UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Aplicación de la Teoría de Juegos de Utilidad Transferible a los Sistemas de Recomendación a Grupos

Juan de Dios Romero Palop

Tutor: Fernando Díez Rubio

Julio 2015

Aplicación de la Teoría de Juegos de Utilidad Transferible a los Sistemas de Recomendación a Grupos

AUTOR: Juan de Dios Romero Palop

TUTOR: Fernando Díez Rubio

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Julio de 2015

RESUMEN

La teoría de juegos y los sistemas de recomendación son dos áreas de conocimiento cuya colaboración no ha sido estudiada en profundidad hasta el momento. El objetivo de este trabajo es analizar la posibilidad de utilizar resultados de teoría de juegos para generar nuevos sistemas de recomendación colaborativos. Para ello, el primer paso llevado a cabo consiste en identificar los conceptos comunes que permitan adaptar los resultados de la teoría de juegos para generar recomendaciones. Una vez hecho esto, se presenta una propuesta para la construcción de un sistema de recomendación colaborativo a partir de técnicas propias de la teoría de juegos cooperativos. El método propuesto se basa en la idea de la formación incremental de coaliciones utilizada por Lloyd S. Shapley para describir su solución del problema de reparto. De esta manera, el sistema lleva a cabo un proceso iterativo que, en cada ejecución, agrupa a los usuarios que mayor grado de similitud tengan entre sí. Así mismo, para medir el grado de similitud entre usuarios se ha combinado el concepto de función característica, propio de los juegos cooperativos, con los conceptos de métrica y distancia entre clusters de los sistemas de recomendación. Para poder probar la validez del método descrito, se ha implementado un caso práctico consistente en un sistema de recomendación de películas basado en el conjunto de datos CAMRa2010 Filmtipset Social. Aunque las implementaciones del método propuesto no mejoran los resultados obtenidos por los métodos tradicionales, sí que se han obtenido tasas de error muy similares y, por lo tanto, se puede concluir que es razonable profundizar en la idea de combinar estas dos áreas utilizando el método propuesto como punto de partida para ello.

PALABRAS CLAVE

Teoría de Juegos, Sistemas de Recomendación, Clustering, Función Característica, Juegos Cooperativos.

ABSTRACT

Game theory and recommendation systems are two areas of knowledge whose cooperation has not been studied in depth so far. The aim of this paper is to analyze the possibility of using game theory results to generate new recommendation systems. To do so, the first step involves identifying common concepts in order to use them as grounds for applying results of game theory to the recommendation process. Once this is done, a proposal to build a collaborative recommendation system based on cooperative games theory techniques is presented. The proposed method is based on the idea of incremental coalition formation used by Lloyd S. Shapley in order to describe his solution to the fair division problem. Thereby, the system carries out an iterative process that, in each iteration, groups together those users with the greatest similarity between them. Also, to measure the degree of similarity between users, the characteristic function concept from game theory has been joined with the recommendation concepts of metric and distance between clusters. In order to test the validity of the described method, we have implemented a case consisting of a movie recommendation system based on the CAMRa2010 Social Filmtipset dataset. Although the implementations of the proposed method do not improve the results obtained by traditional methods, the error rates are very similar and, therefore, we can conclude that it is reasonable to go further with the idea of combining these two areas using the designed method as starting point for this purpose.

KEYWORDS

Game Theory, Recommendation Systems, Clustering, Characteristic Function, Cooperative Games .

A Fernando, por haber sido el guía perfecto.

Al Dr. Pedro Campos, por su colaboración desinteresada.

A María, por ser mi soporte en los malos momentos.

Y a mis padres porque sin ellos nada de esto hubiese sido posible.

TABLA DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

1. INTRODUCCIÓN

A principios del siglo XX, la teoría de juegos pasó de ser considerada una parte recreativa de las matemáticas a ser utilizada como herramienta de resolución de gran cantidad de problemas en diversos ámbitos como la economía, la biología o la política. En cualquiera de ellos, surgen problemas relacionados con la repartición de recursos y la selección de estrategias para obtener el mayor beneficio posible. La teoría de juegos permite modelizar estas situaciones de conflicto para ayudar a cada empresa, individuo o partido político en la toma de decisiones. Dentro de las distintas caracterizaciones de escenarios definidas por la teoría de juegos, aparecen aquellos en las que los jugadores no tienen comportamientos totalmente antagónicos, sino que pueden colaborar para obtener un mayor beneficio a repartir. Estos escenarios son conocidos como juegos cooperativos**.** Si, además, no existe ningún tipo de restricción a la hora de cooperar se habla de **juegos cooperativos de utilidad transferible (TU Games)**.

Por otro lado, la gran cantidad de datos generada diariamente por los usuarios de los sistemas informáticos ha hecho que el análisis de esta información tenga una importancia capital para los proveedores de servicios ya que les permite tener mayor conocimiento sobre sus usuarios, mejorar su experiencia y, en definitiva, obtener mayores réditos. Una de las herramientas basadas en este tipo de análisis que mayor desarrollo está teniendo en los últimos tiempos son los **sistemas de recomendación**. Básicamente, estas herramientas se ocupan de analizar los datos relativos al comportamiento de los usuarios al utilizar los sistemas, para poder recomendarles nuevos servicios que puedan ser de su interés. Hasta hace algunos años, las técnicas más habituales utilizadas para efectuar recomendaciones consistían en determinar patrones de comportamiento de cada usuario individualmente (*e.g.* características de los productos comprados, productos visualizados, etc.) para inferir las características que debía tener un producto para ser recomendado. Sin embargo, en los últimos tiempos ha ganado peso la corriente de intentar aumentar el alcance del estudio pasando de utilizar únicamente los datos del usuario que va a recibir la recomendación a utilizar también datos del resto de usuarios poniéndolos en el contexto adecuado. Es decir, intentar buscar relaciones entre usuarios (gustos similares, amistad, mismo rango de edad, etc.) de forma que la cantidad de información a la hora de recomendar aumente y esto repercuta positivamente en el número y la calidad de las recomendaciones. Este tipo de estrategias se conocen como **algoritmos de recomendación colaborativos**.

En este punto cabría preguntarse la utilidad de **construir un sistema de recomendación colaborativo basado en la modelización de un problema de teoría de juegos cooperativos de utilidad transferible**, objetivo de este Trabajo de Fin de Grado. No se tiene constancia de la existencia de ningún trabajo previo que utilice esta metodología, aunque en los últimos años sí se ha comenzado a estudiar la utilización de técnicas propias de la teoría de juegos en el análisis de datos. En concreto estos estudios se han centrado en la utilización de la teoría de juegos para evaluar qué atributos de un conjunto de datos proporcionan mayor cantidad de información y, por tanto, deben ser utilizados para llevar a cabo los análisis. Por ejemplo, (Wang & Tseng, 2011) utilizan estas herramientas para optimizar el proceso de configuración de búsquedas en la web por parte de usuarios con el

objetivo de minimizar el número de campos que el usuario debe rellenar para realizar una búsqueda eficaz y eficiente. En la misma línea, (Littlechild & Thompson, 1977) intentan determinar cuál sería el valor óptimo de la tasa de aterrizaje de cada aeropuerto de forma que se repartiese el flujo aéreo, optimizándose los recursos y evitando congestiones. Mientras que, por su parte, (Torkaman, Moghaddam, Aghaeipour, & Hajati, 2009) estudian qué agrupación de marcadores celulares tiene mayor repercusión a la hora de diagnosticar la leucemia.

1. 1. OBJETIVOS

De acuerdo al contexto descrito en la sección anterior, los principales objetivos de este trabajo son:

1. Determinar de qué manera se pueden aplicar las técnicas propias de la teoría de juegos a la hora de construir un sistema de recomendación.
2. Diseñar un sistema de recomendación colaborativo que se base en la teoría de juegos y que permita comprobar si la idea de juntar estas dos áreas tiene fundamento práctico, teniendo en cuenta los requisitos de rendimiento de estos sistemas.
3. Realizar una búsqueda para identificar las tecnologías existentes que puedan ser útiles para implementar el sistema propuesto y, en el caso de no encontrar ninguna que se adapte a los requisitos, desarrollar las funciones necesarias para ello.
4. Implementar un sistema de recomendación de películas a partir del conjunto de datos *CAMRa2010 Filmtipset Social*[1] basado en el diseño propuesto.
5. Analizar los resultados obtenidos y compararlos con los de algún sistema de recomendación tradicional para probar la validez del diseño.

Para llevar a cabo este trabajo se han utilizado los conocimientos obtenidos en las siguientes asignaturas del doble Grado:

- Estadística I (utilizada en los objetivos 2 y 5)
- Inteligencia Artificial (1, 2, 3 y 4)
- Estructuras Discretas y Lógica (1 y 2)
- Probabilidad I (1 y 2)
- Estructuras de Datos (4)
- Análisis de Algoritmos (2 y 3)
- *Social Data Modelling* (Optativa Erasmus) (4 y 5)

1. 2. ESTRUCTURA DE LA MEMORIA

Esta memoria se estructura en cuatro capítulos. En el primero se presentan los conceptos teóricos básicos relativos a la teoría de juegos y a los sistemas de recomendación así como

[1] http://camra2010.dai-labor.de/

el conjunto de datos que se va a utilizar para la implementación práctica. En el segundo capítulo se realiza un análisis en profundidad de la estructura del conjunto de datos con el fin de adaptar el método propuesto a las características del conjunto y a la cantidad de recursos disponibles. Una vez hecho esto, se describe el método propuesto y su implementación dedicando una sección al estudio de las tecnologías existentes. En el tercer capítulo, se muestran los resultados obtenidos junto a un análisis de los mismos y, finalmente, se presentan las conclusiones finales del estudio y las líneas de trabajo futuras.

2. CONCEPTOS TEÓRICOS BÁSICOS

En esta sección se presentan los conocimientos básicos sobre teoría de juegos y sistemas de recomendación que serán utilizados posteriormente para el diseño de la solución propuesta. Así mismo, se introduce el conjunto de datos que será utilizado en la parte práctica del estudio.

2. 1. TEORÍA DE JUEGOS

La teoría de juegos es una disciplina de las matemáticas encargada de estudiar situaciones de conflicto o colaboración entre agentes y las posibles estrategias a seguir por estos para conseguir un objetivo propuesto. Para ello, utiliza una serie de elementos que sirven para caracterizar cada una de estas estrategias: un conjunto (normalmente finito) de jugadores, una lista finita de posibles acciones que puede llevar a cabo cada uno de los jugadores y una lista que asigna un valor de recompensa para cada jugador a cada combinación válida de acciones. Por ejemplo, el ajedrez podría ser modelizado como un juego de dos jugadores, cada uno de los cuales puede realizar una serie de movimientos de sus piezas que resultan en distintos estados del tablero, cada uno de los cuales con un valor para cada jugador en tanto en cuanto le acerque o le aleje de la victoria.

Dependiendo de sus características, los juegos pueden clasificarse en:

- **Juegos bipersonales y *n*-personales:** según el número de jugadores que intervienen en él. Muchos de los estudios realizados hasta el momento han partido de juegos de dos jugadores, extrapolando más tarde los resultados a un número finito de ellos.

- **Juegos de suma cero y de suma distinta de cero:** dependiendo de si el beneficio que recibe un jugador es igual a la pérdida en que incurre otro, es decir, la suma de las recompensas de los jugadores en cualquier estado es o no cero.
- **Juegos cooperativos y juegos competitivos:** según el tipo de relación que tengan los jugadores. Los juegos cooperativos permiten colaboraciones entre ellos para obtener beneficios comunes mientras que los competitivos no permiten este tipo de acciones. Por su interés para este estudio a continuación se incluye información adicional sobre los juegos cooperativos.

2. 1. 1. JUEGOS COOPERATIVOS

Como se ha indicado anteriormente, los juegos cooperativos son aquellos que permiten colaboraciones entre los usuarios para obtener beneficios, pudiendo establecerse o no restricciones a la hora de construir coaliciones válidas y repartir los beneficios obtenidos al cooperar. Así, aquellos juegos cooperativos en los que la recompensa puede ser dividida sin restricciones se conocen como **juegos de utilidad transferible** mientras que aquellos en los que existen algunas restricciones (*e.g.* simpatías o desavenencias entre jugadores pueden aumentar o eliminar las posibilidades de formación de coaliciones) se conocen como **juegos de utilidad no transferible**. Un ejemplo de juegos de utilidad no

transferible son las coaliciones políticas en los que la divergencia de ideas entre los partidos impide ciertas colaboraciones.

En términos matemáticos, un juego cooperativo de *n* jugadores queda totalmente definido por un par (N, ν) siendo N ={1,2,3,…,*n*} el **conjunto finito de jugadores** y **ν**: $\mathbf{2^n \longrightarrow \mathbb{R}}$**,** $\nu(\emptyset) = 0$, la **función característica** que asigna un valor a cada uno de los 2^n subconjuntos de N conocido como **coalición**. El valor asignado por la función característica determina el beneficio total obtenido por la cooperación de todos los jugadores que debe ser repartido entre ellos.

2. 1. 1. 1. PROPIEDADES DE LOS JUEGOS COOPERATIVOS

Existen tres propiedades principales que describen la naturaleza de la función característica y, por tanto, de un juego cooperativo: monotonía, superaditividad y convexidad.

Un juego se dice **monótono** si $\nu(A) \leq \nu(B)$ para cualesquiera coaliciones A, B tales que $A \subseteq B$. Esto es, cuantos más jugadores se unen a una coalición, mayor es el beneficio que la coalición genera. Adicionalmente, un juego se define como **simple** si es monótono y además $\nu(A) = \{0,1\}$ para toda coalición A.

Un juego se dice **superaditivo** si dadas dos coaliciones cualesquiera *A, B* disjuntas ($A \cap B = \emptyset$) se cumple que $\nu(A \cup B) \geq \nu(A) + \nu(B)$. Es decir, el beneficio de formar una coalición es mayor que la suma de los beneficios de trabajar por separado.

Un juego se dice **convexo** si para todo par de coaliciones *A, B* y para cualquier jugador *i* tales que $A \subset B \subset N \backslash \{i\}$ se cumple que $\nu(B \cup \{i\}) - \nu(B) \geq \nu(A \cup \{i\}) - \nu(A)$. Esto es, la contribución de un jugador a una coalición es creciente al aumentar el tamaño de la coalición. La propiedad de convexidad modela el comportamiento conocido como bola de nieve o efecto llamada. Cabe resaltar que la superaditividad es una condición necesaria para la convexidad.

2. 1. 2. EL PROBLEMA DE REPARTO

A lo largo de la historia, al estudiar los juegos cooperativos se ha trabajado con la premisa de que **todos** los jugadores acabarían cooperando y se han invertido los esfuerzos en determinar la forma más justa, en términos de recibir la parte proporcional a lo que el jugador aporta, de repartir los beneficios obtenidos al formar esa gran coalición. Para ello, se introduce un concepto adicional conocido como **reparto** que modeliza el beneficio individual que recibe cada jugador participante en la coalición total. En términos matemáticos, el reparto es un vector de *n* componentes, $x = (x_1, x_2, \dots, x_n)$, cada una de las cuales representa el beneficio obtenido por el jugador correspondiente. Además, se han definido una serie de propiedades para intentar describir un reparto justo y que, idealmente, el algoritmo de repartición debe cumplir:

- **Eficiencia**: el algoritmo debe repartir la totalidad del beneficio asignado a la coalición por la función característica. $(\sum_{i \in N} x_i = \nu(N))$

- **Racionalidad individual**: el beneficio de cada jugador en el reparto debe ser mayor que el que obtendría si no colaborase y participase en el juego individualmente. ($x_i \geq v(\{i\})\ \forall i$)
- **Existencia**: el algoritmo debe ser capaz de realizar un reparto justo para cualquier escenario.
- **Unicidad**: debe existir un único reparto justo.
- **Computacionalmente calculable**: el algoritmo debe obtener una solución en tiempo polinómico con respecto al número de jugadores.
- **Simetría**: dos jugadores *i, j* se consideran simétricos si se cumple que $v(A \cup \{i\}) = v(A \cup \{j\})\ \forall A \in P(N)$. Un reparto es simétrico si asigna el mismo beneficio a cualquier pareja de jugadores simétricos.
- **Linealidad**: un algoritmo de repartición se dice lineal si para dos juegos distintos sobre un mismo conjunto de jugadores (N, v) y (N, μ) se cumple que la suma de los beneficios de un jugador en cada juego por separado es igual al beneficio del jugador al aplicar el algoritmo sobre la suma de los valores de las funciones características: $\phi_i(v + \mu) = \phi_i(v) + \phi_i(\mu)$
- **Jugador nulo**: un reparto cumple esta propiedad si a cualquier jugador *i* que no aporta nada a ninguna coalición le asigna un beneficio nulo ($v(A \cup \{i\}) = v(A)\ \forall A \in P(N) \Rightarrow x_i = 0$).

El conjunto de repartos de un juego que cumplen las propiedades de eficiencia y racionalidad individual se denomina **conjunto de imputaciones** y se define de la siguiente manera:

$$I(N, v) = \{x = (x_1, x_2, \dots, x_n) | \sum_{i \in N} x_i = v(N)\,; x_i \geq v(\{i\})\ \forall i\}$$

2. 1. 3. SOLUCIONES AL PROBLEMA DE REPARTO. EL VALOR DE SHAPLEY

Para resolver el problema de reparto descrito en la sección anterior se define el concepto de **solución para un juego cooperativo** como una regla, algoritmo o fórmula que determina cómo obtener repartos justos basándose en algunas de las propiedades indicadas anteriormente. Existen gran cantidad de soluciones para juegos cooperativos entre las que destacan las siguientes:

- **Los conjuntos estables de Von Neumann y Morgenstern**: en (Von Neumann & Morgenstern, 1944) los autores definen la relación de **dominación** entre imputaciones de la siguiente manera: sean x e y dos imputaciones para una coalición A, se dice que x domina a y si $x_i > y_i\ \forall i \in A$ y además $\sum_{i \in A} x_i \leq v(A)$. Si estas dos condiciones se cumplen, los jugadores podrían abandonar la coalición total en el caso de realizarse la repartición usando la imputación *y* ya que podrían obtener, al menos, el mismo beneficio que les correspondería con la imputación *x* trabajando individualmente. Basándose en esta relación, se define el **conjunto estable** como el conjunto de imputaciones que cumple que todas aquellas que no forman parte del conjunto son dominadas por al menos una imputación del

conjunto (estabilidad externa) y además no existe ninguna relación de dominación entre los elementos del conjunto (estabilidad interna). De este modo, cualquiera de las imputaciones que forma parte del conjunto estable es considerada una solución justa para el problema de reparto.

