaciones).

Sin embargo, nuestro abordaje al problema de recomendación será realizado de una manera diferente, debido a motivos que detallaremos más adelante.

3.2. Suavizado

Una mejora posible para el funcionamiento de un método de recomendación es perfeccionar la forma en que el método generaliza. Muchas veces sucede que los modelos que obtenemos están tan ligados a los datos en los que se basan que no podemos generalizar a nuevos casos. Para mejorar esto, es necesario abstraer el modelo a partir de los datos, en lugar de dejarlo ligado a ellos. Mucha de la información sobre la que aplicamos métodos de recomendación tiene una distribución de Long Tail, que es frecuente en los datos de origen humano-social, y que describiremos a continuación.

En estadística, Long Tail [15] (o Cola Larga), es una característica de ciertas distribuciones estadísticas en las cuales gran parte de la población tiene un bajo número de ocurrencias lejos del llamado «centro» de la distribución, es decir, la zona donde se concentran los elementos de la población con mayor número de ocurrencias. Este es el caso de muchos catálogos de productos: unos pocos productos acumulan la mayor parte de las ventas, mientras que la mayor parte de productos se venden muy pocas veces.

El fenómeno puede representarse más claramente mediante el siguiente ejemplo: supongamos la existencia de una tienda de libros, en donde sus limitaciones físicas (espacio en estanterías) hacen imposible la oferta de todo el material existente, obligando a ofrecer sólo un segmento del mismo (usualmente los más populares o que generan mayor ganancia). Así, el material no exhibido tendrá una menor cifra de popularidad y adquisición. Esta tendencia en empresas físicas se ha denominado como regla de 80/20, esto es, la empresa decide concentrarse en el 20 % de sus productos que le otorgan el 80 % de ganancias.

En tiendas en línea como Amazon o Netflix, al no existir limitaciones físicas (por no tener estanterías), el fenómeno de Long Tail no sucede de la misma manera puesto que no hay dificultades en mostrar todos los productos disponibles, sino que se evidencia a través de, por ejemplo, un bajo número de compras o puntuaciones para ciertos ítems en el sistema. En un sistema de recomendación, este bajo número de compras, unido a una falta de generalización, provoca que el sistema no pueda realizar recomendaciones respecto a los ítems raros, ya que no dispone de suficiente información sobre ellos ni puede generalizar adecuadamente a partir de los ítems de los que sí dispone de información.

En algunos casos, aplicar ciertas técnicas de factorización de matrices ayuda a disminuir el problema de Long Tail, suavizando la cola de la distribución y logrando así una mejor representación de, en nuestro caso, ítems a recomendar en el sistema. La Descomposición en Valores Singulares (SVD por sus siglas en inglés) ofrece interesantes propiedades para factorizar matrices en sistemas de recomendación, obteniendo así metodologías para inferir las preferencias de un usuario con simples multiplicaciones de vectores [16]. Existen aplicaciones como Latent Semantic Indexing que utilizan SVD de una manera eficiente para encontrar similitudes entre filas de una matriz (recordemos que nuestra matriz base representa a usuarios en las filas e ítems en las columnas, donde cada elemento [i,j] indica la calificación del usuario i para el item j), por lo que en nuestros experimentos pondremos a prueba este método, analizando así si el suavizado mediante factorización logra una performance notoria en la búsqueda de similitudes junto al Aprendizaje Activo.

3.3. Clasificación

En este trabajo hemos tomado la decisión de abordar el problema de recomendación llevándolo a un problema de clasificación, sustituyendo la búsqueda de usuarios más parecidos a uno dado por un algoritmo donde se clasifica al usuario según un conjunto de clases predeterminadas, como alternativa a la aproximación intuitiva en recomendación, *K-nn*.

El algoritmo *K-nn*, si bien es sencillo e intuitivo, presenta algunos inconvenientes para nuestro trabajo:

- La detección de los elementos más cercanos puede llevar mucho tiempo si el conjunto de datos es grande, pues requiere explorar y analizar gran parte del espacio de búsqueda.
- La elección del valor de K no es trivial, e incluso puede variar junto a la evolución del sistema, lo que dificulta asegurar un nivel de precisión ante un sistema cambiante.

En el ámbito de aprendizaje automático, se denomina clasificación al problema de identificar a qué categoría pertenece una observación, teniendo como base un conjunto de observaciones para las cuales ya se conoce su categoría [17]. Clasificación es una instancia de aprendizaje supervisado, es decir, es necesario un conjunto de entrenamiento para el cual ya se conoce su resultado antes de poder etiquetar una nueva observación.

Un algoritmo que resuelve el problema de clasificación es llamado «clasificador». Los clasificadores que utilizaremos en este trabajo serán detallados en la sección de experimentos.

Nuestro objetivo es determinar un conjunto de clases de forma que cada usuario del conjunto de entrenamiento pertenezca a una clase. Con ese conjunto entrenaremos un clasificador y así, dado un nuevo individuo, determinaremos a qué clase pertenece, para luego ofrecer las recomendaciones determinadas por su clase.

Esta aproximación supone ventajas sobre los inconvenientes que planteamos en K-nn, por ejemplo, se reduce el espacio de búsqueda, gran parte del proceso puede ser realizado offline, etc. Sin embargo, también surge otro tipo de desventajas: el hecho de generar clases fijas provoca una pérdida de información ante un determinado usuario (puesto que se trata a todos los usuarios dentro de la misma clase como si fueran iguales), perdiendo así ciertos matices que contribuyen a la personalización de los resultados. Sin embargo, si se cuenta con una diversidad y cantidad de usuarios adecuada, pueden determinarse la cantidad de clases necesarias para evitar generalizaciones indeseadas.

En nuestro caso no disponemos de clases predeterminadas o establecidas por un experto de dominio en el conjunto de entrenamiento, por lo tanto, debemos generarlas utilizando la técnica correspondiente para cuando no se dispone de un conjunto de datos etiquetado. La misma se denomina *Clustering* y será analizada en la próxima sección.

3.4. Clustering

El Clustering (o Agrupamiento) es una técnica de aprendizaje no supervisado a través de la cual podemos generar grupos de elementos «cercanos» o similares bajo alguna característica. En nuestros experimentos realizaremos clustering de perfiles de usuarios, obteniendo así grupos de usuarios similares con el objetivo de, ante un nuevo usuario, encontrar el agrupamiento más apropiado con el cual asociarlo.

Distintos algoritmos de clustering están disponibles, siendo los más utilizados K-Means y Expectation-Maximization. En este trabajo utilizaremos el primero ya que sus ejecuciones finalizan en un tiempo prudente, habilitándonos a realizar diversas pruebas en relativamente poco tiempo.

Cabe mencionar que distintos parámetros en la ejecución del algoritmo de clustering pueden ser configurados, tales como número de grupos (o clústers), máximo número de iteraciones del algoritmo, etc. En secciones posteriores detallaremos los parámetros utilizados para la generación de los clústers y su evaluación.

3.5. Aprendizaje Activo

Habiendo reconsiderado el problema de recomendación como uno de clasificación, y luego de haber obtenido clases para los usuarios mediante clustering, podemos aplicar Aprendizaje Activo para lograr nuestros objetivos. Los mismos se basan en optimizar el uso de información provista por el usuario hasta alcanzar un rendimiento aceptable del sistema.

Existen diferentes escenarios posibles de Aprendizaje Activo, donde cada uno de ellos utiliza diversas estrategias para seleccionar qué preguntarle al usuario. A continuación presentaremos uno de los más utilizados, llamado Pool-Based, que es el implementado en las herramientas elegidas para el presente trabajo.

El escenario Pool-Based tiene la siguiente estructura (ilustrada en la figura 3.1) [7]:

- Un conjunto de instancias (o features) sin etiquetar.
- Un conjunto de instancias o features etiquetados.
- Un modelo de Aprendizaje Automático.
- Un oráculo, que etiquetará los ejemplos solicitados por el sistema.