- ***Core*****:** aunque el concepto de *core* (núcleo) de un juego cooperativo ya había sido introducido previamente, fue el matemático canadiense D. B. Gillies quien en (Gillies, 1959) acuñó la definición moderna de *core* adaptándola a juegos de suma distinta de cero. Al igual que en los conjuntos estables, el *core* es un conjunto de repartos definido de la siguiente manera: $C(v) = \{x = (x_1, x_2, \dots, x_n) | \sum_{i \in N} x_i = v(N) \, ; \sum_{i \in A} x_i = v(A) \, \forall A \subseteq N\}$. Los elementos del *core* se consideran justos ya que ninguno de sus componentes podría obtener un beneficio mayor abandonando la coalición.

Sin embargo, de todas las soluciones de juegos cooperativos propuestas a lo largo de la historia, la de mayor éxito es la conocida como **valor de Shapley,** en honor a su creador Lloyd S. Shapley, matemático y economista estadounidense. Esta solución fue introducida por primera vez en (Shapley, 1953) y se basa en intentar calcular qué grado de participación tiene cada jugador en la coalición. Para ello, utiliza el concepto de **contribución marginal** presentado a continuación:

Dado un juego cooperativo (N, v), y suponiendo la coalición total, la contribución marginal del jugador *i* al beneficio total $v(N)$ viene determinado por $v(N) - v(N \backslash \{i\})$.

El valor de Shapley se sustenta en la idea de que la coalición total se forma de manera **incremental**, es decir, al comienzo se coaligan dos jugadores, más tarde se puede crear otra coalición de dos jugadores o unirse otro a la creada en el paso anterior y así hasta formar la coalición total. En este proceso, en los pasos intermedios podrían generarse todas las coaliciones correspondientes a las permutaciones de los n jugadores que participan y cada una de ellas tendría asignado un valor de la función característica. De este modo, el valor de Shapley asigna un beneficio a cada jugador teniendo en cuenta cuanto aportaría ese jugador a cada una de las coaliciones intermedias posibles, quedando definido de la siguiente manera:

Sea (N, v) un juego cooperativo, el **valor de Shapley** para el jugador *i* es:

$$\phi_i(v) = \sum_{A \subseteq N \backslash \{i\}} \frac{|A|! * (n - |A| - 1)!}{n!} (v(A \cup \{i\}) - v(A))$$

siendo |A| el número de jugadores participantes en la coalición A y n el número de jugadores participantes en el juego (n = |N|). A continuación se presenta un ejemplo sencillo de aplicación del valor de Shapley.

Ejemplo *Tres ciudades (Ávila, Burgos y Córdoba) pretenden que un investigador extranjero imparta un curso en cada una de ellas. Para ello, cooperan para hacerse cargo de los costes de transporte del investigador. Suponiendo que los costes de transporte entre cada una de las tres ciudades sean los indicados en la siguiente tabla, ¿qué parte del coste total de transporte debe corresponder a cada una?*

	{A}	{B}	{C}	{A,B}	{A,C}	{B,C}	{A,B,C}
coste	1500	1600	1900	1600	2900	3000	3000

Esta situación puede modelizarse como un juego cooperativo de tres jugadores cuya función característica asigna a cada coalición el valor correspondiente a la diferencia de coste que habría entre estas dos situaciones, si se formase la coalición o si el investigador visitase las mismas ciudades individualmente. De esta forma, la función característica toma los siguientes valores:

	{A}	{B}	{C}	{A,B}	{A,C}	{B,C}	{A,B,C}
$\boldsymbol{v}$	0	0	0	1500	500	500	2000

Si suponemos que las tres ciudades acaban cooperando, el siguiente paso es calcular qué parte del beneficio obtenido corresponde a cada ciudad, para poder calcular qué coste debe asumir cada una. Para ello calculamos el valor de Shapley para cada jugador y restamos el beneficio individual obtenido al coste de cada ciudad:

$$\phi_A(v) = \frac{|\{B\}|! * (n - |\{B\}| - 1)!}{n!}\big(v(\{A,B\}) - v(\{B\})\big) + \frac{|\{C\}|! * (n - |\{C\}| - 1)!}{n!}\big(v(\{A,C\}) - v(\{C\})\big) + \frac{|\{B,C\}|! * (n - |\{B,C\}| - 1)!}{n!}\big(v(\{A,B,C\}) - v(\{B,C\})\big)$$

$$= \frac{1! * 1!}{3!}(1500 - 0) + \frac{1! * 1!}{3!}(500 - 0) + \frac{2! * 0!}{3!}(2000 - 500)$$

$$= \frac{1500}{6} + \frac{500}{6} + \frac{3000}{6} = 833.33 \Rightarrow \boldsymbol{coste_A} = 1500 - 833.33 = \mathbf{666.67}$$

$$\phi_B(v) = 833.33 \Rightarrow \boldsymbol{coste_B} = 1600 - 833.33 = \mathbf{766.67}$$

$$\phi_C(v) = 333.33 \Rightarrow \boldsymbol{coste_C} = 1900 - 333.33 = \mathbf{1566.67}$$

Con respecto a las propiedades indicadas anteriormente, el valor de Shapley es el único reparto posible que es eficiente, simétrico, lineal y asigna beneficio nulo a jugadores nulos.

2. 2. SISTEMAS DE RECOMENDACIÓN

Los sistemas de recomendación son herramientas surgidas a partir del análisis de datos y que en la actualidad son imprescindibles para guiar a los usuarios entre la cantidad ingente de datos que manejan. Algunos ejemplos cotidianos en los que se aplican estas herramientas son las recomendaciones de productos que realiza Amazon o de videos que realiza Youtube. Dado que la cantidad de áreas en los que se aplican estos sistemas es muy amplia, se hablará de **usuarios** para referirse a los receptores de la recomendación, independientemente de que sean visualizadores de películas o videos, compradores de artículos o usuarios de buscadores web. Así mismo, se denominará **productos** a los elementos que van a ser recomendados ya sean películas, videos, textos de búsqueda o prendas de vestir.

La base del funcionamiento de los sistemas de recomendación es el procesamiento tanto de la información disponible sobre los perfiles de usuario como de las características de los productos que pueden ser recomendados e intentar encontrar conexiones entre ellos. Estas conexiones pueden ser relaciones básicas entre los productos (películas del mismo género, ropa de la misma marca, etc.), entre los usuarios (películas que han visto personas de la misma edad, productos comprados por gente de la misma nacionalidad) o pueden ir un paso más allá y buscar relaciones menos obvias o mezclar relaciones de ambos tipos. Puede encontrarse una descripción extensa y detallada de este tipo de sistemas en (Ricci, Rokach, Shapira, & Kantor, 2011). Teniendo en cuenta esto, se distinguen seis tipos principales de sistemas de recomendación:

- **Basados en contenido**: son aquellos que analizan los datos de los productos comprados, vistos o puntuados por los usuarios previamente y buscan otros productos que tengan características parecidas para ser recomendados. Durante mucho tiempo este tipo de sistemas han sido los más populares pero tienen un problema grave: la ***sobre-especialización***. Este término hace referencia a la restricción a productos similares que se impone y que choca con la amplitud de comportamientos que suelen desarrollar los seres humanos. Por ejemplo, si un usuario únicamente ha comprado trajes desde que comenzó a utilizar el sistema no quiere decir que solo esté interesado en este tipo de productos pero al aplicar recomendación basada en contenido solo se le recomendarán más trajes. Un ejemplo de recomendador que utiliza que utiliza esta estrategia es YouTube.
- **Demográficos**: estos sistemas clasifican a los usuarios según datos puramente demográficos: edad, sexo, estudios, lugar de residencia, etc. Una vez hecho esto, basta con generar una serie de estereotipos que relacionen estas agrupaciones con determinados productos. Dado que la cantidad de datos de cada individuo que se manejan en estos sistemas es mucho menor, su complejidad computacional es también mucho menor que en el resto. Sin embargo, tienen dos problemas principales: los usuarios suelen mostrarse reacios a proporcionar este tipo de datos y los resultados obtenidos no son tan personalizados. El recomendador de libros Grundy (Rich, 1979) fue el primer sistema que utilizó estereotipos demográficos para sugerir lecturas.
- **Colaborativos**: en estos sistemas se agrupan los usuarios según sus gustos comunes y se utiliza la información del resto de usuarios del grupo para generar recomendaciones. La clave del buen funcionamiento de este tipo de recomendadores es utilizar un buen algoritmo de agrupamiento (***clustering***). Un ejemplo de sistema de recomendación colaborativo es el utilizado por FilmAffinity, servicio web de recomendación de películas y series. Por su interés para este estudio, a continuación se incluye una explicación más detallada de los mismos.
- **Basados en conocimiento**: estos sistemas se basan en un conocimiento más exhaustivo de los productos que ofrecen para poder llevar a cabo un proceso de correspondencia entre productos y contextos de usuario. Es decir, se describen una serie de problemas que el usuario va a querer resolver utilizando el servicio web y se asigna un conjunto de productos a cada situación. Una vez hecho esto, basta con que el usuario describa de alguna manera el problema que quiere resolver para que la herramienta proceda a recomendarle productos. En este caso,

la necesidad de datos para realizar la recomendación es mínima pero debe invertirse mucho esfuerzo en la descripción de los problemas y en los procesos de asignación. El recomendador de coches alemán *MyProductAdvisor* es un claro ejemplo de este tipo de sistemas. A través de una pequeña encuesta obtienen la descripción del problema del usuario (marca, número de plazas, tipo de coche, etc.) y le proporcionan sugerencias sobre el modelo que debe adquirir.

- **Basados en comunidades:** estos sistemas agrupan a los usuarios en comunidades utilizando criterios de amistad y utilizan los datos de todos los usuarios de la comunidad para generar recomendaciones. Se trata de un punto de vista similar al de los sistemas colaborativos con la diferencia de criterio a la hora de agrupar usuarios. Obviamente, los recursos utilizados en estos sistemas son ínfimos ya que únicamente necesitan tener acceso a alguna red social pero en general sus resultados son peores ya que en muchos casos una relación de amistad no implica gustos similares (Golbeck, 2006).
- **Híbridos o mixtos**: como se ha indicado, todos los sistemas anteriores tienen ventajas e inconvenientes y los sistemas híbridos, que resultan de la mezcla de dos o más de los anteriores, buscan mantener los beneficios de cada uno de ellos minimizando sus inconvenientes. El principal problema que tienen los sistemas híbridos es la cantidad de operaciones que deben realizar y, por tanto, las restricciones de rendimiento que sufren. El algoritmo de recomendación de Amazon implementa un algoritmo híbrido entre estudios de contenido y algoritmos colaborativos.

2. 2. 1. SISTEMAS COLABORATIVOS

Los sistemas colaborativos, por su naturaleza, tienen la capacidad de generar recomendaciones basándose únicamente en datos relativos al comportamiento previo de los usuarios, a diferencia de los sistemas basados en datos demográficos. Además, por tratarse de datos totalmente relacionados con los productos (puntuaciones, comentarios, compras, etc.) el conjunto de datos suele estar mucho más poblado que cuando se trabaja con datos más sensibles como los demográficos. Este hecho, junto a la mayor complejidad computacional de los algoritmos de agrupamiento con respecto a los sistemas basados en contenido, generan problemas de **escalabilidad** en los sistemas colaborativos.

Otro de los problemas a los que se enfrentan los sistemas colaborativos es conocido como ***arranque en frío***. Este término hace referencia a la entrada de un nuevo usuario o de un nuevo producto al sistema ya que, cuando esto ocurre, el sistema de recomendación no posee ningún tipo de información sobre el recién llegado y, por tanto, el algoritmo colaborativo no puede situarlo en ninguno de los grupos. Los sistemas basados en contenido sufren este mismo problema, lo que no sucede en los sistemas demográficos ya que estos pueden solicitar una serie de datos obligatorios al usuario cuando se registra y con ellos alimentar el algoritmo de recomendación. Algunas de las soluciones más habituales para el problema de arranque en frío consisten en generar un perfil del nuevo usuario a partir de información disponible de su comportamiento fuera del sistema (búsquedas en la web, redes sociales, páginas visitadas, etc.) o, mientras el perfil del usuario se completa, basar las recomendaciones en los gustos del resto de usuarios del sistema. Un ejemplo de esto consistiría en recomendar las películas mejor puntuadas de

todo el sistema o sugerir la visualización de los últimos ganadores del Oscar a la mejor película.

Entre los puntos fuertes de los sistemas colaborativos, cabe destacar la capacidad de mejora de su precisión a medida que aumenta la cantidad de información sobre el usuario. Esto es debido a que cuanta más información recibe el algoritmo de agrupamiento, mayor es la probabilidad de encontrar usuarios parecidos y, por tanto, de que los resultados de la recomendación mejoren. Además, el uso de los datos de otros usuarios evita el problema de *sobre-especialización* que sufren los sistemas basados en contenido.

Como se ha destacado en el apartado anterior, la clave del buen funcionamiento de un sistema colaborativo reside en la precisión del algoritmo de agrupamiento utilizado. Un buen algoritmo de clustering debe ser capaz de descubrir los datos más relevantes a la hora de buscar relaciones y adaptarse a los cambios en los valores de los mismos. Además, el algoritmo debe ser capaz de identificar aquellos atributos que no tienen tanta importancia en las relaciones, para permanecer estable cuando se produzcan cambios en sus valores.

2. 2. 2. CLASIFICACIÓN DE LOS ALGORITMOS DE CLUSTERING

Existen numerosas clasificaciones de algoritmos de clustering atendiendo a sus características. A continuación, se presentan las clasificaciones correspondientes a los criterios más utilizados.

Según el grado de conocimiento previo sobre los grupos que deben formarse se distinguen dos tipos de algoritmos:

- **Algoritmos supervisados:** son aquellos en los que se conoce de antemano cuántos grupos deben formarse y qué características debe tener cada uno. Su objetivo es, por lo tanto, conseguir asignar a los usuarios al grupo al que pertenecen, motivo por el que también se conocen como algoritmos de **clasificación**. Un ejemplo tradicional de este tipo de algoritmos es la clasificación de elementos naturales (flores, pájaros, etc.) en la especie a la que pertenecen basándose en sus características (número de pétalos, tamaño, envergadura, etc.).
- **Algoritmos no supervisados:** al contrarios que los anteriores, los algoritmos no supervisados parten del total desconocimiento del número de clusters que deben ser generados o de las características que estos deben tener. La mayoría de los procesos de recomendación colaborativa utilizan algoritmos no supervisados.

A su vez, los algoritmos no supervisados se subdividen en otros dos tipos dependiendo de si el proceso de agrupamiento es o no secuencial, distinguiéndose:

- **Algoritmos jerárquicos**: son aquellos en los que se genera un árbol de jerarquías durante su ejecución. Es decir, no solo generan una solución final sino que van construyendo de forma incremental soluciones intermedias que permiten al responsable del análisis seleccionar qué solución se adapta mejor a sus requisitos de precisión, número de clusters o número de elementos en cada cluster. Este mayor grado de información suele suponer un aumento en el tiempo de procesamiento y en el gasto de recursos.

- **Algoritmos no jerárquicos:** al contrario que los jerárquicos, en este caso no se genera ningún resultado intermedio sino que se obtiene un único resultado final. Este hecho hace que estos algoritmos sean mucho menos pesados a nivel computacional pero exigen que, al ejecutarlos, se indique el número de clusters que se quiere obtener y la determinación de este número es, en la actualidad, un problema sin solución óptima. Entre los algoritmos no jerárquicos destaca el **algoritmo de *K-means*** en el que en cada iteración, y partiendo de una selección aleatoria de *K* puntos que en la primera iteración juegan el papel de clusters, cada punto es asignado al cluster cuyos puntos estén a una distancia media menor. Este algoritmo suele tener como condición de parada el momento en el que todos los puntos permanecen en el mismo cluster después de una iteración. Además, y para contrarrestar los efectos de la selección aleatoria de puntos iniciales, suelen llevar a cabo el proceso de agrupamiento varias veces para posteriormente mezclar los resultados de cado uno de ellos y obtener un agrupamiento final.

Finalmente, entre los algoritmos jerárquicos se distinguen también dos grandes grupos dependiendo del orden en el que realicen los cálculos:

- **Algoritmos aglomerativos o *bottom-up*:** son aquellos en los que de partida cada usuario forma su propio cluster y en cada iteración se procede a unir aquellos clusters que tienen más en común. De esta forma, se obtiene una estructura similar a un árbol invertido que permite al responsable del sistema de recomendación seleccionar en qué fase del algoritmo se obtiene el resultado óptimo.
- **Algoritmos de división o *top-down*:** son aquellos en los que, por el contrario, se parte de un único cluster que contiene a todos los usuarios y se van haciendo divisiones hasta obtener la división total del conjunto. Al igual que en el caso anterior queda en manos del responsable del sistema la selección de la fase del proceso en la que se obtiene el resultado óptimo.

En ambos casos los resultados obtenidos suelen representarse en estructuras conocidas como **dendrogramas** que permiten ver la evolución del algoritmo facilitando el proceso de selección de la fase óptima. Por ejemplo, la Figura 1 muestra el dendrograma obtenido al ejecutar un algoritmo de división sobre 30 usuarios.

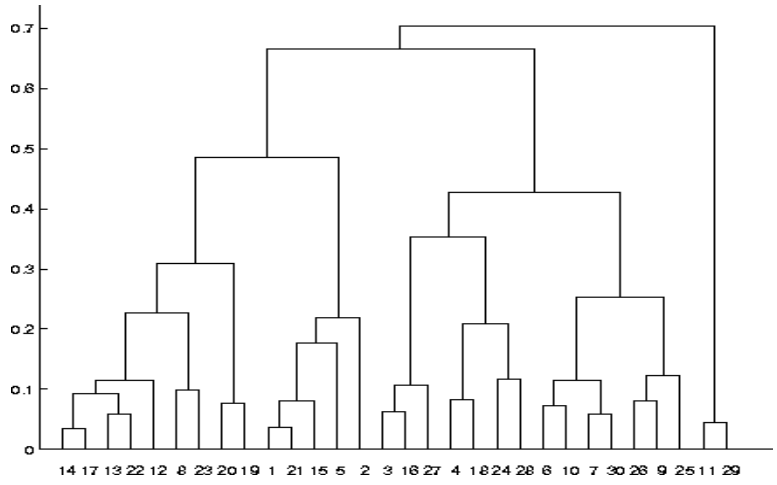

Figura 1.- Dendrograma obtenido al ejecutar un algoritmo de división.