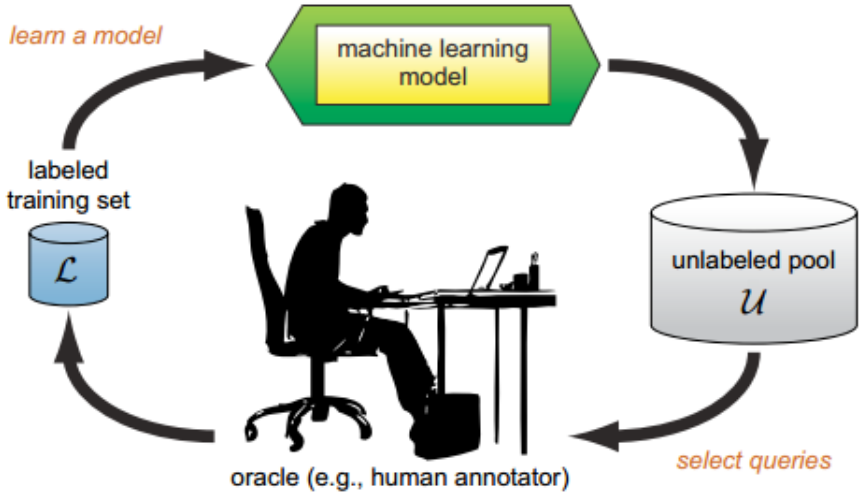

FIGURA 3.1: Ciclo de Aprendizaje Activo

El modelo de aprendizaje comienza con un conjunto pequeño de instancias etiquetadas, luego, utilizando una estrategia de selección, elige cuidadosamente instancias del conjunto no etiquetado, se le solicita al oráculo su etiqueta, se reaprende utilizando estos resultados y se los incorpora al conjunto de instancias etiquetadas. A partir de ello comienza nuevamente el ciclo, eligiendo nuevas instancias para etiquetar.

3.5.1. Ganancia de Información

Para el ordenamiento de instancias del conjunto no etiquetado se utilizará el concepto de Ganancia de Información, avalada como medida de selección de características en estudios realizados por Settles [9] y Forman [18], entre otros. En este último se define la Ganancia de Información como la medida de reducción de la entropía cuando una característica está presente o ausente.

La idea detrás de la aplicación de esta técnica en nuestro trabajo es seleccionar aquellas características o ítems que mejoren la precisión de la clasificación de un usuario. Esto es, escoger características o ítems a calificar para los cuales, si se conociera su preferencia por el usuario, se contaría con un aporte de información significativo que contribuiría a recomendaciones más acertadas.

3.6. Librerías y Software utilizado

Para construir nuestro sistema de recomendación y las técnicas con las cuales realizaremos las pruebas y experimentos detallados más adelante, precisamos de un conjunto

de herramientas de software, algunas disponibles con anterioridad y otras programadas para el presente trabajo.

La arquitectura general consta básicamente de los siguientes módulos:

- El sistema de recomendación detallado en la primera sección de este capítulo, basado en Programming Collective Intelligence [14], con modificaciones necesarias para operar con distintas combinaciones de algoritmos, como veremos posteriormente en los experimentos.
- ActiveLearning: Este componente implementa el ciclo de Aprendizaje Activo ilustrado en la figura 3.1 utilizando Ganancia de Información. Contiene las interfaces necesarias para proveerle los conjuntos de datos y un modelo de aprendizaje. En este módulo utilizamos un cálculo de Ganancia de Información que ya ha sido implementado, probado y utilizado [19].
- Clustering: Este módulo implementa funciones para obtener agrupamientos, utilizando para ello los algoritmos e interfaces presentes en la librería Scikit Learn [21].
- Dataset: Módulo que permite, a partir de cierto conjunto de datos, crear las matrices necesarias para nuestros experimentos y ejecución del sistema. Entre estas matrices se destacan las de *usuariosXratings* (donde se indican las calificaciones otorgadas por cada usuario a cada ítem en el sistema) y *usuariosXcaracterísticas*, donde por cada ítem en el sistema se indican las características de ese ítem (esto nos será útil para realizar Aprendizaje Activo sobre Características).
- Además, se utilizan librerías adicionales como GenSim [22], que provee una sencilla y eficiente manera de trabajar con Latent Semantic Indexing.

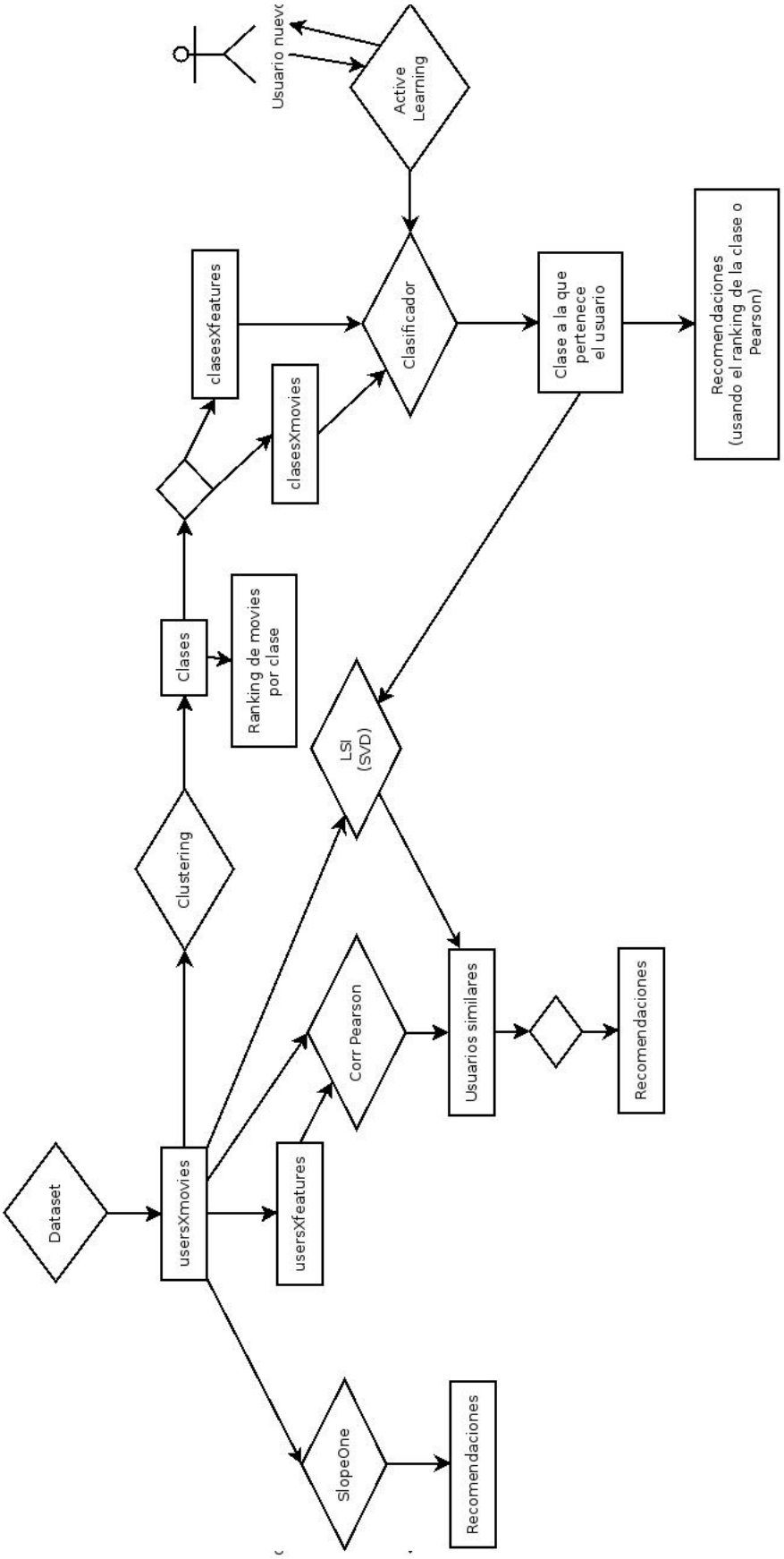

FIGURA 3.2: Arquitectura del sistema y Ciclos de ejecución

Capítulo 4

Entorno de experimentación

4.1. Conjunto de Datos

El trabajo presentado en este documento puede ser aplicable a cualquier tipo de sistemas de recomendación, pero en este caso, y debido a la disponibilidad de conjuntos de datos involucrando películas, nuestro sistema simulará la recomendación de películas a usuarios. Para ello, hacemos uso de los datos ofrecidos por MovieLens, disponibles públicamente en su sitio web [23].

MovieLens es un sistema de recomendación online que sugiere películas a sus usuarios basándose en sus preferencias y haciendo uso de la técnica de Filtro Colaborativo. El sitio web es un proyecto de GroupLens Research, laboratorio del Departamento de Ciencias de la Computación e Ingeniería de la Universidad de Minnesota. El objetivo principal del sitio es recolectar datos para investigación sobre sistemas de recomendación personalizados, es por esto que este conjunto de datos se hace disponible libremente.

Los datos allí disponibles ofrecen distintas cantidades de calificaciones de usuarios a películas, «etiquetas» o características sugeridas por los usuarios para ciertas películas, e información adicional (como edad, género y ocupación) que, si bien inicialmente no consideramos en este trabajo, podrían ser involucrados en el futuro.

El conjunto de datos utilizado específicamente es «MovieLens 100k», que consta de **100000 calificaciones por parte de 1000 usuarios para 1700 películas**. Además, se utilizan las etiquetas (características) provistas por el conjunto de datos «MovieLens 10M», que consta de **100000 etiquetas** asignadas por usuarios del sistema a películas en el mismo.

Para realizar experimentos adicionales y analizar el comportamiento de los mismos en conjuntos de datos mayores se han utilizado también los datos provenientes de

«MovieLens 10M», que consta de 10 millones de calificaciones para 10000 películas por 72000 usuarios.

4.2. Métricas de Evaluación

Son conocidos diversos métodos de evaluación para sistemas de recomendación dependiendo de qué es lo que se desea analizar: predicción de calificaciones, uso de items en el sistema, cobertura, etc. Como se menciona en Evaluating Recommender Systems [24] y en la presentación Recommender Systems (IJCAI 2013) [25] , una de las métricas más utilizadas para analizar la precisión de las predicciones de calificaciones por parte de un usuario es la raíz del error cuadrático medio (RMSE, por su denominación en inglés), cuya fórmula es la siguiente

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2}$$

donde $\hat{Y}$ es el vector con predicciones realizadas por el sistema e Y es el vector con los valores reales para esos ítems.

Esta es la métrica que utilizaremos para analizar y comparar los resultados de nuestros experimentos.

4.3. Clases obtenidas a partir de los datos

Como mencionamos en capítulos anteriores, la aplicación de técnicas de Clustering (o agrupamiento), permite formar grupos de elementos similares en un conjunto de datos. En este trabajo hemos tomado la decisión de realizar clustering sobre perfiles de usuarios, con el objetivo de crear diferentes grupos y así, ante un nuevo usuario en el sistema, poder ubicarlo en el grupo que más se corresponde con su perfil.

Los resultados de este agrupamiento serán utilizados en conjunto con técnicas de aprendizaje activo, como veremos en la sección experimentos. Así, dado un nuevo usuario, se aplicará aprendizaje activo para inferir sus preferencias y recién allí asociarlo con el grupo adecuado.

4.3.1. Clustering de usuarios

Diferentes configuraciones pueden ser especificadas al aplicar esta clase de soluciones. En este caso, como pretendemos mejorar al máximo la identificación de un usuario con su grupo, hemos tomado la decisión de crear grupos relativamente pequeños (alrededor de cien usuarios por grupo) utilizando el algoritmo KMeans [26] sobre los perfiles de mil usuarios de nuestro conjunto de datos. El pequeño tamaño de cada grupo nos permite asegurar la cercanía entre sus miembros.

Inicialmente se consideraron dos posibilidades: realizar el clustering con respecto a las *calificaciones otorgadas por los usuarios* o con respecto a las *características de las películas calificadas por los usuarios*. Con la presunción de que la segunda alternativa brindaría resultados más acertados, ejecutamos el algoritmo y obtuvimos diez grupos de usuarios con diversas inconsistencias tales como: grupos muy pequeños o grupos muy grandes cuyos miembros no eran cercanos en cuanto a preferencias. Es por ello que nos decantamos por la primera opción, así, utilizando calificaciones para ejecutar el clustering, obtuvimos diez grupos cuya evaluación realizamos a continuación.

4.3.1.1. Evaluación

Luego de ejecutar el algoritmo K-Means sobre el conjunto de datos de Movielens, que cuenta con mil usuarios, especificando que deseamos obtener diez grupos (para reducir la cantidad de usuarios en cada clase) y con generación aleatoria de los centros de cada grupo (puesto que nuestro conjunto de datos no está etiquetado y no podemos establecer previamente elementos indicativos de cada clase), se obtuvieron diez clases de entre 40 y 182 elementos cada uno.

Sobre este resultado realizamos un proceso de evaluación que consistió en dos chequeos:

- Para dos usuarios cualesquiera en el sistema se calcula su correlación (utilizando el coeficiente de Pearson) y a partir de ella, si su valor es próximo a 1, se comprueba que ambos usuarios pertenezcan al mismo grupo. Caso contrario, si el coeficiente es menor a 0, se comprueba que ambos usuarios estén en distintos grupos, pues no tienen comportamiento similar al momento de calificar. Este chequeo fue exitoso con los grupos obtenidos.
- Se realizó una evaluación anecdótica de las películas en cada grupo, para observar si resultaban naturalmente relacionadas bajo alguna característica en común. Se encontró así que en la mayoría de los casos las películas de un mismo director, de un mismo género o subgénero o de un mismo actor se encontraban dentro de

los mismos clusters con calificaciones similares por parte de sus usuarios. Ejemplo de esta inspección resulta una clase donde se vio que gran parte de sus miembros habían calificado películas como «Star Wars», «Independence Day», «Alien» o «Star Trek» de forma similar.

Luego de estas evaluaciones se pudo continuar con el proceso teniendo como base un resultado de clustering satisfactorio y acorde a lo que necesitábamos para nuestro ciclo de recomendación.

4.3.1.2. Ranking de ítems por clase

Una vez finalizada la generación de cada cluster (o clase), elaboramos un ranking por cada una de ellas promediando las calificaciones otorgadas por sus usuarios a los ítems en el sistema, ordenándolos luego de manera decreciente. Así, posteriormente, cuando deseemos realizar recomendaciones a un usuario dado, sólo debemos recorrer este ranking encontrando aquellos ítems que no hayan sido calificados aún por el usuario en cuestión.

Capítulo 5

Experimentos y Análisis de Resultados

5.1. Experimentos agregando preferencias (calificaciones) para un usuario

En los experimentos detallados a continuación se utilizarán las calificaciones de ítems otorgadas por los usuarios (salvo que se indique lo contrario) para obtener recomendaciones a través de diferentes técnicas. Este proceso nos posibilitará poner a prueba los distintos mecanismos propuestos para mejorar el desempeño del sistema durante el Arranque en Frío. En algunos casos, los resultados de un experimento desencadenaron nuevas variaciones y técnicas a probar, algunas de las cuales están presentes en este análisis mientras que otras se proponen como trabajo futuro.

Concretamente, lo que haremos es, para cada experimento, armar una matriz de preferencias para todos los usuarios menos el primero, que utilizaremos para realizar la evaluación. Esto es, estaremos simulando un sistema en producción al cual se incorpora un nuevo usuario. Luego, iremos agregando preferencias para ese usuario en evaluación, generaremos recomendaciones basadas en esas preferencias y sobre ellas calcularemos el error cuadrático medio como mencionamos previamente.

Este modo de evaluación se aplicará a todos los usuarios, es decir, repetiremos el proceso pero evaluando al segundo usuario, luego al tercero, y así sucesivamente hasta el último. Finalmente obtendremos un promedio de error cuadrático para cada número de preferencias, resultado que quedará plasmado en los gráficos.

Cabe mencionar que en los experimentos con Aprendizaje Activo, el proceso será similar, con la diferencia de que, en lugar de calcular el error para todos los usuarios, se hará

con una parte de ellos, puesto que la parte restante será utilizada para entrenar el clasificador. Además, el oráculo mencionado en la sección 3.5 será reemplazado por los valores presentes en el conjunto de datos para el usuario bajo evaluación, puesto que al tener disponibles *a priori* el valor de las calificaciones de cada usuario, no es necesario un oráculo humano u otro algoritmo que cumpla su función.

Así mismo, algunos experimentos fueron dejados de lado debido a que, como se mencionó en la sección 4.3, algunos de los resultados de clustering obtenidos fueron poco satisfactorios como para realizar pruebas partiendo de ellos. Este es el caso de, por ejemplo, la generación de clases a partir de etiquetas asignadas por los usuarios a los ítems, como veremos más adelante.

5.1.1. Baselines

A continuación presentaremos algunas soluciones clásicas para recomendación, que tomaremos como marco de comparación con respecto a las innovaciones propuestas.

5.1.1.1. Experimento 1: Recomendaciones utilizando correlación de Pearson

Como describimos con anterioridad en la sección Arquitectura (sección 3), el coeficiente de correlación de Pearson es utilizado para encontrar usuarios similares en el sistema, y a partir de allí generar recomendaciones. Esta aproximación es una de las más básicas y difundidas [13], y suele brindar resultados prácticos aceptables en gran parte de los casos. Es por esto que la utilizamos como uno de los baselines.

Resultados Como puede observarse en la Figura 5.1, a medida que se añaden nuevas preferencias para el usuario en evaluación, el error (RMSE) aumenta de manera considerable, llegando a valores superiores a 1.6 cuanto mayor es la cercanía a conocer la preferencia para la totalidad de elementos.