Además del tipo de algoritmo a implementar, el otro factor con mayor importancia en el funcionamiento de un sistema de recomendación es la selección del criterio a utilizar para determinar cuándo se deben agrupar o separar usuarios. En los algoritmos jerárquicos estos criterios se determinan mediante la selección de dos funciones que miden la distancia entre usuarios y entre clusters. Las funciones que definen las distancias entre usuarios se conocen como **métricas** y son funciones que tienen las propiedades de distancia matemática, es decir, son definidas positivas, simétricas y verifican la desigualdad triangular. Hay que tener en cuenta que estas funciones se aplican sobre un universo con tantas dimensiones como atributos de los usuarios se quieran utilizar. Por lo tanto, cada usuario queda caracterizado con un punto cuyas coordenadas corresponden con los valores que tenga asignados para cada uno de estos atributos. A continuación se presentan las métricas más populares siendo *n* el número de atributos utilizados:

- **Distancia Euclídea:** $d(u_1, u_2) = \sqrt{\sum_{i=1}^{n}(u_{1_i} - u_{2_i})^2}$
- **Distancia Euclídea al cuadrado:** $d(u_1, u_2) = \sum_{i=1}^{n}\left(u_{1_i} - u_{2_i}\right)^2$
- **Distancia Manhattan:** $d(u_1, u_2) = \sum_{i=1}^{n} |u_{1_i} - u_{2_i}|$
- **Distancia de Hamming** o de **Levenshtein**, utilizadas en el caso de atributos textuales.

El otro elemento que es necesario definir es la función de distancia entre clusters. Estas funciones se construyen a partir de las métricas y determinan en cada fase del algoritmo qué cluster se debe crear, ya sea mediante división o mediante unión. Los más utilizados son:

- **Enlace completo:** $d(A, B) = max\{d(u_1, u_2) | u_1 \in A, u_2 \in B\}$
- **Enlace simple:** $d(A, B) = min\{d(u_1, u_2) | u_1 \in A, u_2 \in B\}$
- **Enlace promedio:** $d(A, B) = \frac{1}{|A||B|} \sum_{u_1 \in A} \sum_{u_2 \in B} d(u_1, u_2)$
- **Método de Ward:** este método, presentado por J.H. Ward en 1963, mide la distancia entre dos clusters a partir de las varianzas de las distancias entre puntos del mismo cluster. De este modo, dos clusters estarán más próximos cuando las varianzas de las distancias entre sus puntos internos tengan valores similares. Matemáticamente hablando la distancia queda definida de la siguiente forma:

$$d(A, B) = \frac{|A||B|}{|A| + |B|} \|\mu_A - \mu_B\|^2$$

siendo μ_I la media de las distancias entre los puntos del cluster I.

Cabe destacar que, al igual que sucedía con el número de clusters a generar, el uso de un método u otro depende de la estructura de los datos manejados. El método de enlace simple suele comportarse mejor en la detección de clusters de forma alargada, mientras que el método de enlace completo suele generar clusters más compactos. El problema de estos dos métodos es que trabajan con valores extremos, es decir, solo dependen de la distancia máxima o mínima, independientemente del resto de relaciones. Esto suele traducirse en una mayor sensibilidad ante valores fuera de lo normal, conocidos como

outliers, característica poco deseable en estos métodos. Diversos estudios sobre la eficiencia de estos métodos como, por ejemplo, (Kuiper & Fisher, 1975), (Blashfield, 1976) y (Hands & Everitt, 1987) han determinado que el método de Ward es el que tiene un mejor rendimiento en la mayoría de los casos.

2. 3. CAMRA2010 FILMTIPSET SOCIAL

El conjunto de datos *CAMRa2010 Filmtipset Social* tiene su origen en el concurso de sistemas de recomendación de películas (*Challenge on Context-Aware Movie Recommendation, CAMRa*) celebrado en el marco de la cuarta edición del Congreso sobre sistemas de recomendación organizado por la ACM (*Association for Computing Machinery*) durante el mes de septiembre de 2010 en Barcelona. En este concurso se propusieron tres tareas de recomendación usando cuatro conjuntos de datos de distinta naturaleza. Todos ellos estaban relacionados con un servicio de visionado de películas, y uno de ellos, el que contenía la información sobre las puntuaciones asignadas a las películas, ha sido utilizado como material en este trabajo, por lo que a continuación se realiza una descripción detallada del mismo. Queda como trabajo futuro el estudio de la utilización de los tres conjuntos restantes, al objeto de conseguir mejoras en el rendimiento.

El dataset utilizado tiene dos entidades principales: los **usuarios** del servicio y las **películas** ofertadas. Además, estas entidades se relacionan mediante las puntuaciones (**ratings**) que asignan los usuarios a las películas. Estas puntuaciones toman valores discretos entre uno y cinco, siendo uno una valoración muy negativa y cinco una valoración excelente. Además de esta relación principal, existen otras relaciones entre los usuarios y las películas en el conjunto de datos: las colecciones, los comentarios, los favoritos y las críticas. La relación de **colección** modeliza la posesión de una película en estado físico (VHS, DVD, BlueRay, etc.), mientras que los **comentarios** se corresponden con las opiniones en forma de texto dejadas por los usuarios en relación a una película determinada. Por otro lado, los usuarios tienen la opción de, además de darle una puntuación numérica a una película, asignarle otra opinión más general basada únicamente en dos valores: **me gusta** (like) o **no me gusta** (dislike). Finalmente, los usuarios pueden publicar **críticas** más extensas de las películas que pueden ser puntuadas por el resto de usuarios como **muy útiles** (thumbs up), **acertadas** (average) o **poco útiles** (thumbs down).

Adicionalmente, el conjunto de datos contiene información sobre las relaciones de **amistad** entre usuarios. Este dato es muy interesante ya que proporciona información sobre los usuarios más allá del contexto del servicio y podría ser utilizado para implementar algoritmos colaborativos basados en la amistad.

En lo que se refiere a las películas, el conjunto de datos también contiene información sobre sus metadatos. Además de contar con una lista de **profesionales del mundo del cine**, la base de datos relaciona cada profesional con las películas en las que ha participado indicando el rol que tuvo en cada una (**actor**, **director** o **guionista**). Además, el dataset contiene una tipología de los **géneros** de las películas y asigna uno o más de estos a cada una. En la misma línea de la relación de amistad entre usuarios, las películas se relacionan entre sí mediante una relación de **similitud**. Finalmente, aparece una entidad más llamada

lista que agrupa películas. En la Figura 2 se muestra el diagrama Entidad-Relación correspondiente a este conjunto de datos.

Cabe destacar que toda la información incluida en el conjunto de datos está anonimizada por lo que el sistema de recomendación debe basarse únicamente en la información contenida en él, ya que resulta imposible establecer relaciones entre los elementos del dataset y películas o usuarios reales.

Para finalizar, también es importante tener en consideración que los conjuntos de ratings de entrenamiento y de test se proporcionaban ya divididos por la organización del concurso, con la particularidad de que todas las entradas del conjunto de ratings de test corresponden a valoraciones positivas (entre tres y cinco puntos). Este hecho sugiere que se llevó a cabo un prefiltrado con el objetivo de medir la capacidad de los sistemas propuestos por los concursantes para sugerir el visionado de películas sin tener en cuenta las recomendaciones negativas. Poniendo la situación en contexto esta decisión cobra sentido porque no se conoce ningún servicio de visionado de películas que además de fomentar el visionado de algunas incluya sistemas de disuasión para otras.

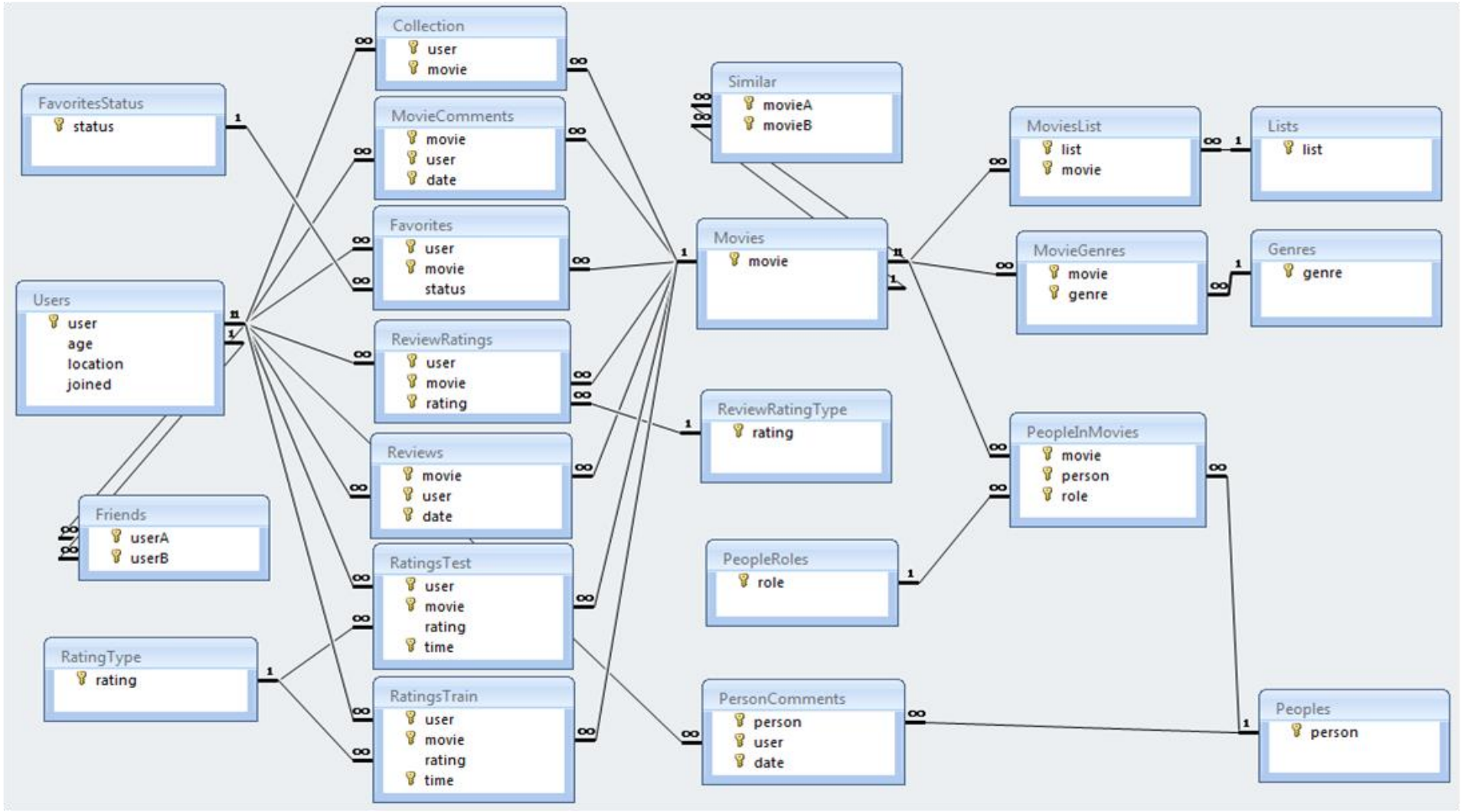

Figura 2.- Diagrama Entidad-Relación proporcionado por la ACM a los concursantes de CAMRa 2010.

3. SISTEMA DE RECOMENDACIÓN COLABORATIVO BASADO EN TEORÍA DE JUEGOS

Para el diseño de un modelo de sistema de recomendación basado en teoría de juegos se han seguido los siguientes pasos:

1. Análisis exhaustivo de la información disponible en el conjunto de datos usado para la parte experimental.
2. Planteamiento del problema global a resolver y propuesta de una solución teórica.
3. Implementación de la solución propuesta para solucionar un problema práctico: la recomendación de películas.

3. 1. ANÁLISIS DEL CONJUNTO DE DATOS

El primer paso a la hora de construir cualquier sistema de recomendación es conocer en profundidad el conjunto de datos que se va a utilizar. No solo es importante saber la estructura del conjunto de datos (entidades, relaciones y atributos) sino también analizar la naturaleza de los mismos. De esta forma resulta más fácil la identificación de los atributos que deben ser utilizados para agrupar los usuarios así como la importancia de cada uno de ellos. A continuación se presentan los análisis realizados sobre los datos contenidos en el dataset *CAMRa2010 Filmtipset Social*.

3. 1. 1. TAMAÑO DEL CONJUNTO DE DATOS

La primera característica analizada es el número de entradas que posee cada tabla. Este dato es importante para evitar futuros problemas a nivel computacional. De hecho, y dado que se está midiendo la eficacia de un nuevo método, es preferible trabajar con un conjunto de datos reducido de forma que se minimice el tiempo necesario para la estimación del rendimiento y, sólo en el caso de que los resultados sean positivos, extender el método a conjuntos de datos más grandes. La Tabla 1 muestra el número de elementos que contienen las tablas correspondientes a las entidades principales.

Tabla 1.- Número de entradas en las tablas del conjunto de datos original.

Usuarios	**Películas**	**Ratings entrenamiento**	**Ratings test**	**Amistades**
16.473	24.222	3.075.346	15.729	12.171
Comentarios	**Criticas**	**Favoritos**	**Colecciones**	**Profesionales cine**
146.510	1.044	15.283	102	88.849

Como puede observarse, la cantidad de datos disponibles en el dataset es demasiado grande para el objetivo perseguido y, por este motivo, se ha decidido realizar un filtrado previo. Con el fin de mantener en el sistema a los usuarios que aportan mayor cantidad de información, se impone la restricción de que los usuarios que deben permanecer en el conjunto de datos sean aquellos que al menos han puntuado **quinientas** películas. Para

determinar este valor se realizaron distintas simulaciones eligiéndose el valor más alto que permitía mantener un conjunto de datos que no generara problemas a nivel computacional. Queda como línea de trabajo futura comprobar cómo se comporta el sistema generado al trabajar con el conjunto de datos completo. Una vez identificados estos usuarios, las películas que permanecerán en el sistema serán aquellas que hayan sido puntuadas por al menos uno de los usuarios que permanecen activos. La Tabla 2 muestra el número de elementos que tiene el conjunto de datos resultante y que serán utilizados en el estudio.

Tabla 2.- Número de entradas en las tablas del conjunto de datos final y porcentaje que representan respecto al conjunto de datos original.

Usuarios	**Películas**	**Ratings entrenamiento**	**Ratings test**	**Amistades**
1.623 (9,85%)	21.603 (89,19%)	1.304.116 (42,41%)	6.452 (41,02%)	710 (5,83%)
Comentarios	**Criticas**	**Favoritos**	**Colecciones**	**Profesionales cine**
59.226 (40,42%)	363 (34,77%)	4.113 (26,91%)	78 (76,47%)	79459 (89,43%)

Puede observarse que al seleccionar aquellos usuarios con mayor número de puntuaciones, la cantidad de películas se ha reducido en una proporción mucho menor que la de usuarios. De esta manera, el conjunto de datos obtenido tiene mucha información sobre cada uno de los usuarios lo que permitirá que el algoritmo de agrupación funcione correctamente. Además, tanto el conjunto de ratings de entrenamiento como el conjunto de ratings de test disminuyen su tamaño de manera similar manteniéndose la proporción entre ellos.

3. 1. 2. MATRIZ DE PUNTUACIONES

Dado que las puntuaciones asignadas por los usuarios a las películas son las relaciones principales del sistema y además van a ser las que determinen el resultado del modelo propuesto, es necesario conocer cómo se distribuyen y cómo se ven afectadas por el resto de entidades del conjunto. En la Figura 3 se observa la distribución de las puntuaciones en el conjunto de entrenamiento.

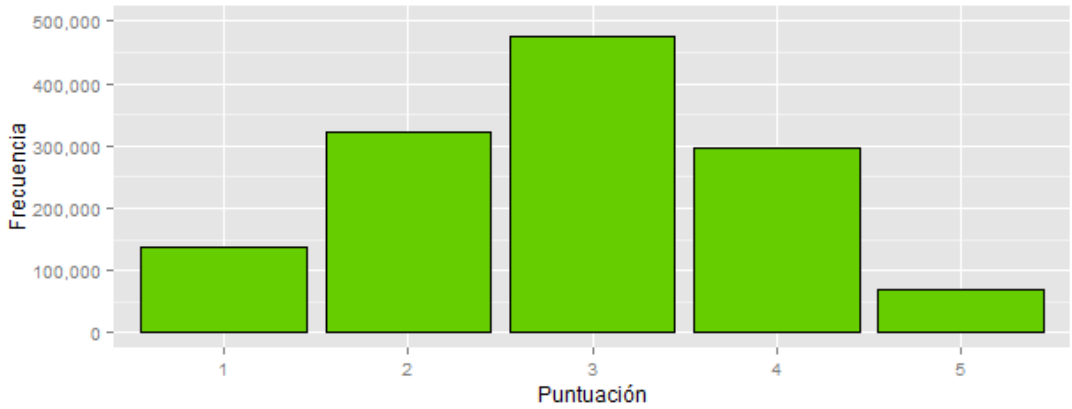

Figura 3.- Distribución de los ratings del conjunto de entrenamiento.

En la gráfica se observa que la puntuación más repetida es el tres y que el número de puntuaciones negativas (unos y doses) es mayor que el número de puntuaciones positivas (cuatros y cincos), lo que aumenta la dificultad de obtener buenos resultados con el conjunto de test disponible debido al filtrado de las puntuaciones negativas del mismo.

Otro de los aspectos importantes es comprobar cuántas películas han puntuado los usuarios, una vez impuesta la restricción de que al menos hayan asignado quinientas puntuaciones. La Figura 4 muestra la frecuencia de películas puntuadas en intervalos. Como cabía esperar por las simulaciones llevadas a cabo para determinar la cantidad de películas mínima, el intervalo de películas puntuadas que engloba a un mayor número de usuarios es aquel comprendido entre quinientas y setecientas cincuenta puntuaciones. Sin embargo, la cantidad de usuarios que han realizado entre setecientas cincuenta y mil quinientas películas no es nada desdeñable. De hecho, al calcular algunas medidas estadísticas descriptivas se obtiene una media de **803.5** puntuaciones y la mediana se sitúa en **696** puntuaciones.

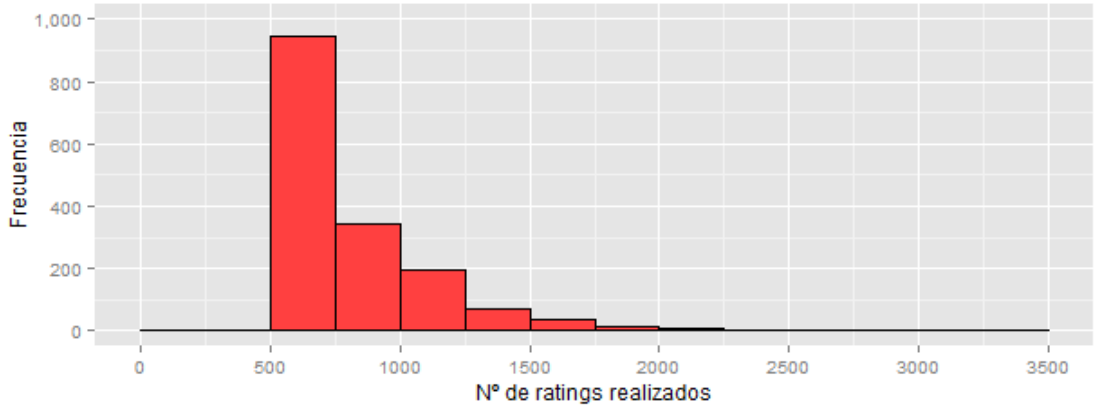

Figura 4.- Distribución en intervalos del número de puntuaciones asignadas por cada usuario.

Así mismo, es importante conocer la distribución del número de puntuaciones recibidas por cada película. La Figura 5 muestra el histograma con intervalos de cien puntuaciones. En este caso la media se sitúa en **60,36** puntuaciones y la mediana en **4**, valores de los que puede concluirse que existe un gran número de películas que han recibido únicamente una valoración.

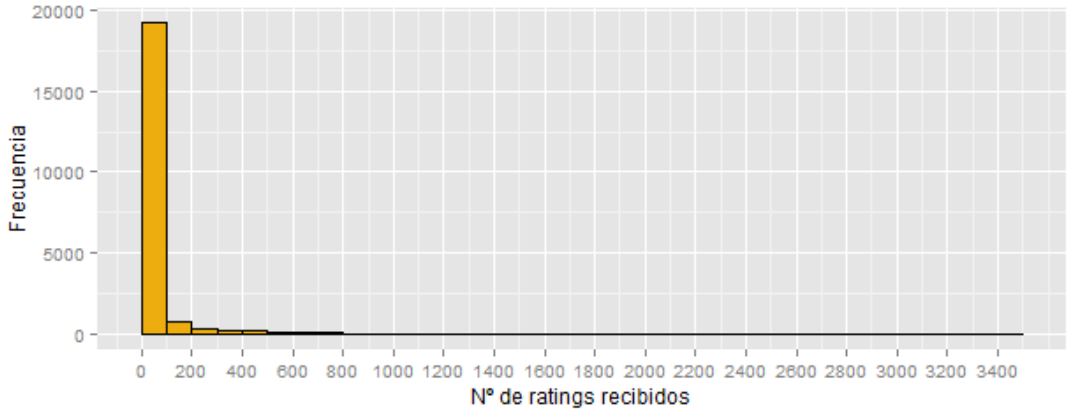

Figura 5.- Distribución en intervalos del número de puntuaciones recibidas por cada película.

Cuando para comprobar este hecho se desglosaron los datos del primer intervalo (Figura 6) pudo observarse que, efectivamente, casi doce mil de las diecinueve mil películas habían recibido menos de cinco puntuaciones.

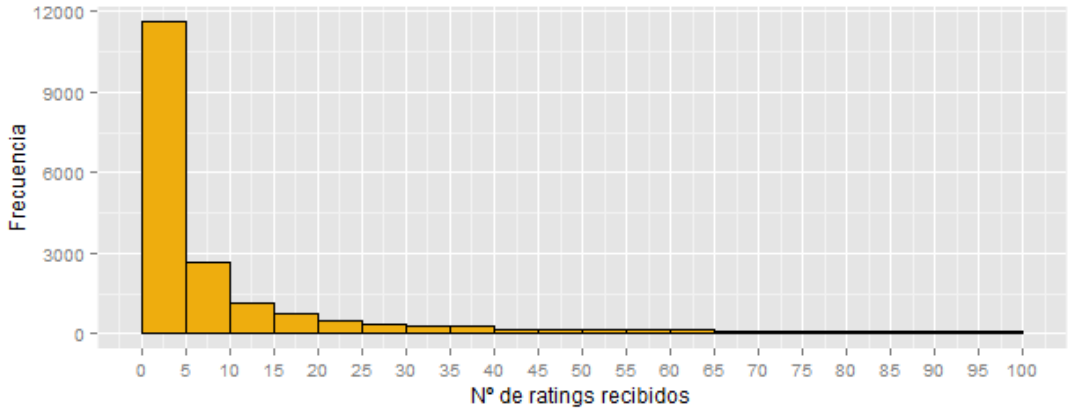

Figura 6.- Distribución en intervalos del número de puntuaciones recibidas por cada película: zoom a intervalos menores de 100 puntuaciones.

Estos resultados sugieren que los sistemas basados en contenido generarían recomendaciones de este tipo de películas en contadas ocasiones y, por tanto, el éxito o fracaso del sistema de recomendación colaborativo que se va a construir puede depender de las recomendaciones que se generen para este grupo de películas. Si el sistema es capaz de recomendar su visionado correctamente será una ventaja con respecto a los sistemas basados en contenido pero si no consigue generar recomendaciones o éstas no son acertadas el rendimiento del sistema colaborativo se verá doblemente penalizado ya que utilizará más recursos y además no mejorará los resultados que se obtienen con las herramientas basadas en contenido.

3. 1. 3. INFORMACIÓN ADICIONAL: FAVORITOS, CRITICAS Y COMENTARIOS

Una vez analizados los datos correspondientes a las puntuaciones, consideramos importante analizar la naturaleza de la información adicional que contiene el conjunto de datos ya que algunas entidades como los comentarios, críticas y favoritos, aunque no aportan información tan directa como las puntuaciones, pueden servir de ayuda para comprender el comportamiento de los usuarios y, por tanto, mejorar el algoritmo de agrupamiento.

Por ejemplo, es interesante comprobar si la proporción entre puntuaciones positivas y negativas descrita anteriormente se ve reflejada en la selección de favoritos. Para ello, hemos analizado en qué puntuaciones se traducen los dos posibles valores de la tabla de favoritos. La Figura 7 muestra las puntuaciones que los usuarios han asignado a las películas incluidas dentro de sus favoritos en función de si han sido catalogadas como "*me gusta*" (A) o "*no me gusta*" (B). Como cabía esperar, los valores de favoritismo se corresponden mayoritariamente con los extremos del rango de puntuaciones: cinco y uno, respectivamente.

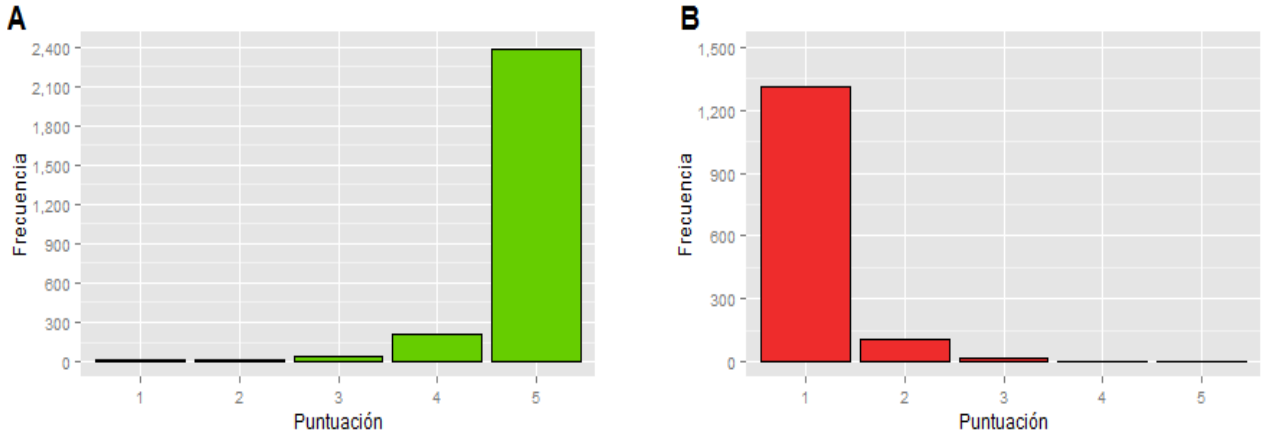

Figura 7.- Distribución de las puntuaciones dadas por los usuarios a películas que han sido seleccionadas como favoritas. (A) Películas marcadas con "*me gusta*" y (B) Películas marcadas con *"no me gusta".*

Confirmado este hecho, se ha establecido una comparación entre el porcentaje de puntuaciones positivas y negativas (excluyendo las puntuaciones con valor tres) y el porcentaje de películas valoradas como "*me gusta*" o *"no me gusta".* En la Figura 8 se muestran los resultados de esta comparación a partir de los que se puede concluir que los usuarios utilizan la herramienta de selección de favoritos para marcar como *"me gusta"* aquellas películas que puntúan positivamente con mucha mayor frecuencia que para marcar como *"no me gusta"* aquellas que no son de su agrado.

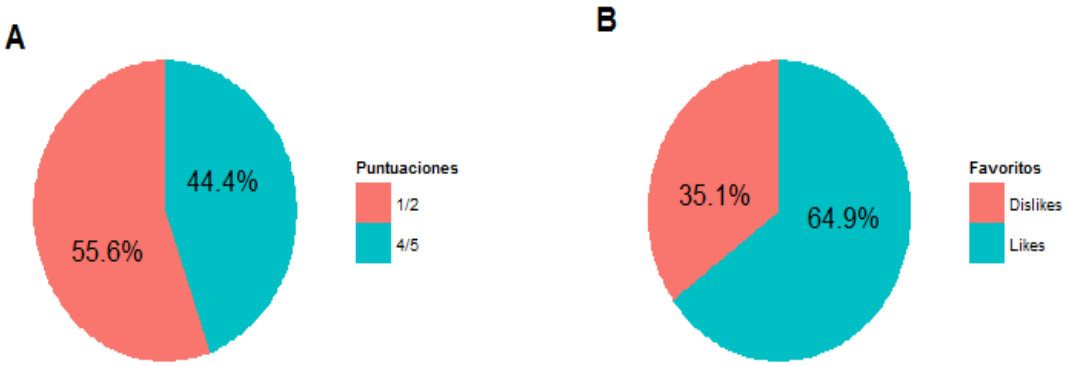

Figura 8.- Comparativa entre el porcentaje de puntuaciones positivas y negativas (sin tener en cuenta las puntuaciones de valor 3) (A) y el porcentaje de favoritos identificados como *"me gusta"* y *"no me gusta"* (B).

En este punto también se ha considerado interesante conocer la relación entre las críticas de las películas y su puntuación. Los resultados mostrados en la Figura 9 indican que el número de películas criticadas a las que se les asignó una buena puntación es mayor que el número de aquellas que, además de haber sido criticadas, recibieron una mala puntuación. Por lo tanto, puede concluirse que los usuarios prefieren invertir su tiempo en ensalzar aquellas películas que les han gustado antes que escribir malas críticas de las que no. Este hecho refuerza la idea de restringir los casos de test a puntuaciones iguales o mayores que tres ya que los usuarios prefieren recibir recomendaciones sobre lo que deben ver en lugar de sobre lo que no.

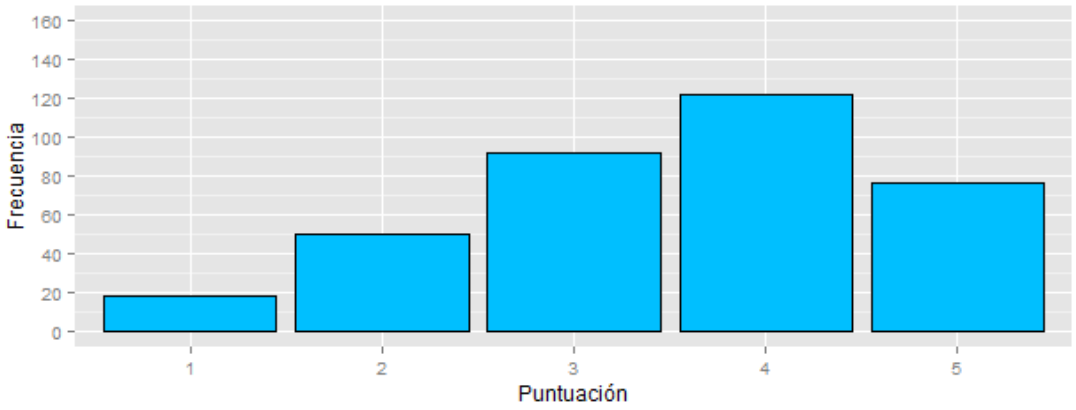

Figura 9.- Distribución de las puntuaciones asignadas por los usuarios a las películas que han criticado.

Para terminar con el análisis de la información adicional, se ha construido la gráfica correspondiente a la distribución de las puntuaciones asignadas a las películas comentadas. En este caso, al tratarse de una acción más rápida que la publicación de una crítica cabía esperar que se replicase la distribución total obtenida. La Figura 10 muestra la gráfica correspondiente y, aunque en este caso el número de cuatros es superior al número de doses, la distribución es mucho más parecida que la obtenida al analizar las críticas.

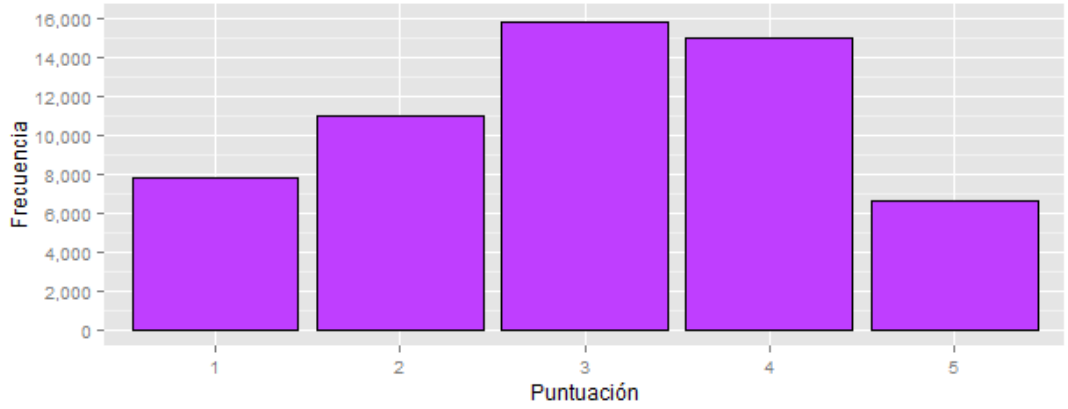

Figura 10.- Distribución de las puntuaciones asignadas por los usuarios a las películas que han comentado.

3. 1. 4. INFORMACIÓN SOCIAL: RELACIONES DE AMISTAD

Aunque las relaciones de amistad entre usuarios no son datos centrales en el estudio llevado a cabo, sí se ha considerado interesante analizar cómo se relaciona con las puntuaciones dadas ya que aunque parece lógico pensar que la amistad debe influir positivamente a la hora de formar parte de un mismo grupo, también cabe la duda de que el hecho de ser amigos implique necesariamente tener gustos similares en términos cinematográficos.

Para comprobar este aspecto se ha calculado el valor del coeficiente de correlación de Pearson entre las puntuaciones de cada par de amigos. Este coeficiente es una medida de

la relación lineal entre dos variables cuantitativas cuyo valor se encuentra en el intervalo [-1,1] y se calcula aplicando la siguiente fórmula:

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y},$$

siendo:

- σ_{XY} la covarianza de (X,Y),
- σ_X la desviación típica de la variable X
- σ_Y la desviación típica de la variable Y
- μ_X la media de la variable X
- μ_Y la media de la variable Y.

Los valores de este coeficiente se interpretan de la siguiente manera:

- Si es igual a 1, existe una correlación positiva perfecta, es decir, cuando el valor de una de las variables aumenta la otra también lo hace y además con una proporción constante.
- Si toma un valor entre 0 y 1, existe una correlación positiva pero el crecimiento no sigue ninguna regla de proporcionalidad.
- Si es igual a 0, no existe una relación lineal entre ambas variables.
- Si el valor se encuentra en el intervalo (-1,0) existe una relación de correlación negativa, es decir, cuando el valor de una de las variables aumenta la otra disminuye.
- Simétricamente, si el coeficiente toma el valor -1 se establece una correlación negativa perfecta.

En el caso que nos ocupa, las variables *X* e *Y* son vectores que contienen las puntuaciones asignadas por dos usuarios que son amigos a las películas que ambos han comentado. La Figura 11 muestra el histograma de frecuencias de los coeficientes calculados con intervalos de tamaño 0.05. Se observa que la mayoría de valores se encuentran en el intervalo (0,1) con una mayor densidad en torno al valor 0.55, lo cual nos lleva a pensar que la relación de amistad influye positivamente a la hora de agrupar usuarios aunque no es determinante. Además, al calcular los valores estadísticos principales del conjunto se ha obtenido un coeficiente medio de **0.5157** siendo el máximo **0.9616** y el mínimo **-0.1383**.

De todos estos valores, podemos concluir que la relación de amistad podría ser utilizada como un factor a la hora de agrupar usuarios ya que aquellos que son amigos suelen comportarse de forma similar a la hora de puntuar las películas que ambos han visto. Sin embargo, un análisis más profundo debería incidir en el número de películas que tienen en común los usuarios que son amigos y cómo afecta esto al coeficiente. Dicho análisis excede las posibilidades de este trabajo por lo que se planteará como trabajo futuro.