Esta desmejora de precisión puede deberse a deficiencias propias del coeficiente de correlación de Pearson, donde en algunos casos se obtiene alta similitud entre dos usuarios aún cuando sólo comparten pocas calificaciones semejantes. Esto suele originarse al no tener en cuenta el número total de calificaciones, causando similitudes erróneas entre usuarios que han calificado solamente 1 ó 2 ítems en forma similar, cuando en realidad no se puede asegurar una semejanza entre ambos.

Imaginemos, a modo de ejemplo, dos usuarios, A y B, donde ambos otorgaron una calificación similar para la película *2001: Odisea del espacio*, pero el resto de sus calificaciones no son compartidas (es decir, las películas calificadas por A no fueron calificadas por

B, y viceversa). Ahora, el coeficiente de Pearson podría identificar una similitud entre ambos usuarios, pero el problema surge cuando las preferencias de B son distintas a las de A (B puede preferir las películas ambientadas en el espacio, mientras que A puede preferir exclusivamente las películas de la década de 1960).

Tal como predice la distribución de Pareto [27], hay pocos usuarios con muchas calificaciones otorgadas, mientras que la gran parte sólo califica una pequeña porción de ítems.

Es por ello que suelen introducirse modificaciones en el cálculo del coeficiente, para contrarrestar las falsas semejanzas. Algunas de ellas consisten en establecer un número mínimo de calificaciones compartidas para asegurar su similitud o escalar el valor de la correlación si los ítems co-calificados caen por debajo de ese mínimo [13].

5.1.1.2. Experimento 2: Recomendaciones utilizando SlopeOne

SlopeOne [2] es un algoritmo de recomendación cuya base se sostiene en, al momento de sugerir potencialmente un cierto ítem, analizar su similitud con otros elementos calificados anteriormente por el usuario, así como también información proveniente de otros usuarios que han calificado el mismo ítem. Su eje se centra en calcular similitudes entre ítems en lugar de perfiles de usuarios. Esto permite evadir los problemas mencionados en el experimento anterior donde surgían similitudes erróneas entre usuarios con pocas calificaciones otorgadas.

A modo de ilustración, en la matriz de usuariosXratings mencionada en capítulos anteriores (3.6 y 3.1), SlopeOne analizará también la similitud entre columnas (basado en ítems), mientras que en el experimento anterior se analizó sólo la similitud entre filas (basado en usuarios).

Hipótesis Debido a su naturaleza y al argumento fundamentado en su presentación, es de esperar que el error cuadrático medio disminuya en lugar de aumentar. Esto se debe a que, entre otras cosas, su diseño no permite que ocurran falsas similitudes, y ante mayor información obtenida sobre los usuarios e ítems en el sistema, mejor serán los resultados.

Resultados Como puede apreciarse en la Figura 5.1, a medida que vamos introduciendo mayor cantidad de ejemplos de preferencias para un usuario, el error disminuye progresivamente, acentuándose su decrecimiento ante mayor cantidad de información añadida. Cabe destacar que, cuanto más cerca está el sistema de conocer las preferencias de un usuario para la totalidad de ítems, el error es menor a 0.5.

Es notable la comparación con los resultados del experimento anterior, puesto que mientras utilizando correlación de Pearson el error aumenta, utilizando SlopeOne sucede todo

lo contrario aproximadamente con la misma rapidez. Esto se debe a los errores propios que puede introducir Pearson si se utiliza en su estado puro sin modificaciones particulares, mientras que SlopeOne fue concebido desde el principio con la premisa de brindar buenos resultados de recomendación [13] [2].

5.1.1.3. Experimento 3: Recomendaciones utilizando Latent Semantic Indexing

Hipótesis Introducir técnicas de factorización de matrices y reducción de dimensionalidad, como Latent Semantic Indexing, que hace uso de Singular Value Decomposition, podría ayudar a mejorar la detección de usuarios similares. Esto se debe a que esta clase de técnicas ayudan a reducir el *overfitting* (o sobreajuste), que surge cuando un modelo se adapta demasiado a los datos de entrenamiento, siendo así muy susceptible a la introducción de nuevos datos, conduciendo a errores. En nuestro caso, estos métodos permiten modelar mejor la cola de la distribución, ofreciendo así una representación más acertada de los datos 3.2.

LSI utiliza además la similitud Coseno, dada por la siguiente fórmula:

$$sim = \frac{A * B}{\| A \| \| B \|}$$

Donde A y B son vectores de atributos y $\|\|$ es la norma euclídea.

La fórmula mencionada es utilizada para encontrar vectores en la matriz que resulten similares al dado. En este caso, el vector dado será el usuario en evaluación, mientras que los vectores similares representarán a los usuarios similares al dado.

Resultados Luego de utilizar el mismo sistema mencionado en el primer experimento, con la única diferencia de que la búsqueda de usuarios similares se realiza a través de LSI en lugar de Pearson, los resultados mostrados en la figura 5.1 ofrecen valores muy cercanos a los obtenidos con el uso de correlación (y por ende, bastante lejano a SlopeOne), por lo que la introducción de LSI no brinda mejoras apreciables.

5.1.1.4. Experimento 4: Recomendaciones utilizando correlación y características

Dados los resultados del Experimento 1 y observando que el error no logra estabilizarse utilizando correlación, surgió la idea de utilizar características de los ítems en lugar de su calificación. Debido a que las características son provistas por los mismos usuarios del

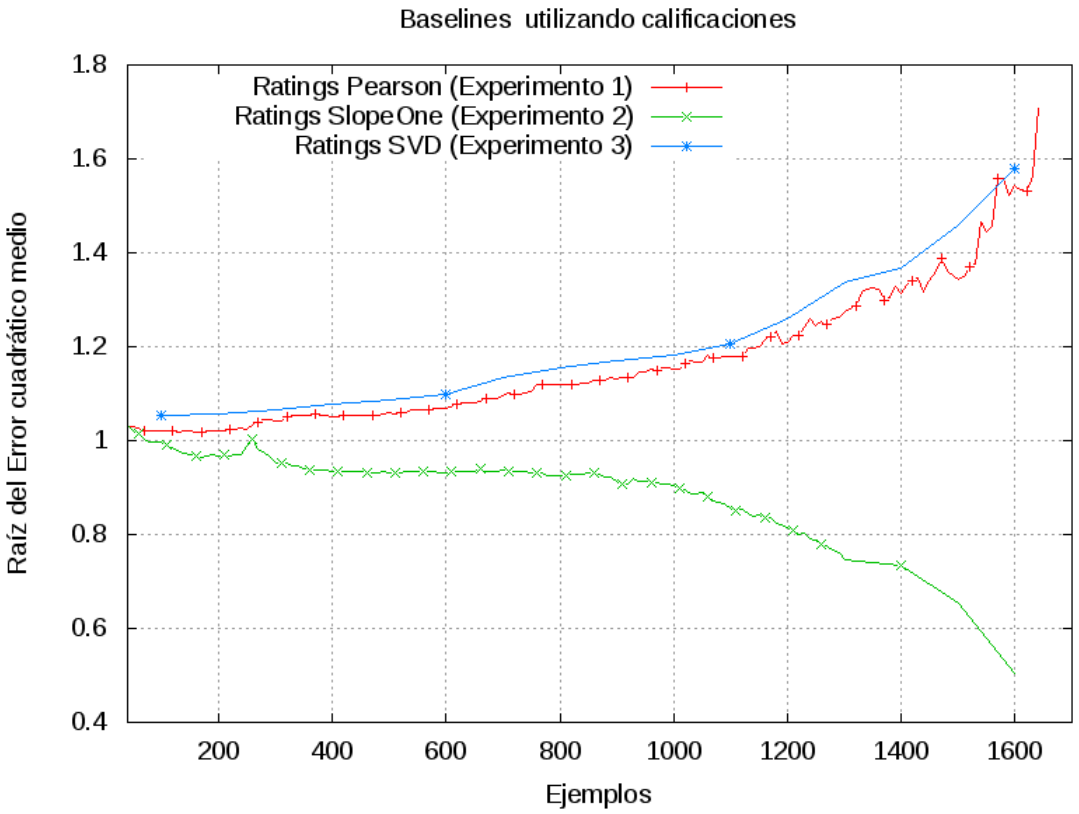

FIGURA 5.1: Baselines

sistema, para una cierta película pueden encontrarse etiquetas como «ciencia ficción», «Keanu Reeves», «deberías verla» y «obra maestra» entre otras, lo cual, como puede apreciarse, muchas veces contiene información subjetiva o irrelevante, introduciendo lo que denominamos «ruido» en los resultados.