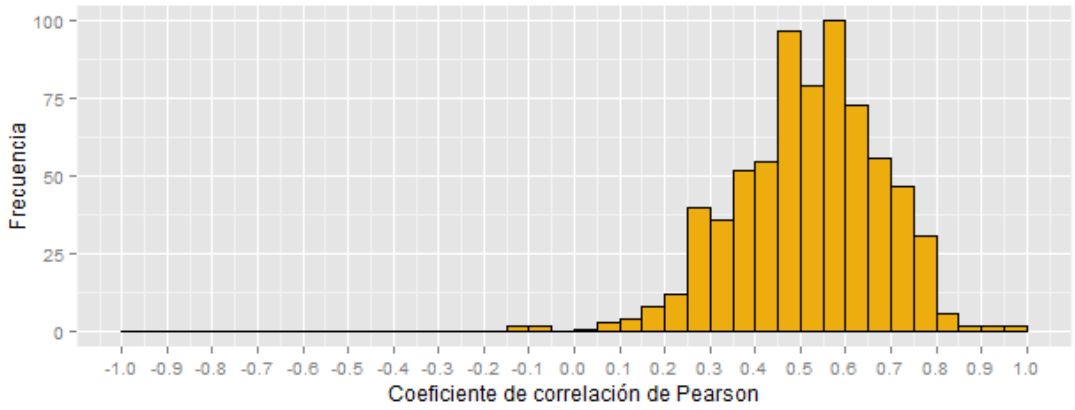

Figura 11.- Distribución de los coeficientes de correlación de Pearson entre vectores de puntuaciones de películas realizadas por usuarios con una relación de amistad.

3. 2. DISEÑO TEÓRICO DEL SISTEMA DE RECOMENDACIÓN

Como se indicaba al inicio de la sección 3 de esta memoria el segundo paso en el diseño de un modelo de sistema de recomendación basado en teoría de juegos es el planteamiento teórico del problema que se quiere resolver. Para ello a continuación se incluye información teórica detallada del diseño del sistema que nos proponemos desarrollar.

3. 2. 1. DEFINICIÓN DEL JUEGO COOPERATIVO

En el trabajo que presentamos se plantea un escenario novedoso: el modelado en términos de juegos cooperativos del proceso de agrupamiento de usuarios en un sistema de recomendación. Para ello, se deben identificar los jugadores que toman parte en él y, de alguna manera, describir la función característica a utilizar. Además, es importante determinar el funcionamiento del algoritmo de clustering propuesto de cara a buscar relaciones con los algoritmos tradicionales, de forma que puedan servir de ayuda a la hora de desarrollar nuestra solución.

El conjunto de jugadores participantes en el problema de recomendación se corresponde totalmente con el conjunto de usuarios al que queremos recomendar productos. Por otro lado, la función característica debe asignar un valor a cada posible coalición en términos de cuanto de valiosa sería para el sistema de recomendación. En este punto, se introduce un pequeño matiz con respecto a la teoría de juegos clásica en relación con la función característica. Se define el concepto de **beneficio de un jugador** como la mejora del sistema de recomendación en términos de alguna métrica estándar. Es decir, un usuario obtendrá mayor beneficio si forma parte de una coalición de usuarios lo más parecidos a él, de forma que la recomendación obtenida sea lo más personalizada posible. Este hecho se traduce en que la función característica no tiene por qué tener la propiedad de la superaditividad ya que la unión de un nuevo jugador a una coalición formada por usuarios que no tienen nada que ver con él, se traducirá en un descenso del valor de la función característica. Por lo tanto, tendremos que tener cuidado con las herramientas de la teoría de juegos que utilicemos ya que la mayoría de resultados parten de la hipótesis de manejar juegos superaditivos.

Teniendo en cuenta todo lo anterior, el problema de recomendación puede definirse en términos de teoría de juegos de la siguiente manera:

- Sean $u_1, u_2, \dots, u_n$ los usuarios del sistema que necesitan recomendaciones. Sea $\gamma(u_i)$ la estructura de datos relativos a u_i que va a ser utilizada en el proceso de recomendación. En el ejemplo práctico tratado en este documento, $\gamma(u_i)$ integraría las puntuaciones asignadas, los comentarios y las críticas, las películas marcadas como "*me gusta*" y "*no me gusta*", los usuarios que tienen una relación de amistad y cualquier otro dato que pueda ser candidato a ser utilizado a la hora de medir la similitud entre usuarios.
- El juego se define como el par (N, v) con $N = \{u_1, u_2, \dots, u_n\}$ y v: $2^n \longrightarrow \mathbb{R}$, función definida para cada combinación de los elementos de *N*, $A = \{u'_1, u'_2, \dots, u'_m\}$, de la siguiente manera: $v(A) = f(\gamma(u'_1), \gamma(u'_2), \dots, \gamma(u'_m))$. La función característica *f* recibirá como parámetros un vector de estructuras de datos de tamaño igual al número de jugadores involucrados en la coalición *A* y devolverá un número real que será el valor de la función característica asignado a la coalición.

Es difícil dar una definición universal precisa de la función *f* ya que depende totalmente de la estructura de datos $\gamma(u_i)$ elegida para representar a cada usuario. Sin embargo, centrándonos en el problema práctico que queremos resolver, y restringiendo la información de cada usuario a las puntuaciones que ha asignado, el dominio de definición de la función *f* será $\mathbb{R}^{n\,x\,m}$, donde *m* es el número total de películas que han sido puntuadas por algún usuario y que, por tanto, pueden ser recomendadas. La función *f* recibirá una matriz de estas dimensiones, la manipulará y obtendrá el valor de la función característica.

De esta manera, el juego cooperativo queda definido pero, de cara a su implementación, faltaría decidir qué información debe utilizarse para modelizar a los usuarios y cómo manipularla para generar un valor real que mida el beneficio de cada coalición. Sin embargo, esta cuestión, al depender de las características particulares de cada sistema, se estudiará más adelante cuando se profundice en el problema concreto de la recomendación de películas.

3. 2. 2. FORMACIÓN DE COALICIONES

La siguiente cuestión a resolver es la formación de las coaliciones ya que, como se comentó en el capítulo de introducción, en la teoría de juegos la práctica habitual ha sido dar por supuesto que todos los jugadores acabarían cooperando y se ha puesto mayor atención en el problema de reparto. Sin embargo, al construir un sistema de recomendación la cuestión fundamental es determinar qué grupos deben formarse. La solución propuesta que proponemos es una adaptación de la idea de Shapley de que la coalición total se formaría **incrementalmente,** a la que se ha añadido el concepto de **beneficio mínimo**. Para poder describir la solución correctamente se ha introducido el término **participante** para referirnos a cualquier jugador o coalición que participa en alguna fase del proceso de agrupación. De esta forma, la solución propuesta tiene muchos elementos en común con un algoritmo de clustering aglomerativo, ya que se basa en la repetición de un proceso que, partiendo de un conjunto de participantes, selecciona la unión de aquellos dos que mayor beneficio genera. Una vez hecho esto, se repite el proceso eliminando los dos

participantes seleccionados y en su lugar se añade uno nuevo que representa a la coalición.

El concepto de beneficio mínimo hace referencia a la condición de parada de este algoritmo. Si se repitiese el proceso sin ningún tipo de restricción se obtendría la coalición total pero, dado que la función característica no tiene por qué ser superaditiva, se forman coaliciones con un beneficio muy bajo o incluso negativo. Por lo tanto, el beneficio mínimo es un parámetro del sistema que determina cuánto beneficio en términos de similitud entre sus componentes, debe generar como mínimo una coalición para ser aceptada. De esta forma, si en alguna iteración la coalición con mayor rendimiento no supera este valor, el proceso se da por terminado y las coaliciones formadas hasta ese momento son consideradas el resultado final del sistema de recomendación.

Uno de los mayores problemas que tiene la construcción incremental de las coaliciones es el coste computacional que conlleva. En cada iteración es necesario calcular el beneficio que obtendría cada posible coalición de dos participantes activos, es decir, si en un momento dado hay *n* participantes activos el número de coaliciones que deben ser estudiadas es igual al número de combinaciones de *n* elementos tomados de dos en dos: $\binom{n}{2} = \frac{n!}{2!\,(n-2)!}$ coaliciones. Esto hace que cuando el número de jugadores iniciales es muy grande, en comparación con los recursos disponibles a nivel computacional, el sistema tarde mucho en realizar las primeras iteraciones.

Este problema puede combatirse de diversas maneras. Utilizar una función característica muy simple cuyo valor para todas las posibles coaliciones se pueda calcular previamente para únicamente tener que acceder al valor correspondiente en el algoritmo de agrupamiento sería una solución sencilla aunque podría redundar en la calidad de las recomendaciones. Otra posible solución, que no tendría por qué reducir la calidad del resultado, consistiría en aumentar el valor del beneficio mínimo requerido y cambiar el algoritmo para que en el momento en que identificara una coalición que lo supere, la considerara válida y pasara a la siguiente iteración sin calcular el resto de beneficios. En este estudio se ha optado por una solución *ad hoc* consistente en realizar un proceso de clustering previo mediante técnicas tradicionales. De esta forma, se soluciona el problema del excesivo número de usuarios convirtiendo un único juego con muchos jugadores en múltiples juegos de menos jugadores. El tipo de algoritmo, la métrica y la distancia a utilizar se discutirán más adelante (véase *3. 2. 4. Proceso de Clustering Previo).*

Otra limitación a tener presente es el problema de arranque en frío descrito anteriormente. Cuando un usuario interviene por primera vez en el sistema, no se tiene suficiente información sobre él como para poder situarlo en ninguno de los clusters. Para salvar este escollo, y poder recomendar películas a los usuarios desde el primer momento se utilizan dos técnicas principalmente: 1) recomendar las películas que mejores puntuaciones tienen de media en todo el sistema y 2) obligar al usuario a puntuar un número mínimo de películas de una lista para crear su perfil en el sistema.

En el sistema que se va a implementar no existe el problema de arranque en frío debido a la restricción impuesta de utilizar únicamente usuarios que hayan puntuado al menos quinientas películas. Sin embargo, sí puede darse el caso de que cuando el algoritmo termine por la condición de parada de beneficio mínimo, queden jugadores que no se

hayan integrado en ninguna coalición. Cuando esto ocurre no se generan nuevas recomendaciones para estos usuarios. En este caso, además de las dos alternativas descritas anteriormente se pueden aplicar otras soluciones. La más sencilla consistiría en reducir el valor del parámetro de beneficio mínimo utilizado, de forma que se facilitara la formación de coaliciones. Las coaliciones que se hubieran formado previamente pueden retirarse del sistema o mantenerse para que aumente su tamaño. Tras analizar todas las posibilidades, hemos optado por aplicar una solución en dos fases: en una primera se aplicará la técnica de reducción del nivel de beneficio mínimo y en la segunda, a los usuarios que aún permanezcan aislados, se les recomendaran las películas más populares del sistema.

3. 2. 3. LA FUNCIÓN CARACTERÍSTICA

Los resultados de cualquier algoritmo de agrupación dependen de la función que determina qué grado de similitud tienen dos o más elementos, independientemente de la forma en que se manejen esas funciones. El modelo propuesto no es una excepción. Hasta ahora todas las técnicas que hemos descrito se basan en conceptos teóricos generales y en similitudes entre los sistemas de recomendación y la teoría de juegos cooperativos, sin embargo, no existe una metodología única ni totalmente fiable para construir la función característica de un juego cooperativo. A continuación se describe la lógica utilizada para generar la función característica en la que se apoyará el sistema de recomendación que se va a implementar. Si en trabajos futuros se optase por usar la solución propuesta, bastaría con adaptar la función característica al problema que se quisiese resolver ya que por definición toda función característica tiene como parámetros de entrada un conjunto de jugadores y como salida un valor real.

Cabe destacar que en este trabajo, la cantidad de información que se podría utilizar para calcular la función característica es muy amplia, como se desprende del análisis exhaustivo del conjunto de datos realizado anteriormnete. Sin embargo, al tratarse de la primera aproximación a un nuevo método y dado que los recursos disponibles son limitados se ha optado por trabajar únicamente con los valores de las puntuaciones asignadas por los usuarios a las películas. En cualquier caso, las puntuaciones, por cantidad e importancia, deben ser los datos básicos en los que se apoye cualquier medida de similitud entre usuarios. Por lo tanto, la función característica propuesta puede servir de punto de partida para la definición de otras funciones que mejoren sus resultados utilizando mayor cantidad de datos.

Otra decisión importante relacionada con la función característica, es la de aprovechar la naturaleza incremental del algoritmo para restringir el dominio de partida de la función característica a vectores de dos jugadores. El motivo de esta decisión es facilitar el proceso de medición de similitud ya que la comparación entre dos elementos es mucho más intuitiva y rápida que entre un número mayor.

Esta decisión implica la adición de una étapa más al algoritmo iterativo, consistente en convertir la coalición formada en un solo participante que la represente antes de introducirla en el conjunto de participantes. Al fin y al cabo, el objetivo de un sistema de recomendación colaborativo es obtener grupos de usuarios a cuyos miembros se les va a recomendar las mismas películas y, en la práctica, esto se traduce en tener un conjunto de

usuarios a los cuales se les asigna un mismo vector de puntuaciones a partir del cual se generan las recomendaciones correspondientes.

Para la fusión de dos participantes en uno que los represente, se ha optado por almacenar el número de jugadores que contiene cada participante y utilizar este dato para obtener la puntuación media ponderada de cada película cuando se forma la coalición. Por ejemplo, si se coaligan dos participantes que representan a 2 y 3 jugadores respectivamente y el primer participante ha asignado una puntuación de 2 a una película que el segundo tiene puntuada con un 4, el nuevo participante representará a cinco jugadores y asignará a esa película una puntuación de $\frac{2*2+3*4}{2+3} = \mathbf{3,2}$ **.** Este método se aplica en los casos en los que ambos participantes han asignado una puntuación a la película pero si sólo uno de ellos lo hubiera hecho, el nuevo participante tomaría directamente ese valor.

Teniendo en cuenta todo lo anterior, y retomando la terminología utilizada en la definición de juego cooperativo, la función característica del sistema de recomendación propuesto tendrá como parámetros de entrada dos participantes y, a través de una función de similitud *f,* devolverá el valor real asignado a la coalición de ambos. En términos matemáticos la definición se puede formalizar como sigue:

Definición 1: Sean A y B dos participantes del juego (N, v) tales que $A = \{u'_1, u'_2, \dots, u'_n\}$, $B = \{w'_1, w'_2, \dots, w'_m\}$ con $u'_i, w'_j \in N$. Se define la función característica asociada al juego como la función v: $2^n x\, 2^n \longrightarrow \mathbb{R}$, $\boldsymbol{v(A,B) = f(\gamma(u'_1 \cup u'_2 \cup \dots \cup u'_n), \gamma(w'_1 \cup w'_2 \cup \dots \cup w'_m))}$ donde $\gamma(\bigcup_{i\in I} u'_i)$ corresponde al vector de puntuaciones de A.

El vector de puntuaciones que describe a un participante *A,* $\gamma(A)$, es un vector de tamaño igual al número de películas que participan en el proceso de recomendación y cuyos valores se corresponden con las puntuaciones asignadas por el usuario a cada película, en el caso de jugadores aislados o, en el caso de coaliciones, con el resultado de la función de unión de jugadores definida a continuación.

Definición 2: Sean A y B dos participantes del juego(N, v) y sean n_A, n_B el número de jugadores que forman parte de A y B respectivamente. Sea m el número de películas involucradas en el proceso de recomendación. Se define la función de unión de participantes como la función $U: \mathbb{R}^m x\, \mathbb{R}^m \longrightarrow \mathbb{R}^m$, $U\big(\gamma(A), \gamma(B)\big) = \gamma(C)$ tal que

$$\boldsymbol{\gamma(C)_i} = \begin{cases} \dfrac{\boldsymbol{n_A * \gamma(A)_i + \ n_B * \gamma(B)_i}}{\boldsymbol{n_A + \ n_B}} \ \boldsymbol{si\ \gamma(A)_i \neq 0, \gamma(B)_i \neq 0} \\ \boldsymbol{\gamma(A)_i \quad si\ \ \gamma(A)_i \neq 0,\ \gamma(B)_i = 0} \\ \boldsymbol{\gamma(B)_i \quad si\ \gamma(B)_i \neq 0,\ \gamma(A)_i = 0} \end{cases}$$

En relación con la función de similitud *f* acudimos directamente, como justificamos más arriba, al caso concreto del dataset que nos ocupa. En el caso práctico de la recomendación de películas a partir del conjunto de datos CAMRa2010 Filmtipset Social, la función característica debe medir el beneficio de cada coalición teniendo en consideración el objetivo perseguido que es obtener agrupaciones de usuarios cuyas puntuaciones sean lo más similares posible pero sin olvidar que el fin último del sistema es generar nuevas recomendaciones. Por este motivo, el número de películas puntuadas entre todos los jugadores que forman la coalición debe formar parte de la función característica junto al

nivel de similitud de las puntuadas en común. Estas dos características deben ser pues los pilares sobre los que se construya la función *f* (***f = S + T***). Es importante, además, que la función mantenga un balance entre ambos factores y que los valores obtenidos se normalicen, ya que si se le da demasiado valor a la cantidad de películas puntuadas o no se normaliza, se beneficiará a las coaliciones formadas por un mayor número de elementos.

Respecto a la similitud de puntuaciones, el estudio previo del conjunto de datos reveló que se pueden dividir en tres grupos: las puntuaciones 1 y 2 para películas que no han gustado a los usuarios, las puntuaciones 4 y 5 para aquellas que han gustado y el valor 3 para las que se han quedado en un término medio. Esta estructura conlleva que cualquier diferencia mayor de un punto entre las puntuaciones de dos participantes implica que no están de acuerdo sobre la calidad de la película y, por tanto, la función *f* debe reflejar este hecho. Siguiendo el mismo razonamiento, una diferencia menor o igual a un punto indica cierta concordancia en las opiniones de los usuarios. Obviamente, si las puntuaciones coinciden el beneficio obtenido es máximo. A continuación se detalla nuestra propuesta para la parte de la función *f* correspondiente al grado de similitud:

- Sea *Eq* el número de películas que ambos participantes han puntuado igual.
- Sea *Lt1* el número de películas a las que los participantes han asignado puntuaciones que se diferencian en 1 punto o menos.
- Sea *Gt1* es el número de películas cuyas puntuaciones se diferencian en más de un punto.
- Sea *TM* el número total de películas puntuadas por alguno de los participantes.
- Sean α, β, δ, ε los coeficiente aplicados a *Eq*, *Lt1*, *Gt1* y *TM* que deben determinarse mediante iteraciones prácticas. El razonamiento llevado a cabo anteriormente implica que δ<0 y que α ≥ β.

$$\boldsymbol{S = \frac{\alpha * Eq + \beta * Lt1 + \delta * Gt1}{\varepsilon * TM}}$$

Por otro lado, se asigna el coeficiente η al número de películas que únicamente uno de los participantes ha puntuado OO (*Only-One*) y se divide entre el número total de películas para obtener la otra parte de la función *f:*

$$\boldsymbol{T = \frac{\eta * OO}{TM}}$$

Por lo tanto, la función *f* queda definida de la siguiente manera:

$$\boldsymbol{f = S + T = \frac{\alpha * Eq + \beta * Lt1 + \delta * Gt1}{\varepsilon * TM} + \frac{\eta * OO}{TM}}$$

3. 2. 4. PROCESO DE CLUSTERING PREVIO

La decisión de realizar un proceso de clustering previo conlleva seleccionar qué tipo de algoritmo utilizar y cómo configurarlo. En este caso, para mantener la estructura del método propuesto, se ha optado por utilizar un algoritmo aglomerativo con la distancia

euclídea como métrica y el método de Ward como sistema de medición de la distancia entre clusters.

Con el fin de que los clusters obtenidos en este proceso sean una primera aproximación válida al resultado final del sistema de recomendación, el algoritmo de clustering previo también se alimentará con los vectores de puntuaciones asignadas por los usuarios a las películas. Sin embargo, para evitar que este filtrado previo sea demasiado preciso y enmascare los resultados, se han sustituido las puntuaciones dadas por valores binarios, representando películas puntuadas o no. De esta manera, se consigue un primer filtrado que agrupará usuarios con bastantes películas puntuadas en común y será responsabilidad del proceso final crear agrupaciones más precisas, teniendo en cuenta, no sólo el que se hayan puntuado o no, sino también que las puntuaciones sean similares.

La razón de la elección del método de Ward como medida de la distancia entre clusters radica en la necesidad de obtener clusters que sean lo más compactos posible. Esta característica se traduciría en clusters formados por usuarios que comparten un gran número de películas puntuadas en común aunque puede generar mayor número de ellos. Además, los estudios referenciados en el capítulo de introducción señalaban este método como el de mayor rendimiento (véase *2. 2. 2. Clasificación de los Algoritmos de Clustering*). Por otra parte, el uso de la métrica euclídea se debe a que su sencillez aligera la carga computacional del algoritmo y, además, es una de las más usadas en combinación con el método de Ward.

Una vez obtenidos los resultados del algoritmo, es necesario decidir qué fase del proceso aglomerativo es óptima para el sistema. El criterio principal para tomar esta decisión será la proporción entre el número de clusters y el número de usuarios por cluster ya que el objetivo principal de este proceso es dividir el conjunto de usuarios para facilitar los procesos computacionales. Además, se aprovecharán al máximo los datos proporcionados por el algoritmo para, a través de un análisis de los procesos aglomerativos llevados a cabo entre fases, intentar seleccionar una fase que sea lo más estable posible.

3. 2. 5. CONSIDERACIONES FINALES

A continuación, se incluye un diagrama (Figura 12) de las fases que componen el sistema de recomendación de películas basado en teoría de juegos propuesto. Además, a modo de resumen, se describen las características básicas de estas fases:

1) **Filtrado inicial.** Se seleccionarán los usuarios que hayan puntuado más de 500 películas y cualquier tipo de información contenida en el dataset que tenga relación con ellos.
2) **Clustering aglomerativo tradicional.** Se ejecutará un algoritmo de clustering aglomerativo basado en la métrica euclídea y el método de Ward como medidas de distancia. Dicho algoritmo se alimentará con vectores de puntuaciones binarios que representarán la puntuación o no de una película por parte de un usuario.
3) **Fase 1 del Sistema de Recomendación basado en Teoría de Juegos.** Por cada uno de los clusters obtenidos en la fase anterior se generará un juego cooperativo con tantos jugadores como usuarios formen el cluster. Utilizando la función característica como medida de similitud y a través de iteraciones aglomerativas se

generarán agrupaciones de usuarios con un vector de puntuaciones asociado que constituirán la primera solución del sistema de recomendación.

4) **Fase 2 del Sistema de Recomendación basado en Teoría de Juegos.** El uso de una condición de parada ocasionar que algunos jugadores permanezcan aislados al término de la primera fase. Para evitar dejar a estos usuarios sin nuevas recomendaciones se llevará a cabo una segunda fase con las mismas características que la anterior pero reduciendo el valor del beneficio mínimo. Las agrupaciones obtenidas en esta segunda fase se añadirán a las obtenidas anteriormente y, si siguen existiendo usuarios aislados, se juntarán en un único cluster y se someterán a una tercera fase de agrupación.

5) **Fase 3 del Sistema de Recomendación basado en Teoría de Juegos.** En caso de llegar a una tercera fase, el valor del beneficio mínimo se irá reduciendo progresivamente hasta alcanzar un valor mínimo. Si se diese el caso de que aún quedasen usuarios aislados querría decir que no tienen casi nada en común con el resto y se les aplicarían recomendaciones estadísticas, es decir, se les recomendarían aquellas películas que han obtenido mejores puntuaciones medias teniendo en cuenta todos los usuarios del sistema.

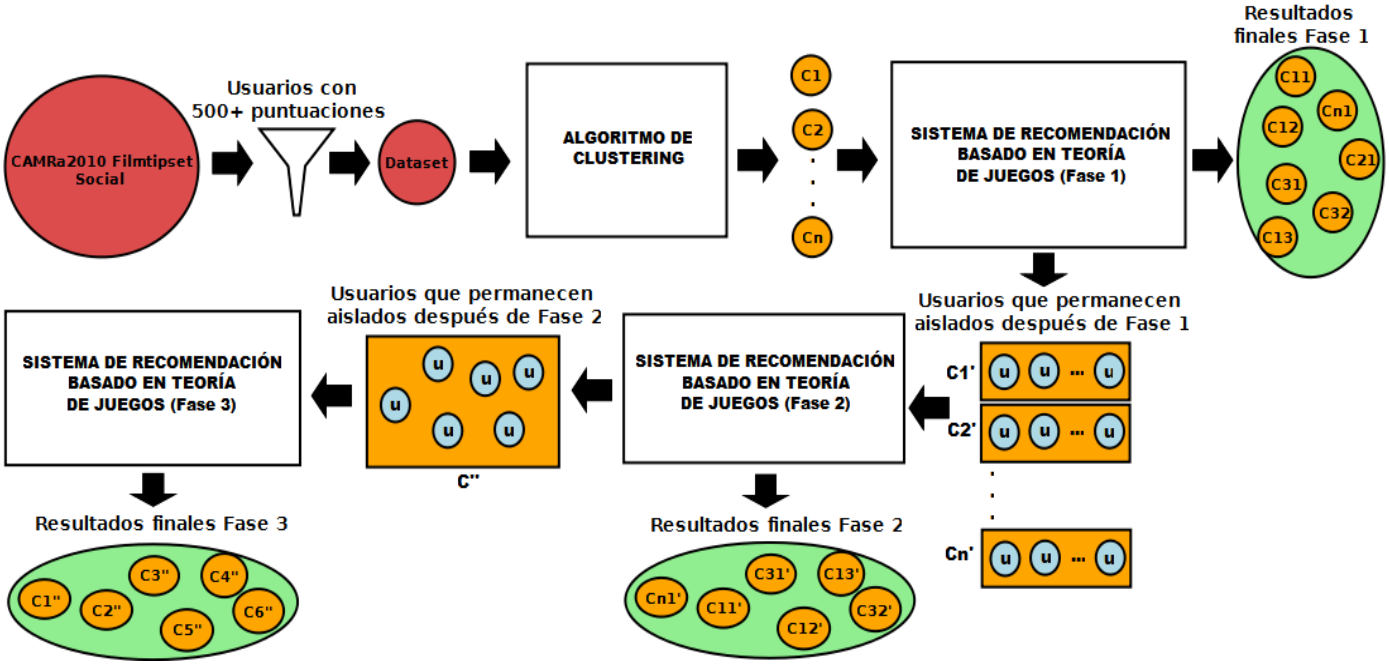

Figura 12.- Fases del modelo de recomendación basado en teoría de juegos.

3. 3. IMPLEMENTACIÓN DEL SISTEMA DE RECOMENDACIÓN

Antes de llevar a cabo la implementación de las distintas etapas del sistema de recomendación propuesto hemos analizado distintas herramientas disponibles, con el fin de encontrar aquella que mejor se adapte a las necesidades de nuestro modelo.

3. 3. 1. TECNOLOGÍAS EXISTENTES

La etapa principal del modelo propuesto es el algoritmo de agrupamiento basado en juegos cooperativos y, además, por tratarse de una nueva visión basada en la unión de dos

campos de conocimiento distintos, su implementación es la que más retos presenta. Por este motivo, la elección del lenguaje de programación se ha orientado a disponer de la mayor cantidad de herramientas posibles para implementar correctamente el sistema.

Otra de las características importantes para cualquier sistema de recomendación es que pueda trabajar una cantidad de datos escalable con unos tiempos de ejecución aceptables. Por lo tanto, a la hora de seleccionar el lenguaje de programación también se ha de tener en cuenta su capacidad para manejar grandes cantidades de datos y realizar cálculos con ellos.

Por último, y de cara a trabajos futuros, es preferible utilizar tecnologías estándar, de forma que no sean necesarios tener gran cantidad de conocimientos sobre el lenguaje o la herramienta para poder entender, utilizar y extender el producto generado.

Con estos requisitos, se realizó una búsqueda de librerías que proporcionasen estructuras y funcionalidades relacionadas con la teoría de juegos obteniéndose un número de resultados bastante reducido. Este hecho resultó a la vez interesante y sorprendente ya que daba a entender que la teoría de juegos hasta el momento no está siendo utilizada con toda su potencia. Entre las herramientas encontradas destaca la librería **TUGlab** desarrollada por profesores de la Universidad de Vigo. Esta librería, desarrollada en lenguaje **MATLAB**, contiene gran cantidad de métodos para estudiar juegos cooperativos de utilidad transferible, centrándose en los aspectos geométricos de los mismos. Se trata de una librería bastante completa pero tiene la restricción de que está preparada para juegos en los que participan únicamente tres o cuatro jugadores, por lo que no podía ser utilizada para el desarrollo del sistema de recomendación propuesto.

Continuando con la búsqueda de lenguajes de programación orientados a cálculos matemáticos, se encontró que **SAGEMATH** incluye una librería orientada a juegos cooperativos con un número finito de jugadores. Esta librería incluye la estructura de datos *CooperativeGame* que se define proporcionando el vector de jugadores y la función característica a utilizar, y entre sus funciones se encuentra el cálculo de los valores de Shapley correspondientes al juego definido. Dado que el sistema propuesto está basado en la estructura de un juego cooperativo pero tiene bastantes diferencias con uno clásico, la herramienta buscada debe proporcionar funciones a bajo nivel, para poder ser manipuladas y combinadas de forma que se adapten al sistema propuesto. Es en este aspecto donde reside el mayor inconveniente de esta librería ya que las funciones proporcionadas cumplen su cometido cuando se trata de resolver un problema clásico y sencillo de juegos cooperativos pero resultan demasiado rígidas para ser utilizadas en otro tipo de problemas.

Otro de los lenguajes de programación que dispone de librerías y herramientas utilizables para trabajar en problemas de teoría de juegos es **Python**. El proyecto ***Gambit*** ha desarrollado un paquete de software de código abierto que proporciona una interfaz gráfica muy útil. Este programa se centra en la parte estadística y probabilística de la teoría de juegos y tiene una gran versatilidad a la hora de trabajar con todo tipo de juegos ya que, como puede apreciarse en la Figura 13, permite modelar todas las fases de decisión de un juego. Sin embargo, no está pensado para ser utilizado con un gran número de jugadores y, por tanto, no cumple los requisitos del sistema que se va a implementar.

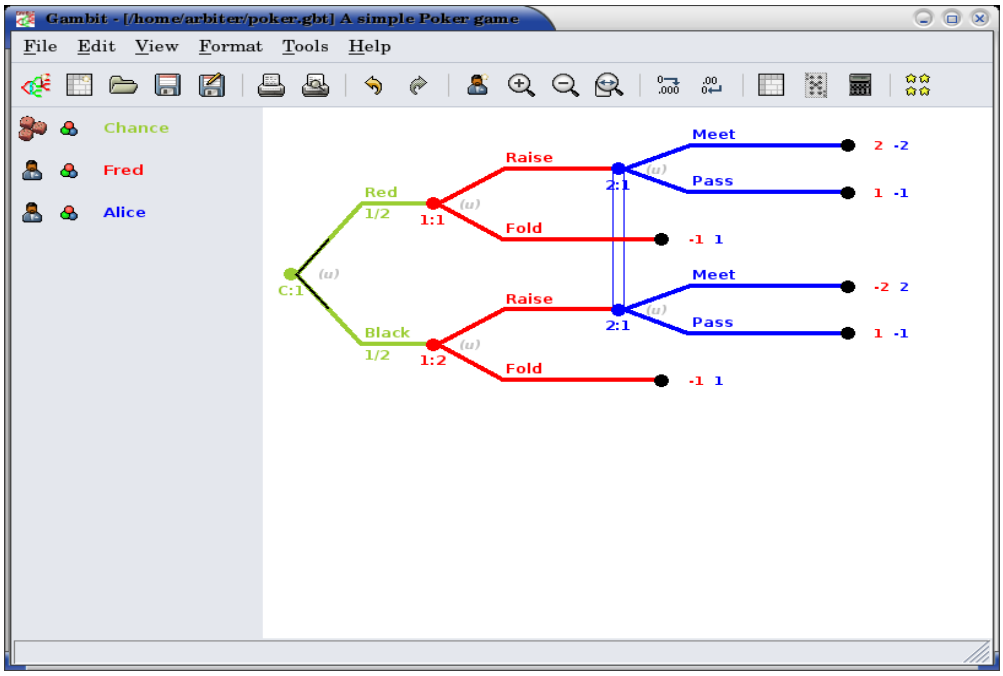

Figura 13.- Modelización de una partida de póker en la herramienta *Gambit*.

Después de analizar todas estas herramientas y comprobar que no existe ninguna capaz de proporcionar la funcionalidad requerida, optamos por utilizar un lenguaje preparado para trabajar de forma eficiente con grandes cantidades de datos e implementar las funciones necesarias en él. Finalmente, el lenguaje de programación elegido fue **R** que, aunque no posee ninguna librería especializada en teoría de juegos, sí proporciona estructuras de datos y funcionalidades que permiten realizar todas las operaciones necesarias con grandes cantidades de datos de una manera fácil y eficaz. Además, el propio lenguaje contiene librerías especializadas en algoritmos de clustering que facilitarán la ejecución de la primera etapa del sistema de recomendación.

3. 3. 2. PREPARACIÓN DE LOS DATOS: FILTRADO Y CLUSTERING

La base de datos proporcionada por los organizadores del concurso CAMRa viene dado en formato correspondiente al gestor de bases de datos Microsoft Access por lo que el filtrado previo del conjunto de datos se realizó aprovechando la funcionalidad de creación de vistas que ofrece este programa. Para ello, bastó con construir una serie de *queries* en lenguaje SQL y aplicarlas para crear las vistas. Una vez hecho esto, y para poder utilizar los datos en los programas escritos en R se exportaron las vistas a ficheros CSV.

Los ficheros obtenidos fueron leídos en R y automáticamente se generaron unas estructuras propias del lenguaje denominadas como ***datasets*** que tienen asociadas una serie de funciones que permiten la realización de un gran número de operaciones con la información contenida en ellas. Por ejemplo, se eliminaron entradas repetidas correspondientes a puntuaciones asignadas por un mismo usuario a una misma película, manteniéndose únicamente la más reciente.

Para terminar de preparar los datos para el proceso de clustering, se transformó el *dataset* de puntuaciones en una matriz con una fila por cada usuario y una columna por cada

película. A partir de esta matriz, se generó también la matriz binaria correspondiente sustituyendo las puntuaciones por unos.

La librería ***cluster*** de R contiene numerosas funciones que permiten realizar todo tipo de procesos de clustering[2]. Por ejemplo, *agnes*, *diana* y *mona* se utilizan para ejecutar algoritmos jerárquicos mientras que *pam*, *clara* y *fanny* implementan algoritmos no jerárquicos. La función ***agnes*** es la encargada de ejecutar procesos de clustering aglomerativos y, por tanto, se utilizó para el proceso de clustering previo de nuestro sistema. Esta función recibe como parámetros una matriz cuyas filas representan a cada una de las observaciones a clasificar y cada columna corresponde a un atributo de esas observaciones. Además, a la función se le puede especificar qué métrica y qué método de distancia entre clusters debe utilizar. La propia librería tiene implementadas las métricas y los métodos más utilizados en la práctica, lo cual resulta de gran ayuda. En particular, tanto la métrica euclídea como el método de Ward están incluidos en la librería.

El resultado de la ejecución de esta función es un objeto de la clase *agnes* que contiene toda la información del proceso de clustering aglomerativo llevado a cabo (usuarios agrupados en cada paso del algoritmo y distancia entre ellos, resultados parciales, el dendrograma final, etc) y que permite seleccionar la fase de unión óptima. Una vez seleccionado el número de clusters que se quieren utilizar, el dendrograma proporciona los identificadores de los usuarios que pertenecen a cada uno de ellos, de forma que podrán ser utilizados fácilmente como datos de entrada de la siguiente etapa del sistema.