Hipótesis A pesar de lo mencionado en el párrafo anterior, consideramos que utilizar las etiquetas podría reducir el error al permitir una mayor flexibilidad en la obtención de similitudes, puesto que el espacio de búsqueda ahora estará particionado en características en lugar de ítems.

Resultados Luego de la ejecución, observamos (en la Figura 5.2) que ante las primeras características el error es alto (alrededor de 1.8), para luego estabilizarse en 1.4. Si bien es importante haber observado un punto de estabilidad, al comparar con los resultados del Experimento 1, donde recién se alcanza un error de 1.4 cuando se conoce cerca de la totalidad de calificaciones para los ítems en el sistema, y ya que nuestro foco se encuentra en el Arranque en Frío, podemos concluir que utilizando características, contrariamente a lo que habíamos pensado, no se brinda una mejora en cuanto a rendimiento. Un experimento que puede realizarse como trabajo futuro es combinar el experimento 3

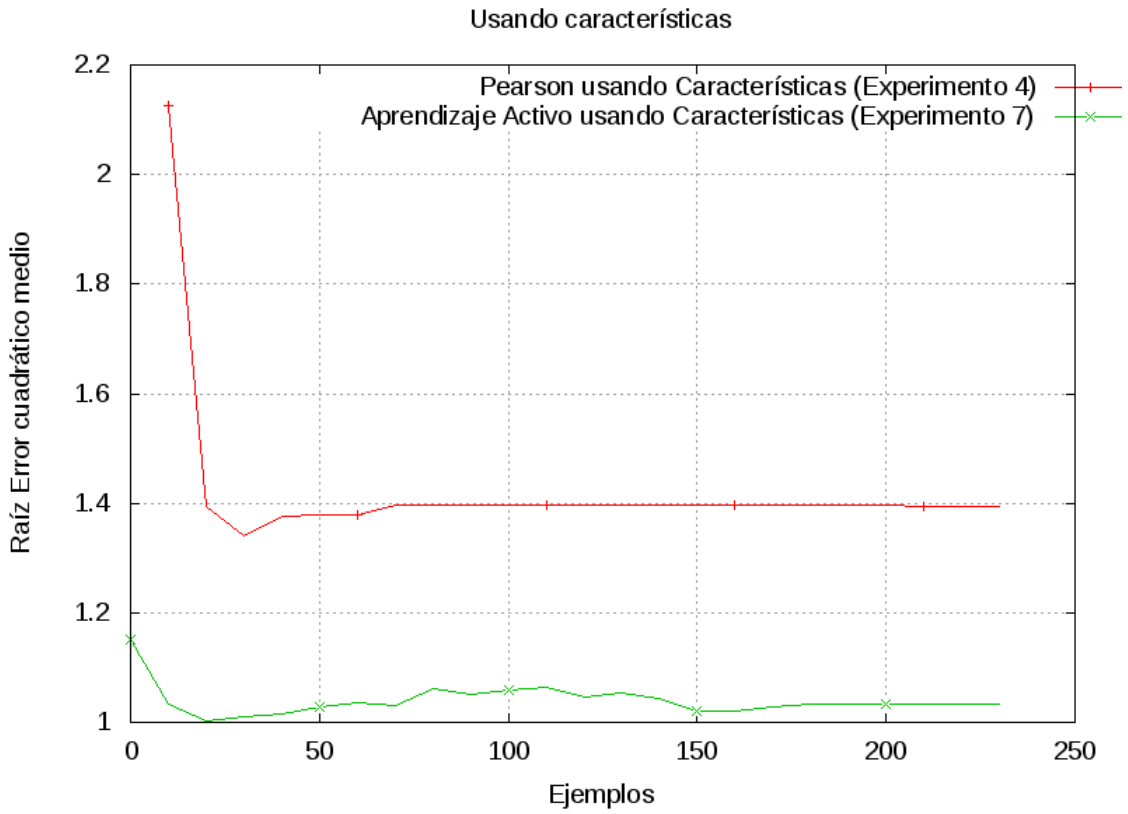

FIGURA 5.2: Usando características otorgadas por usuarios (etiquetas)

(usando LSI) con la matriz de usuariosXcaracterísticas, con la hipótesis de que LSI podría suavizar el ruido introducido con las etiquetas.

Cabe mencionar que, si bien las características presentes en los ítems son más de 5000, en el gráfico (Figura 5.2) se observa que con 200 es suficiente para llegar a las conclusiones expresadas en el párrafo anterior.

5.1.2. Aprendizaje Activo

En esta sección se describirán las pruebas realizadas utilizando Aprendizaje Activo, para ello ejecutaremos el ciclo detallado en el capítulo 3.5, donde el modelo será un clasificador. El mismo será entrenado con las clases resultantes del Clustering de usuarios con respecto a las calificaciones otorgadas como se detalló en 4.3.

5.1.2.1. Experimento 5: Utilizando Aprendizaje Pasivo (usando sólo clasificador y añadiendo calificaciones aleatoriamente)

Este experimento constará sólo de un clasificador con el cual evaluaremos un usuario para obtener recomendaciones a partir de un ranking elaborado para cada clase. Luego, iremos

añadiendo progresivamente calificaciones de ítems con el objetivo de analizar la precisión de la clasificación y mejorarla. Estas calificaciones serán añadidas aleatoriamente (sin utilizar ganancia de información), por lo que estaremos utilizando Aprendizaje Pasivo, lo cual nos servirá como base para luego compararlo con Aprendizaje Activo.

Los clasificadores utilizados para esta prueba serán *Multinomial Naive Bayes* y *Support Vector Machines*. El primero es un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales que asumen independencia entre las variables, y es por eso que recibe el apelativo de «ingenuo» [28]. El segundo es un conjunto de algoritmos de aprendizaje supervisado que construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación apropiada, ya que cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase [29].

Resultados Al observar los resultados (Figura 5.3) con MNB, vemos que el error disminuye más rápidamente que con SlopeOne a partir de 400 calificaciones conocidas para un usuario, mientras que en las primeras iteraciones el comportamiento de ambos es similar. En cuanto a SVM, inicialmente el error es mayor que SlopeOne, para luego alcanzar niveles similares al mismo a partir de las 900 calificaciones conocidas.

5.1.2.2. Experimento 6: Utilizando calificaciones de ítems (y usando el ranking de la clase)

En este experimento utilizaremos el clasificador MNB e introduciremos Aprendizaje Activo a través de **Ganancia de Información** (3.5.1). Introduciremos calificaciones en cada iteración para el usuario en evaluación, al igual que en experimentos anteriores, con la novedad de que los datos serán añadidos dependiendo de cuánta información provean para mejorar la clasificación. Para lograr esto, previamente ordenamos los ítems en el sistema en función de su Ganancia de Información en forma decreciente, así, en cada iteración, añadimos la calificación para el ítem que mayor información aporta y que no haya sido agregado en una iteración anterior para el mismo usuario.

Cabe mencionar que el uso de Aprendizaje Activo a través de Ganancia de Información será conservado en todos los experimentos de aquí en adelante, salvo que se indique lo contrario.

Al igual que en el experimento anterior, las recomendaciones serán realizadas en base a un ranking por clase.