3. 3. 3. ALGORITMO DE RECOMENDACIÓN BASADO EN TEORÍA DE JUEGOS

Cada uno de los clusters obtenidos en la fase anterior constituirá el conjunto de partida para la ejecución del algoritmo de recomendación basado en teoría de juegos. Como se describió en la sección de diseño, el sistema propuesto utiliza un algoritmo iterativo en el que en cada repetición se coaligan dos participantes. Para poder mantener la información del estado del juego en cada momento, así como las operaciones llevadas a cabo, se utilizan las estructuras de datos siguientes:

- **Participantes activos**: se trata de un objeto de tipo *dataset* que contiene toda la información necesaria sobre los participantes para, en caso de formar parte de una nueva coalición, poder calcular los datos que la describan. En el caso de las coaliciones identifica sus miembros mediante una cadena de texto que contiene los identificadores de los jugadores separados con el carácter '@'. Además, la estructura guarda el número de jugadores que forma cada participante (1 en el caso de usuarios individuales) y el valor de la función característica cuando se formó esa coalición (0 cuando aún permanecen aislados). De esta manera, cuando se añaden nuevos jugadores a una coalición basta con añadir al final de la cadena sus identificadores, sumar el número de nuevos miembros a los ya existentes y actualizar el valor de la función característica.
- **Matriz de puntuaciones**: la función característica utilizada se alimenta de las puntuaciones asignadas por cada participante a cada película. Por lo tanto, es necesario mantener dichas puntuaciones actualizadas ya que cada vez que se

[2] Como curiosidad indicar que estas funciones tienen todas ellas nombres de mujer.

genera una coalición se obtiene un nuevo vector de puntuaciones a partir de las de los dos participantes que van a cooperar y se asocia al nuevo participante. Además, una vez concluido el algoritmo, el vector correspondiente a cada participante será utilizado para comprobar cuan precisas son las recomendaciones asignadas a cada grupo de usuarios.

- **Log de coaliciones**: con el objetivo de estudiar el funcionamiento del algoritmo se ha creado un fichero que registre las uniones realizadas en cada iteración del algoritmo, siguiendo la estructura de dendrograma generado por los algoritmos jerárquicos. A través de este fichero se pueden detectar algunos comportamientos del algoritmo que ayudan a mejorar la función característica como, por ejemplo, que un cluster vaya recibiendo jugadores de uno en uno y no se creen otros, lo cual indicaría que la función característica está beneficiando a los clusters con mayor número de jugadores.

La decisión de utilizar un lenguaje de programación que no proporciona ningún tipo de funcionalidad relacionada con la teoría de juegos obliga a implementar una **librería de funciones** que permitan la construcción del algoritmo diseñado. A la hora de desarrollar esta librería, no solo nos hemos centrado en la implementación de la funcionalidad requerida sino que también hemos tratado de utilizar las estructuras de datos propias de R y sus funcionalidades para, de esta manera, poder optimizar el rendimiento de la librería generada.

La primera de las funciones a implementar es la **función característica**, que recibirá los identificadores de los dos participantes cuya coalición se quiere analizar así como la matriz de puntuaciones y devolverá el valor de similitud asignado a la coalición. Para ello, primero calculará el número de películas que han puntuado ambos y el número de películas que han puntuado por separado. Hecho esto, calculará cuántas películas de las que han sido puntuadas por ambos participantes han recibido puntuaciones similares, cuántas han recibido puntuaciones que difieren en un punto o menos y cuántas han diferido en más de un punto. Una vez obtenidos estos valores solo quedará aplicar la fórmula de f descrita anteriormente con los valores de α, β, δ, ε y η seleccionados. De esta manera, cuando se quiera construir una nueva función característica sólo será necesario cambiar los valores asignados a estos parámetros.

Además de la función característica, la otra parte del algoritmo que tiene mayor importancia es la **formación de las coaliciones**. Por este motivo, se ha creado una función que recibe los identificadores de los participantes que van cooperar y el valor de la función característica que se les ha asociado y con estos datos accede a las estructuras descritas en el comienzo de esta sección para realizar los cambios pertinentes. En primer lugar concatena los identificadores y suma el número de jugadores de ambos participantes para, añadiendo el valor de la función característica recibido como parámetro, crear una nueva entrada en la estructura de participantes activos. Una vez hecho esto, elimina las entradas correspondientes a los jugadores que se han unido.

Esta función también se encarga de actualizar la matriz de puntuaciones. Para ello, accede a través de sus identificadores a los vectores de puntuaciones de cada uno de los participantes que va a coaligar y, utilizando el número de jugadores que van a participar en la coalición obtenido al actualizar el *dataset* de jugadores activos, genera el vector de

puntuaciones del nuevo participante aplicando las reglas descritas en la sección anterior. Finalmente, elimina de la matriz de puntuaciones las entradas correspondientes a los participantes que cooperan e inserta el vector de la coalición.

Una vez preparadas estas dos funciones, se puede proceder a implementar el sistema de recomendación propuesto. El sistema recibirá como parámetros una lista con los identificadores de los usuarios que van a participar, la matriz de puntuaciones asignadas por estos usuarios y el valor del beneficio mínimo que se quiere aplicar. Antes de iniciar el proceso iterativo, la función inicializará las estructuras que va utilizar y creará el log de coaliciones. Hecho esto, se generará un bucle cuyas condiciones de parada serán que la bandera que indica si la anterior iteración superó el valor de beneficio mínimo sea verdadera y que el número de participantes activos sea mayor que uno. El cuerpo del bucle comenzará obteniendo todas las posibles coaliciones que se pueden formar en esa iteración. Para ello calculará todas las combinaciones existentes de dos participantes activos y, para cada una de ellas, se invocará la función característica almacenando temporalmente el resultado. Cuando se hayan obtenido los valores correspondientes a todas las combinaciones, la función seleccionará el máximo de todos ellos y comprobará si supera el beneficio mínimo establecido. Si no lo hace, la iteración parará, se asignará el valor falso a la bandera y la función finalizará al comprobar las condiciones del bucle. Si, por el contrario, el valor seleccionado es mayor que el beneficio mínimo establecido, se procederá a la formación de la coalición a la que corresponde el valor, utilizando la función correspondiente. Además de las operaciones llevadas a cabo en ella, se añadirá una entrada en el log de coaliciones que contendrá los identificadores de los dos participantes.

Cuando el sistema termina, los resultados finales, que se encuentran en las estructuras de datos, se escriben en ficheros con formato CSV a los que se podrá acceder más tarde para comprobar el rendimiento obtenido. Esta información permite, una vez acabada la primera fase, identificar los jugadores que han permanecido aislados en cada cluster.

Para la segunda fase basta con retirar de cada estructura de participantes las coaliciones formadas, disminuir el valor del beneficio mínimo y volver a ejecutar la función. Finalmente, para llevar a cabo la tercera fase, se juntarán en un solo cluster todos los jugadores que aún permanezcan aislados y se ejecutará la función de agrupamiento una vez más.

Por si se diese el caso de qué, después de las tres fases, aún existieran usuarios aislados se ha implementado una función más que calcula la puntuación media asignada por todos los usuarios del sistema a cada película. De esta forma, si al cruzar los resultados del sistema de recomendación con los del conjunto de puntuaciones de test, alguna entrada no tuviese asignada una puntuación se le asignaría la calculada por esta función.

4. EXPERIMENTACIÓN Y RESULTADOS

A continuación se presentan los experimentos prácticos llevados a cabo utilizando las funciones descritas en el capítulo anterior.

4. 1. CLUSTERING PREVIO

Utilizando los datos previamente filtrados mostrados en la Tabla 2, se genera la matriz binaria de puntuaciones que tiene unas dimensiones de **1.623** filas por **21.603** columnas. Esta matriz se pasa como parámetro a la función *agnes* y se seleccionan la distancia euclídea y el método de Ward como métricas.

Al utilizar un número de observaciones mucho más grande que los utilizados normalmente, el dendrograma obtenido resulta ilegible, ya que intenta incluir en su base a todos los jugadores. Sin embargo, la librería permite cortar el dendrograma en ramas para poder ser estudiado. En la Figura 14 se muestra a modo de ejemplo una de las ramas del dendrograma correspondiente a uno de los clusters obtenidos más pequeños y que será utilizado en fases posteriores. En el eje Y se muestra la distancia entre jugadores y clusters. Este valor sirve, en ocasiones, como medida para la elección de la fase final del algoritmo de clustering. Por ejemplo, en esta rama se observa qué los usuarios que tienen como identificadores los números *82800328* y *91864084* son los que tienen un mayor grado de similitud y se encuentran a una distancia ligeramente inferior a 40. Si se tomase la decisión de utilizar los clusters que se encuentran a una distancia menor de 40 como resultado final del algoritmo, únicamente esta pareja y la formada por los identificadores *78378581* y *93783234* generarían clusters de más de un elemento, quedando aislados el resto de jugadores.

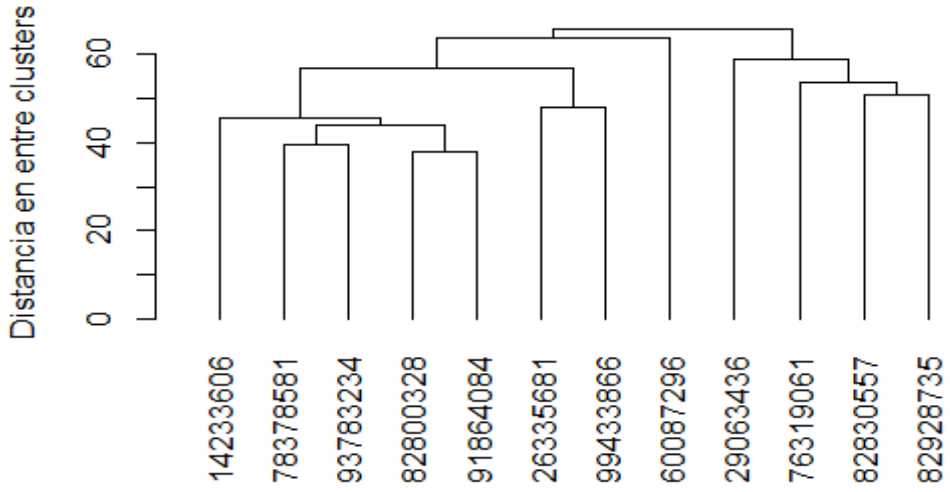

Figura 14.- Rama del dendrograma obtenido con el algoritmo de clustering aglomerativo utilizando la métrica euclídea y el método de Ward.

El uso de la distancia entre clusters como forma de selección de la fase de finalización implica un estudio profundo de la estructura del dendrograma y del resto de datos proporcionados por la función. Sin embargo, dado que este proceso no es más que una aproximación al objetivo final del estudio, se va a determinar la fase final teniendo en

cuenta la proporción entre el número de clusters y el número de usuarios que contiene cada uno de ellos, dejando como trabajo futuro tanto el estudio pormenorizado de la estructura del dendrograma como la experimentación con distintas cantidades de clusters.

Debido a nuestras restricciones computacionales, es conveniente obtener entre diez y veinte clusters intentando que ninguno de ellos supere los cuatrocientos usuarios. Idealmente, el objetivo sería encontrar un grupo de clusters que tuviesen las mismas dimensiones pero, al trabajar con datos reales, esto es muy difícil que ocurra. Para facilitar la toma de decisión, se ha generado la imagen de la Figura 15 que muestra la relación entre el número de clusters y el número de usuarios contenidos en cada uno de ellos, para el intervalo entre diez y veinte clusters. En ella se observa cómo a partir de trece clusters empiezan a aparecer algunos con un número muy reducido de usuarios y este hecho no es deseable, ya que limita en gran medida los resultados de los procesos posteriores. Por lo tanto, se ha seleccionado la división en **trece clusters** como resultado óptimo.

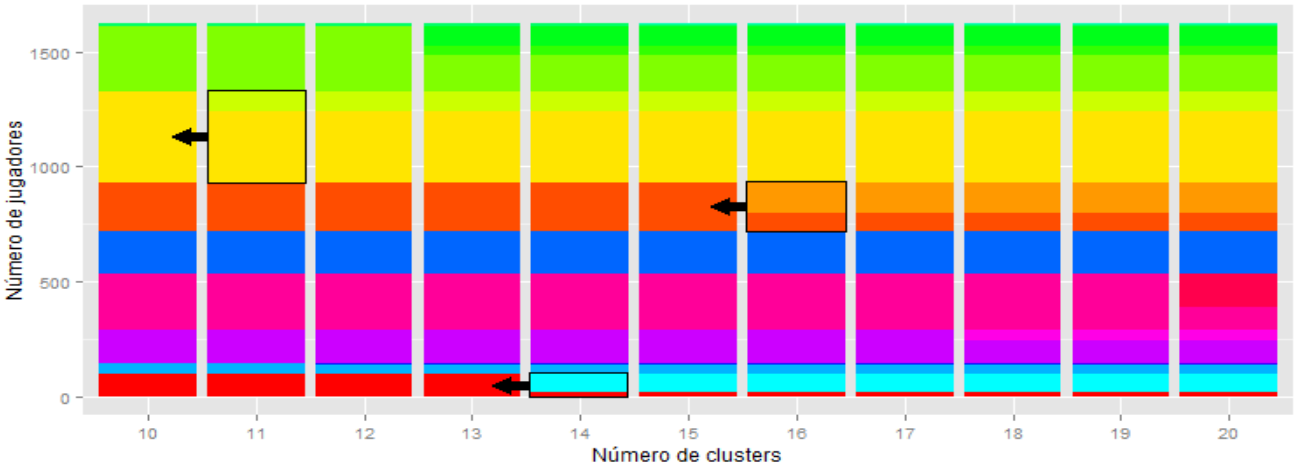

Figura 15.- Relación entre el número de clusters y el número de usuarios contenido en cada uno de ellos a partir de los resultados del algoritmo de clustering previo.

En la Tabla 3 se incluye el número de usuarios contenidos en cada uno de los clusters seleccionados como datos de entrada del algoritmo de recomendación.

Tabla 3.- Número de usuarios contenidos en cada cluster tras la ejecución del algoritmo de clustering aglomerativo.

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
100	36	147	246	184	213	306
Cluster 8	**Cluster 9**	**Cluster 10**	**Cluster 11**	**Cluster 12**	**Cluster 13**	
159	41	86	6	87	12	

4. 2. RESULTADOS DEL SISTEMA DE RECOMENDACIÓN

El factor que va a determinar la calidad de los resultados del sistema de recomendación es la capacidad de la función característica para describir correctamente la medida de similitud entre participantes. Previamente se han identificado las componentes principales comunes a todas las funciones características que se van a generar, pero el

ajuste de los parámetros α, β, δ, ε y η dependerá de los resultados intermedios que se vayan obteniendo.

Dado que el coste computacional de ejecutar la función de recomendación para todos los clusters es muy alto[3], es necesario evitar tener que hacerlo para cada función característica propuesta. Para ello, se ha seleccionado un subconjunto de clusters (1, 3, 5, 6, 8 y 12) que serán utilizados en un test de resultados parciales que permitirá determinar si continuar el proceso de recomendación o modificar la función característica y empezar de nuevo.

Durante el proceso de experimentación se han construido y probado quince funciones características distintas seleccionándose entre ellas las cuatro que mejores resultados han generado. En la

Tabla 4 se muestran los valores de los parámetros utilizados en estas cuatro funciones.

Tabla 4.- Valores de los parámetros para las cuatro funciones características que han generado mejores resultados.

	α	**β**	**δ**	**ε**	**η**
F. característica 1	4	2	-1	1	1/2
F. característica 2	4	2	-1.5	1	1/2
F. característica 3	4	2	-2	1	1/3
F. característica 4	4	2	-3	1	1/2

Cabe destacar que todas ellas comparten los valores de los parámetros correspondientes al número de puntuaciones iguales (α), al número de puntuaciones que se diferencian en menos de un punto (β) y al primer término de la función *f* (ε). Además, para evitar tener un parámetro más que influyese en la comparación de las funciones, el valor del beneficio mínimo en cada fase del sistema se ha fijado en los mismos valores para todas las funciones características. Así, en la primera fase se asigna un valor de **0.9** al beneficio mínimo, en la segunda fase se reduce a **0.8** y en la tercera fase a **0.7**. Estos valores se han determinado a partir de un pequeño análisis consistente en estudiar los valores estadísticos de la primera iteración de la primera fase del sistema en cada uno de los clusters con la primera función característica que se construyó (función característica 1). El valor del beneficio mínimo utilizado (0.9) corresponde al tercer cuartil del conjunto de valores obtenidos, de manera que quedasen el 25% de las posibles coaliciones por encima de él. De esta forma, las cuatro funciones cuyos resultados se van a analizar se diferencian únicamente en el peso restado por cada puntuación común que difiere en más de un punto (δ) y en el parámetro correspondiente a la parte de la función que mide la proporción de películas que solo uno de los participantes ha puntuado con respecto al total de películas (η).

4. 2. 1. EVOLUCIÓN DEL SISTEMA DE RECOMENDACIÓN FASE A FASE

Uno de los aspectos que es importante analizar con respecto a las funciones características es su comportamiento durante cada una de las fases del sistema. En particular, es

[3] La ejecución de la primera fase sobre el cluster 7 tiene una duración media de **12** horas y la ejecución de la primera fase tarda una media de **36** horas. Como valor extremo, la tercera fase, utilizando la función característica 3, se ejecutó sobre un cluster de 347 usuarios y tardó **28** horas.

interesante comprobar qué porcentaje del total de usuarios se agrupa en cada fase del algoritmo ya que, al utilizar el mismo valor de beneficio mínimo para todas las funciones características, los clusters obtenidos en cada fase con cada una de ellas tendrán el mismo nivel de similitud. De esta manera, se puede comprobar si todas las funciones características tienen comportamientos similares en todas las fases del algoritmo o, por el contrario, hay diferencias entre las más restrictivas y las que lo son menos. Por ejemplo, después de la primera fase la función característica 3 mantendrá aislados mayor cantidad de usuarios que la función característica 1 ya que siempre asignará un valor menor a todas las coaliciones, ya que los valores asignados a todos sus parámetros son menores o iguales que los que utiliza la función característica 1. Sin embargo, una vez comience la segunda fase cada una de las funciones características podrá crear distintas coaliciones, ya que partirán de conjuntos de jugadores distintos al haber sido eliminadas las coaliciones formadas en la fase anterior. De hecho, podría darse el caso de que al final del proceso de recomendación la función característica 3 mantuviese menos usuarios aislados que la función característica 1.

En la Figura 16 se muestra el porcentaje de usuarios que forman parte de algún cluster al final de cada fase del sistema para cada función característica. Puede observarse que las funciones 1 y 2 tienen un comportamiento similar durante las tres fases del proceso manteniéndose por encima del resto.