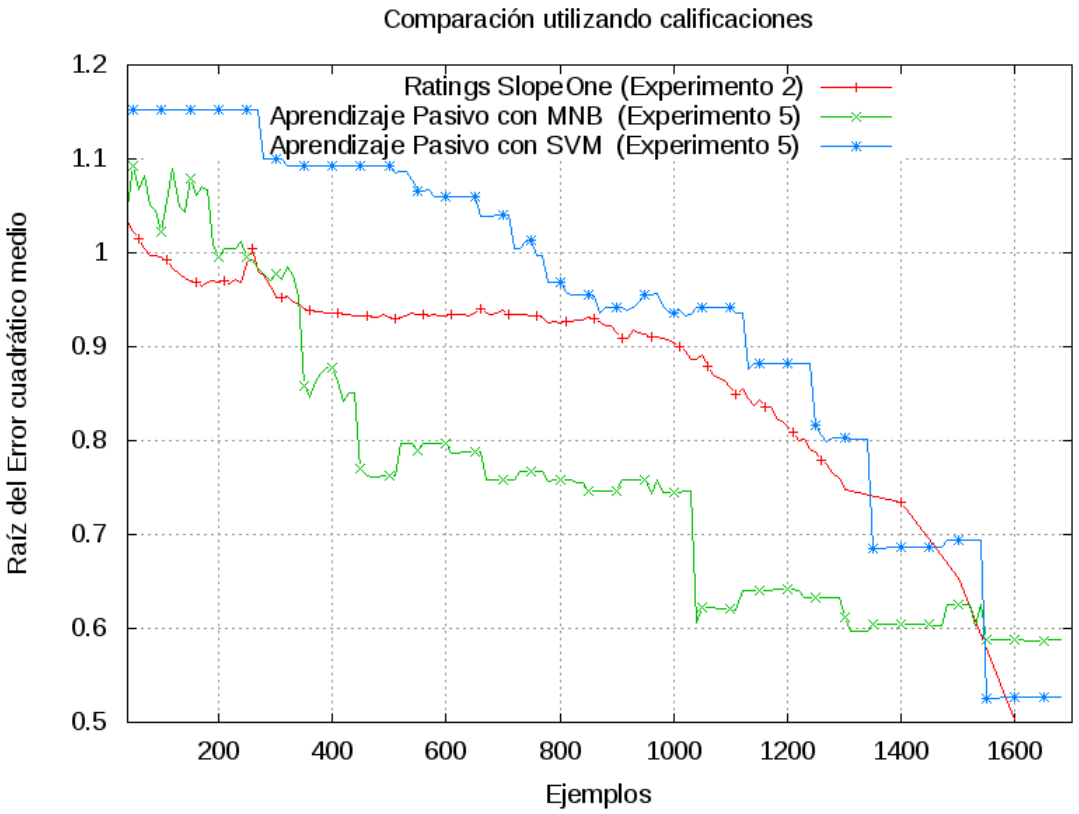

FIGURA 5.3: Utilizando calificaciones

Hipótesis La introducción de Aprendizaje Activo mejorará los resultados reduciendo el error de precisión en las primeras iteraciones al seleccionar aquella información más pertinente para mejorar la clasificación del usuario en evaluación.

Resultados La introducción de aprendizaje activo mostró algunas mejoras en los resultados, como puede verse en la Figura 5.4, siendo la diferencia con el experimento anterior de alrededor de 0.2 durante las primeras 300 iteraciones. Luego, la diferencia entre ambos tiende a reducirse.

5.1.2.3. Experimento 7: Utilizando características de ítems (usando ranking)

Hipótesis Ante los resultados del experimento 6, proponemos, al igual que cuando utilizamos correlación, realizar el experimento anterior utilizando características de los ítems en lugar de su calificación, pudiendo así lograr una mayor diferencia en cuanto a reducción del error.

Resultados En este caso, el error se mantiene prácticamente constante con un valor cercano a 1 (figura 5.2), ofreciendo un rendimiento mejor que el obtenido durante el

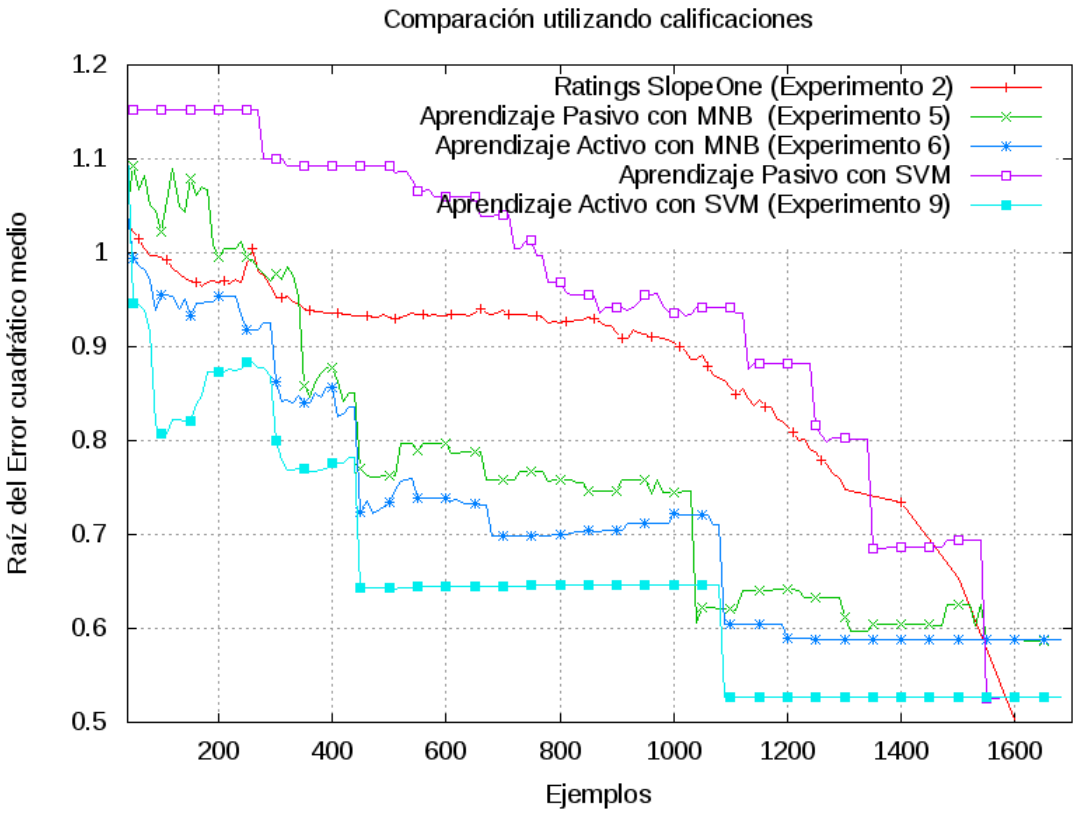

FIGURA 5.4: Comparación de Aprendizaje Activo

experimento 4. A pesar de ello, no se obtienen valores menores a los presentados en el experimento 6 (usando MNB con calificaciones).

5.1.2.4. Experimento 8: Utilizando calificaciones + SlopeOne por clase

Ante los resultados del experimento 6, que por el momento han sido los que más disminuyen el error tanto durante el comienzo con un nuevo usuario como ante el sistema en su pleno funcionamiento, surge la idea de utilizar SlopeOne internamente en cada clase para realizar las recomendaciones en lugar del uso de un ranking.

Hipótesis Aplicar SlopeOne en cada clase nos permitirá brindar una mayor precisión en las recomendaciones aprovechando las ventajas conocidas de este algoritmo.

Resultados Al ejecutar el experimento los errores promedio alcanzaban valores cercanos a 13, por lo que inmediatamente se echó por tierra nuestra suposición previa. Una posible explicación a este fenómeno consiste en que SlopeOne funciona mejor cuanto más información se tiene por ítem, por lo que al aplicarlo en un subgrupo de usuarios (los que constituyen la clase), el algoritmo sufre de este problema, fallando así sus predicciones.

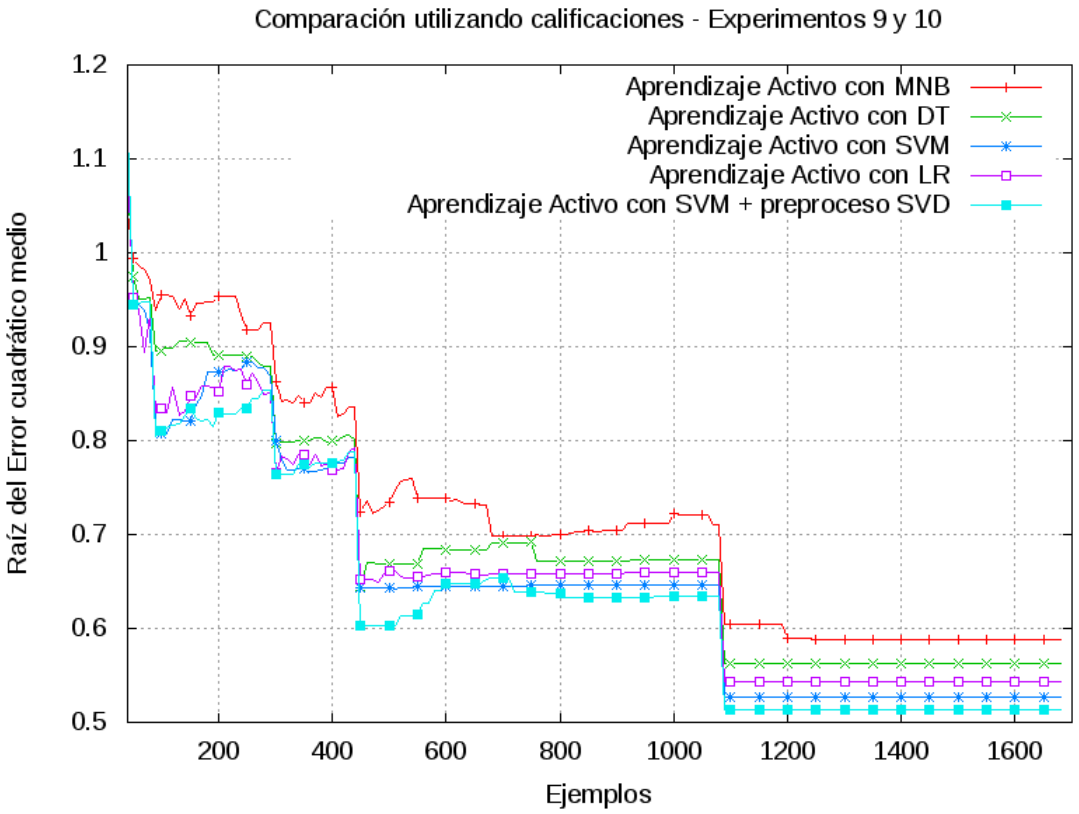

FIGURA 5.5: Aprendizaje Activo: Comparación de clasificadores

Estos resultados podrían mejorar utilizando un conjunto de datos mayor, donde tengamos mayor cantidad de usuarios por cada clase, pero se deja esta prueba como trabajo futuro puesto que son necesarias ciertas optimizaciones para mejorar la eficiencia de SlopeOne ante grandes volúmenes de datos.

5.1.2.5. Experimento 9: Utilizando calificaciones + SVM, DT ó LR

En la búsqueda por continuar optimizando resultados al máximo, hemos decidido experimentar con otros clasificadores además de MNB. Es por eso que en este experimento analizaremos los resultados obtenidos utilizando los clasificadores: Árboles de Decisión, Regresión Logística y *Support Vector Machines*.

Resultados Como puede observarse en la Figura 5.5, ninguno de los tres clasificadores presentados muestra diferencias notables con respecto a los resultados utilizando MNB, sin embargo, el que ofrece mayor precisión, principalmente durante las primeras iteraciones, es *Support Vector Machines*, con un error de alrededor de 0.8.

5.1.2.6. Experimento 10: Preproceso con LSI

Las ventajas que ofrece LSI pueden aplicarse como preproceso para mejorar el aprendizaje del clasificador, logrando así una clasificación más precisa para el usuario bajo evaluación con la consecuente disminución del error en las primeras instancias.

Sin embargo, luego de realizar esta prueba, los resultados (Figura 5.5) no ofrecieron diferencias con respecto al experimento anterior. Eso nos permite concluir que la aplicación de técnicas de reducción de dimensionalidad (LSI) o selección de características (*Feature Selection*) no influyen en la reducción del error. Sería adecuado llevar a cabo este experimento con un conjunto de datos mayor, donde quizás sí las técnicas de reducción de dimensionalidad ofrezcan diferencias en los resultados.

5.2. Experimentos añadiendo usuarios

En las evaluaciones llevadas a cabo hasta el momento hemos puesto énfasis en medir el Error Cuadrático Medio (ECM) para un usuario nuevo incorporado al sistema, con distinta cantidad de información conocida sobre ese usuario.

Por otro lado, es posible también calcular el Error Medio del modelo, lo cual nos permite obtener una idea de qué tan rápido se estabiliza el sistema a medida que se van incorporando nuevos usuarios. Para llevar a cabo este procedimiento realizamos lo siguiente:

- Iniciamos el sistema con sólo 50 usuarios, para los cuales calculamos sus ECM y finalmente promediamos para obtener el error del modelo.
- Agregamos 50 usuarios más en cada iteración hasta alcanzar la totalidad de usuarios y repetimos el cálculo realizado en el punto anterior.

Este proceso lo realizamos con los sistemas base que utilizan correlación de Pearson y SlopeOne, el que utiliza Aprendizaje Pasivo y el que utiliza Aprendizaje Activo con SVM. Los resultados pueden verse en la Figura 5.6, donde puede observarse que el menor error es el obtenido utilizando Aprendizaje Activo. Esto nos permite concluir que los sistemas de recomendación que incorporan Aprendizaje Activo tienen un menor error promedio que las demás implementaciones analizadas.

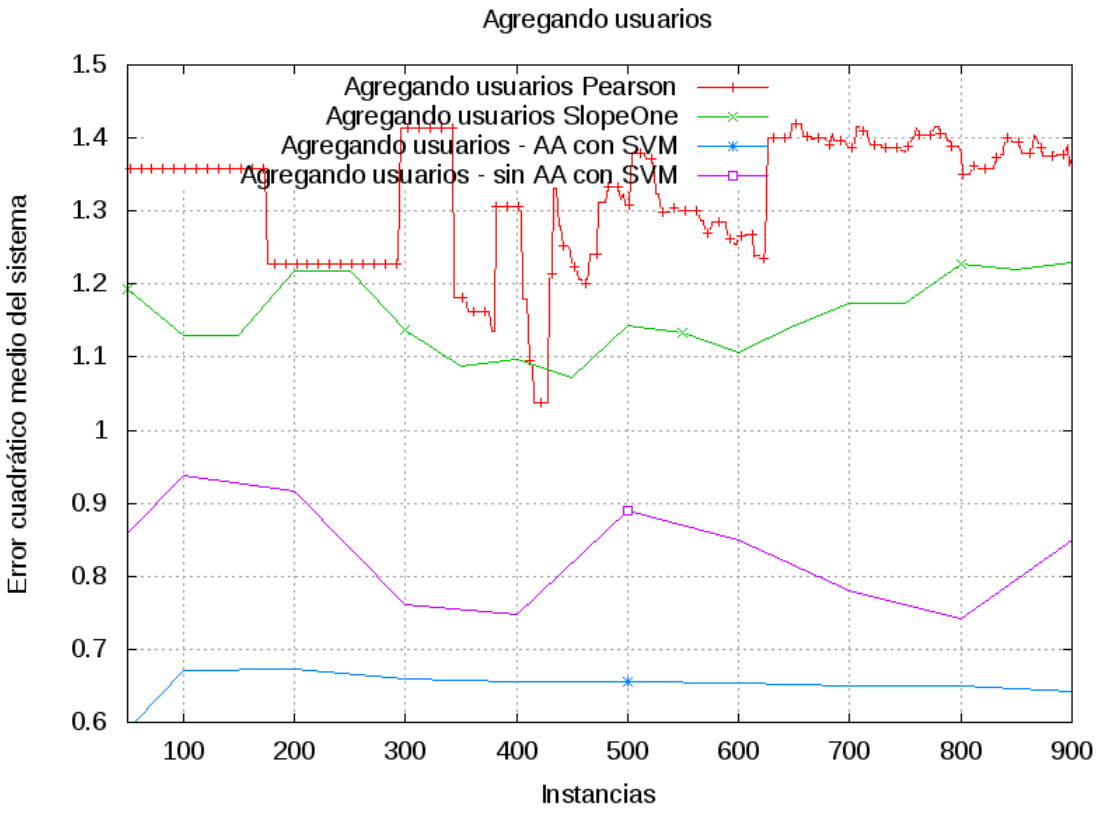

FIGURA 5.6: Error cuadrático medio del Sistema

5.3. Test de significación

En cualquier tipo de experimentos es ideal estimar la significación de los resultados, es decir, un resultado es estadísticamente significativo cuando no es probable que haya sido debido al azar. En nuestro caso, deseamos determinar si las ventajas en cuanto a reducción del error cuadrático del sistema durante su arranque en frío utilizando Aprendizaje Activo y SVM es significativo, es decir, que los resultados (y en especial, las diferencias entre los resultados) no hayan sido obtenidos azarosamente o debido a una muestra particular que los condiciona de manera benévola.

Como se analiza en *Evaluating Recommendation Systems* [24], uno de los métodos más estándares es la realización de un test de hipótesis, estableciendo un nivel de significación para luego obtener el p-valor y determinar la aceptación o rechazo de la hipótesis nula.

El test de hipótesis es un procedimiento para juzgar si una propiedad que se supone en una población estadística es compatible con lo observado en una muestra de dicha población. Para llevarlo a cabo, deben contrastarse dos hipótesis: nula y alternativa. Se denomina hipótesis nula H_0, a la hipótesis que se desea contrastar. Su nombre sugiere que debe identificarse con la hipótesis de no cambio, no diferencia, no mejora, etc. H_0

representa la hipótesis que mantendremos a no ser que los datos indiquen su falsedad. De manera explícita o implícita, la hipótesis nula se enfrenta a otra hipótesis que denominaremos hipótesis alternativa y que se denota H_1. En los casos en los que no se especifica de manera explícita, podemos considerar que ha quedado definida implícitamente como «H_0 es falsa».

Una vez planteadas las hipótesis, y dependiendo del test de hipótesis a realizar, debe determinarse un p-valor, que está definido como la probabilidad de haber llegado al resultado obtenido suponiendo que la hipótesis nula es cierta. Esto es, si el p-valor supera cierto umbral α (nivel de significación), entonces los resultados del experimento no son significativos. Cuanto menor sea el nivel de significación, más fuerte será la evidencia de que un hecho no se debe a una mera coincidencia.

Los niveles utilizados tradicionalmente son $\alpha = 0{,}05$, $\alpha = 0{,}01$ o incluso menores cuando el costo de cometer un error es alto.

Algunos de los tests comúnmente utilizados en este ámbito son: la prueba t de Student [30], la prueba de los rangos con signo de Wilcoxon [31] y la prueba U de Mann-Whitney [32] [24]. En el paper citado ([24]) se menciona que para tamaños de muestra pequeños, la prueba t de Student no es siempre la más adecuada, pues realiza asunciones sobre los datos que podrían ser incorrectas. En su lugar se recomienda utilizar la prueba de Wilcoxon.

En nuestro caso los tamaños de muestra son grandes, pues en el conjunto de datos contamos con alrededor de 1700 ítems, cuyas calificaciones utilizamos para calcular el error al realizar recomendaciones a los 1000 usuarios. Aún así, por una cuestión de prudencia y debido a las facilidades que ofrecen algunas librerías, realizaremos las tres pruebas mencionadas con anterioridad y comentaremos sus resultados.

En el desarrollo de las tres pruebas, tomaremos la hipótesis nula como: «Los resultados del sistema utilizando aprendizaje activo con SVM no son significativamente diferentes a los resultados obtenidos utilizando SlopeOne». Para realizar las pruebas tomaremos como muestras los resultados (valores del ECM a medida que incorporamos más información al sistema) de ambos experimentos mencionados en la hipótesis (experimento 2 5.1.1.2 y experimento 9 con SVM 5.1.2.5).

Luego de realizar el test t de Student, el p-valor obtenido es igual a $9{,}341 * 10^{-38}$; con la prueba de Wilcoxon, el p-valor es $4{,}326 * 10^{-23}$; y con la prueba U de Mann-Whitney, es $7{,}677 * 10^{-32}$.

En los tres casos puede observarse que, incluso tomando un nivel de significación bajo ($\alpha = 0{,}01$), el p-valor siempre cae notoriamente por debajo de dicho nivel. Esto nos

ofrece suficiente evidencia para rechazar la hipótesis nula, concluyendo así que los resultados con aprendizaje activo, los cuales ofrecieron menor error cuadrático con respecto a SlopeOne, son significativos.

Capítulo 6

Conclusiones y trabajo futuro

Luego de realizar los experimentos presentados en el capítulo anterior, aquí retomaremos las metas propuestas en la introducción para evaluar su alcance con los resultados obtenidos.

Nuestro principal objetivo a lo largo de este trabajo fue reducir el tiempo de aprendizaje del sistema a través de la maximización de la utilidad de información aportada por el usuario. En el capítulo anterior analizamos que el mejor desempeño del sistema durante el arranque en frío se logra utilizando aprendizaje activo aplicado a las calificaciones de los ítems en el sistema. Previamente habíamos llevado nuestro problema de recomendación a un problema de clasificación, con el objetivo de clasificar a un usuario según sus preferencias en el grupo que mejor se ajuste a ellas, al mismo tiempo que redujimos la complejidad computacional al evitar la exploración de la totalidad del espacio de búsqueda.

La aplicación de aprendizaje activo brindó mejoras significativas con respecto a SlopeOne y al aprendizaje pasivo, principalmente en dos aspectos a destacar:

- Reducción del error medio al recomendar ítems a un usuario nuevo en un sistema maduro.
- Reducción del error medio del sistema durante su arranque inicial, donde cuenta con una cantidad de usuarios muy baja como para brindar un buen análisis de semejanzas.

Esto nos permite llegar a la conclusión de que la aplicación de técnicas de Aprendizaje Activo en sistemas de recomendación es una buena forma de reducir los problemas del arranque en frío, ofreciendo además algunas ventajas a considerar:

- Utilizar un sistema de clasificación permite reducir el espacio de búsqueda tanto al momento de generar recomendaciones como al de detectar las similitudes entre usuarios.
- El proceso de generación de clases y determinación de ítems o características con mayor ganancia de información puede realizarse de manera *offline*, reduciendo los procedimientos a ejecutar en tiempo real ante una solicitud de recomendación.

Como trabajo e investigación futura, proponemos las siguientes líneas a considerar:

- **Ejecutar los experimentos con un conjunto de datos más grande:** Debido a limitaciones computacionales, los experimentos del presente trabajo fueron realizados utilizando un conjunto de datos relativamente pequeño comparado al tamaño que puede tener un sistema real. Por lo tanto, utilizar una mayor cantidad de datos permitiría realizar un análisis más detallado.
- **Ejecutar los experimentos sobre un sistema en producción:** Crear un sistema desde cero, o implementar el ciclo de aprendizaje activo en un sistema ya en funcionamiento permitiría evaluar el rendimiento en un entorno real, pudiendo medir así características como la eficiencia o rapidez debido a la ventaja de cómputo *offline* que permite esta implementación.
- **Utilizar una combinación de clasificadores:** Como se menciona en *Active Learning for Recommender Systems* [10], en algunos un tipo de clasificador puede modelar mejor las preferencias de un usuario, mientras que otro puede mejorar la precisión al modelar otro perfil de usuarios. En estos casos podría ser conveniente, según las elecciones del usuario, escoger el clasificador que mejor se adapte a su perfil.

Bibliografía

[1] F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor. *Recommender Systems Handbook*. Springer, 2010. ISBN 9780387858203. URL https://books.google.com.ar/books?id=2hFG21Cp2qcC.

[2] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering.

[3] Netflix. . URL http://www.netflix.com/.

[4] Netflix prize. . URL http://www.netflixprize.com/.

[5] Association for computing machinery. URL http://www.acm.org.

[6] Recsys. URL http://www.recsys.acm.org.

[7] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[8] Andrew McCallum Gregory Druck, Burr Settles. Active learning by labeling features.

[9] Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. 2011.

[10] Neil Rubens, Dain Kaplan, and Masashi Sugiyama. Active learning in recommender systems. In P.B. Kantor, F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 735–767. Springer, 2011. doi: 10.1007/978-0-387-85820-3_23.

[11] Coeficiente de correlación de pearson. URL http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient.

[12] Wikipedia. Knn algorithm, 2014. URL http://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=635197932. [Online; accessed 22-February-2015].

[13] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, February 2011. ISSN 1551-3955. doi: 10.1561/1100000009. URL `http://dx.doi.org/10.1561/1100000009`.

[14] Toby Segaran. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly, Sebastopol, CA, USA, 2007. ISBN 0-596-52932-5.

[15] Long tail. URL `http://en.wikipedia.org/wiki/Long_tail`.

[16] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[17] Ethem Alpaydin. *Introduction to Machine Learning*. O'Reilly, 2010. ISBN 978-0-262-01243-0.

[18] George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305, 2003.

[19] Activepipe. URL `https://github.com/mit0110/activepipe`.

[20] Clasificador naive bayes. URL `http://en.wikipedia.org/wiki/Naive_Bayes_classifier`.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[23] Movielens. URL `http://grouplens.org/datasets/movielens/`.

[24] Guy Shani and Asela Gunawardana. Evaluating recommender systems. Technical Report MSR-TR-2009-159, November 2009. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=115396`.

[25] Dietmar Jannach and Gerhard Friedrich. Td3: Recommender systems, August 2013. http://ijcai13.org/program/tutorial/TD3.

[26] K-means clustering. URL `http://en.wikipedia.org/wiki/K-means_clustering`.

[27] Distribución de pareto. URL http://en.wikipedia.org/wiki/Pareto_distribution.

[28] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010. ISBN 9780136042594. URL http://books.google.com.ar/books?id=8jZBksh-bUMC.

[29] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. ISBN 9780521780193. URL http://books.google.com.ar/books?id=B-Y88GdO1yYC.

[30] R. Mankiewicz. *The Story of Mathematics*. Mathematics (Princeton University Press). Princeton University Press, 2004. ISBN 9780691120461. URL http://books.google.com.ar/books?id=JXxrpwAACAAJ.

[31] S. Siegel. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill series in psychology. McGraw-Hill, 1956. URL http://books.google.com.ar/books?id=ebfRAAAAMAAJ.

[32] Michael P. Fay and Michael A. Proschan. Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statist. Surv.*, 4:1–39, 2010. doi: 10.1214/09-SS051. URL http://dx.doi.org/10.1214/09-SS051.