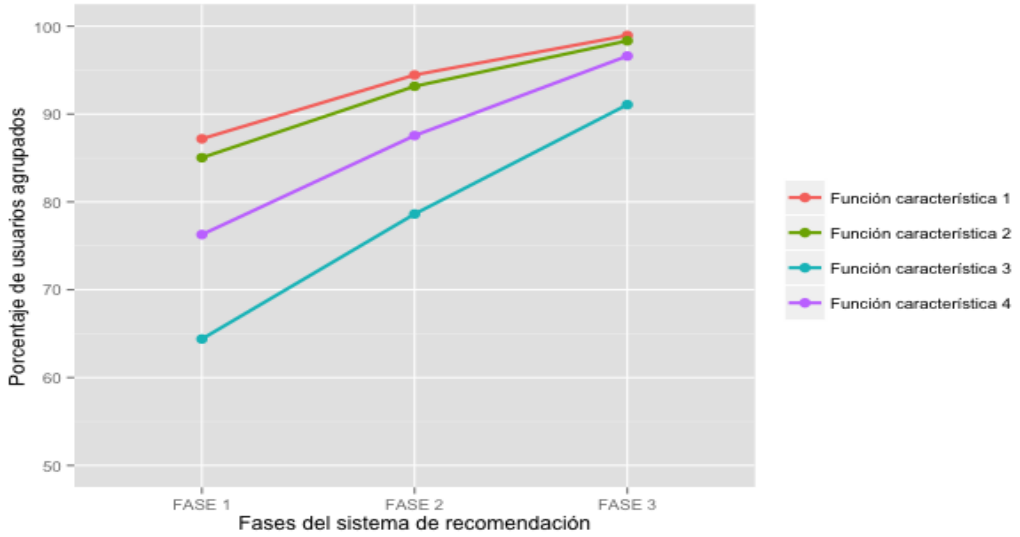

Figura 16.- Porcentaje de usuarios agrupados por cada función característica al final de cada fase del sistema de recomendación.

Es también destacable el crecimiento, en las dos últimas fases, de las funciones características 3 y 4 que comenzaron agrupando menor número de usuarios al ser más restrictivas. Por último, cabe destacar que la función característica 3 forma parte de las cuatro funciones características que mejores resultados finales han obtenido dejando casi un 10% de los usuarios aislados tras la tercera fase. De este hecho se puede concluir qué las recomendaciones obtenidas con la función característica 3 son más precisas que las obtenidas con el resto de funciones descartadas, ya que el procedimiento de completar las recomendaciones del conjunto de test con las puntuaciones medias de las películas es común a todas las funciones.

Además de comprobar cuántos usuarios se agrupan en cada fase, también es interesante conocer qué porcentaje del conjunto de test representan esos usuarios. Esta característica es importante ya que en el caso de un sistema de recomendación real pueden existir requisitos temporales, por lo que sería positivo obtener el mayor número de recomendaciones posible en las fases tempranas del sistema, de forma que pudiesen mostrarse sin necesidad de haber terminado el proceso. La Figura 17 muestra los porcentajes obtenidos en este estudio.

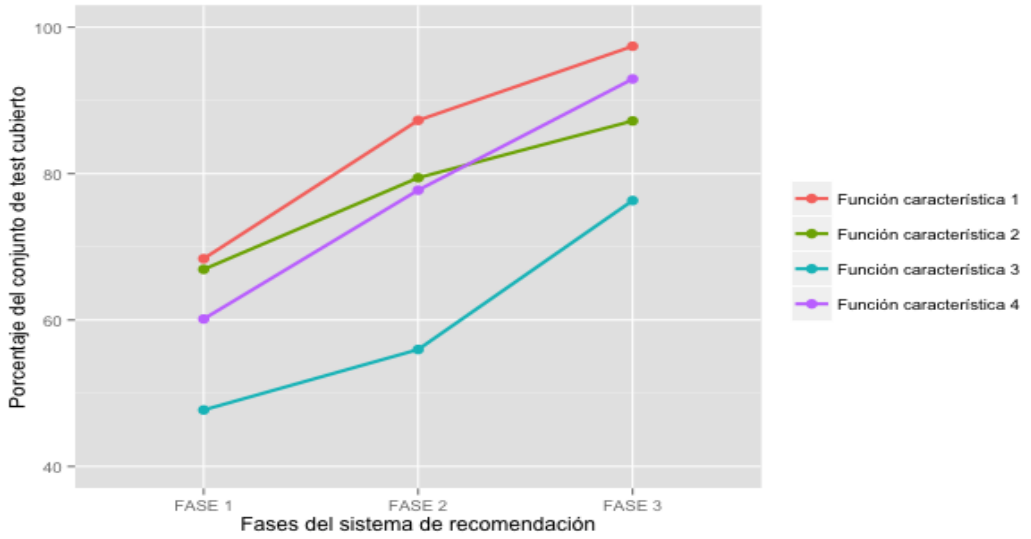

Figura 17.- Porcentaje de entradas del conjunto de test cubiertas por cada función característica al final de cada fase del sistema de recomendación.

Si comparamos los resultados de ambas figuras lo primero que llama la atención es que la función característica 4, aun dejando más usuarios aislados al final de la tercera fase, genera recomendaciones para un porcentaje de entradas significativamente más alto que la función característica 2. Este hecho indica que la función característica 4 genera clusters con mayor número de jugadores que cubren conjuntos de películas más amplios mientras que la función característica 2 crea mayor número de clusters con menor densidad de usuarios. Este tipo de información sobre el tipo de agrupaciones que genera cada una de las funciones puede ser muy útil a la hora de construir otras en el futuro.

4. 2. 2. DETERMINACIÓN DEL RENDIMIENTO: MEDIDAS DE ERROR

Para determinar el grado de eficacia de las recomendaciones obtenidas es común emplear métricas estándar de la literatura. En nuestro caso las métricas empleadas han sido el error medio absoluto (*MAE* en sus siglas en inglés) (Herlocker, Konstan, Terveen, & Riedl, 2004) cuyo valor corresponde con la siguiente función:

$$\mathbf{MAE} = \frac{1}{n} \sum_{i=1}^{n} |\boldsymbol{pred_i} - \boldsymbol{test_i}|$$

siendo n el número de entradas del conjunto de ratings de test, $pred_i$ la puntuación que el sistema de recomendación predice para el par usuario-película i y $test_i$ la puntuación real asignada por el usuario a la película.

Por otro lado, también se ha calculado el error cuadrático medio (*RMSE*) (Herlocker, Konstan, Terveen, & Riedl, 2004) que, con la misma terminología, responde a la siguiente expresión:

$$\textbf{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(pred_i - test_i)^2}{n}}$$

Además de hallar los errores cometidos con los vectores de puntuaciones obtenidos se ha calculado también el error que se cometería si se redondeasen las puntuaciones asignadas (MAE Round y RMSE Round). De esta manera, se puede comprobar que error se cometería en el caso de tener que asignar puntuaciones discretas a las películas. En la Tabla 5 se muestran los errores calculados para cada función característica al finalizar la ejecución del sistema de recomendación.

Tabla 5.- Errores cometidos por las cuatro funciones características.

	MAE	MAE Round	RMSE	RMSE Round
F. característica 1	**0.589806**	**0.538712♦**	**0.769158♣**	**0.815230**
F. característica 2	0.619714	0.574705	0.8161	0.859107
F. característica 3	0.595705	0.554401	0.773438	0.8226
F. característica 4	0.616038	0.577960	0.816798	0.861359

Analizando los valores de la tabla, se puede concluir que la **función característica 1** obtiene los mejores resultados para todas las medidas de error (valores en negrita). Este hecho, unido a que es la que mayor número de usuarios agrupa, señala a la función característica 1 como la mejor de todas las implementadas. Se indica con un diamante el mejor valor en términos de MAE y con un trébol en términos de RSME, que corresponden igualmente a la función característica 1.

También cabe destacar los resultados que obtiene la función característica 3 que, aun habiendo cubierto muchas menos entradas del conjunto de test, comete errores finales bastante cercanos a los que comete la función característica 1. Este resultado sugiere que, al ser junto a la función característica 4 una de las más restrictivas, los grupos de usuarios que genera tienen un mayor nivel de similitud y esto compensa el hecho de dejar mayor cantidad de usuarios aislados. Finalmente se observa que, aunque tras la tercera fase cubre menos entradas del conjunto de test, los resultados de la función característica 2 son muy similares a los de la función característica 4 e incluso comete un error medio absoluto menor con las puntuaciones sin redondear.

Además de los errores calculados con las recomendaciones finales del sistema también se han calculado los errores parciales que se cometerían con las recomendaciones obtenidas en cada fase. De esta forma, se puede analizar cómo de precisas son las recomendaciones generadas en cada una de las fases y, en caso de concluir que el rendimiento obtenido disminuye demasiado en alguna de ellas, intentar buscar formas de mejorar el sistema. Dado que el valor del beneficio mínimo va decreciendo, cabe esperar que las coaliciones formadas tengan cada vez menor similitud y por tanto los errores vayan creciendo. La Figura 18 muestra los errores medios absolutos cometidos por cada función característica tras cada fase del sistema.

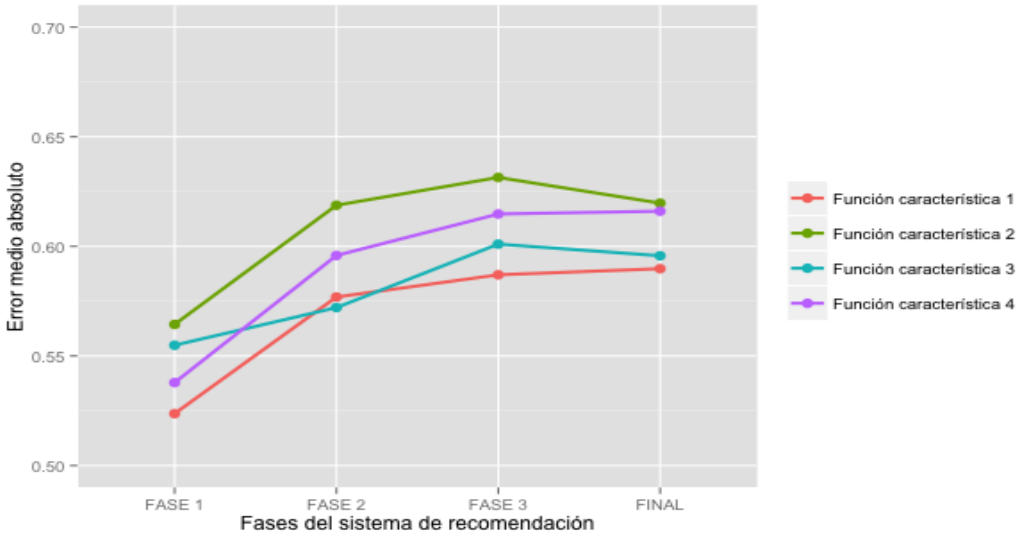

Figura 18.- Evolución del error medio absoluto cometido por las funciones características en cada fase del sistema de recomendación.

La figura valida la hipótesis planteada anteriormente, ya que todas las gráficas son estrictamente crecientes entre la primera y la tercera fase. Es interesante comprobar que, en algunos casos, al utilizar, tras la tercera fase, las puntuaciones medias para completar la recomendación a los pares usuario-entrada del conjunto de test, el error cometido disminuye.

Al analizar el comportamiento de la medida de error en cada fase, se observa que, en las funciones menos restrictivas (1, 2 y 4), es en la segunda fase en la que mayor pérdida de rendimiento se produce. Como parte de un trabajo futuro, quedaría comprobar qué resultados se obtendrían si se eliminase la segunda fase y directamente se juntaran en un solo cluster los usuarios que permanecen aislados tras la primera fase.

Por otro lado, la función característica 3 sufre una pérdida muy sostenida de rendimiento en la segunda fase pero su medida de error crece mucho en la tercera. Una posible explicación de este hecho podría residir en que, al contrario que el resto de funciones, el número de entradas del conjunto de test que cubre la función característica 3 aumenta mucho más en la tercera fase que en la segunda.

La Figura 19 muestra los valores de RMSE correspondientes a cada función en cada fase. Las relaciones entre las gráficas son muy similares a las descritas para la figura anterior y, por tanto, las conclusiones obtenidas son también válidas.

4. 2. 3. COMPARATIVA CON MEDIOS TRADICIONALES

Para completar el análisis se han comparado los resultados obtenidos con un *baseline* formado por los resultados de dos algoritmos de recomendación colaborativa tradicionales: *k*-vecinos más cercanos (*kNN*) y factorización de matrices (*MF*) [4].

[4] Estos resultados han sido proporcionados por el **Dr. Campos** de la Universidad del Bío-Bío (Chile)

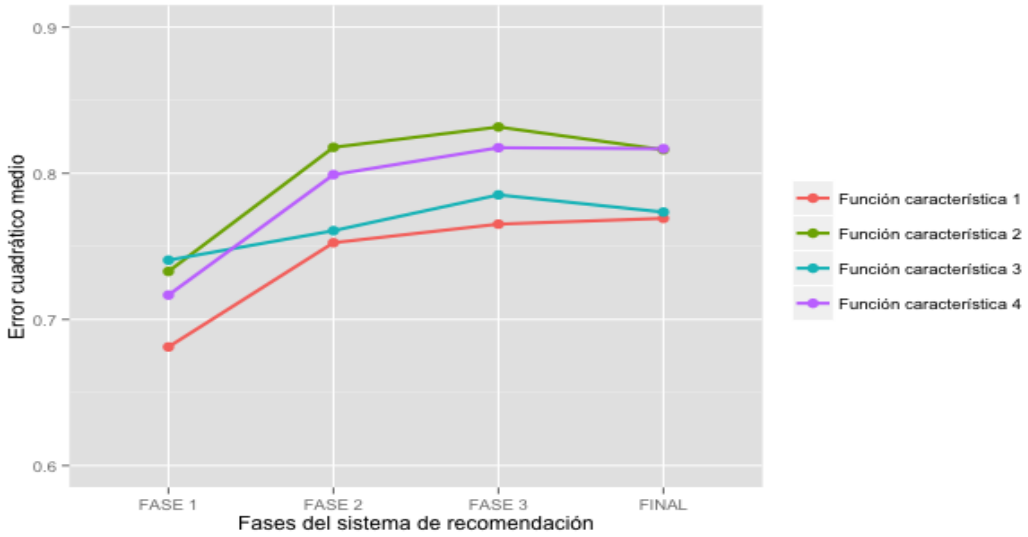

Figura 19.- Evolución del error cuadrático medio cometido por las funciones características en cada fase del sistema de recomendación.

El algoritmo de kNN se basa en buscar los *k* usuarios más cercanos a uno dado utilizando alguna métrica basada en las puntuaciones asignadas a las películas (*e.g.* correlación de Pearson, coseno, etc.). Una vez determinados los usuarios más cercanos se generan recomendaciones para el usuario de partida utilizando las puntuaciones de sus vecinos. En este caso se estableció *k*=30 y se utilizó la correlación de Pearson como métrica.

Por otro lado, la factorización de matrices es una técnica que representa las puntuaciones de los usuarios como matrices y aprovecha las propiedades de éstas para obtener recomendaciones. Para poder llevar a cabo el proceso de recomendación hay que seleccionar qué cantidad de atributos, denominados factores, se utilizarán para llevarlo a cabo. En el algoritmo ejecutado se utilizaron puntuaciones de películas como factores y se estableció en 60 el número de ellas a utilizar.

La Tabla 6 muestra los errores cometidos por estos dos algoritmos.

Tabla 6.- Errores cometidos por los métodos implementados.

	MAE	MAE Round	RMSE	RMSE Round
kNN	0.591596	0.544792	0.743799	0.7957
MF	**0.548206**	**0.494420♦**	**0.691323♣**	**0.746141**
F. característica 1	0.589806	0.538712	0.769158	0.815230
F. característica 2	0.619714	0.574705	0.8161	0.859107
F. característica 3	0.595705	0.554401	0.773438	0.8226
F. característica 4	0.616038	0.577960	0.816798	0.861359

Los resultados obtenidos por el algoritmo de kNN son similares a los obtenidos con la función característica 1. De hecho, dependiendo de la medida de error utilizada los mejores resultados corresponden a uno u otro método. Por otro lado, las recomendaciones obtenidas con el método de factorización de matrices reducen las tasas de error en aproximadamente 0.05 puntos con respecto a la función característica 1.

Adicionalmente a las medidas de error, se ha calculado la distribución de los errores que cometen cada uno de los métodos con sus puntuaciones redondeadas. La Figura 20 contiene la gráfica correspondiente a estos datos.

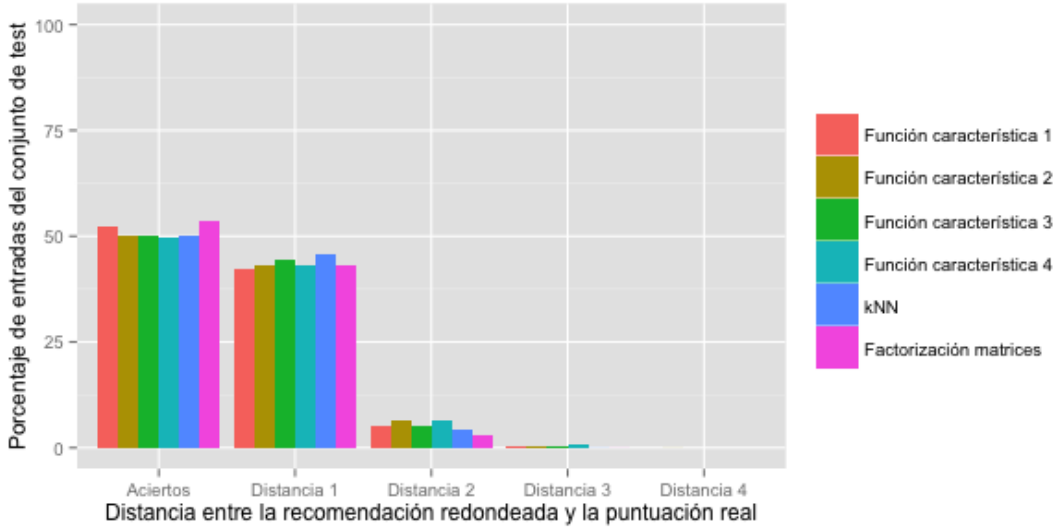

Figura 20.- Distribución de las diferencias entre las puntuaciones redondeadas recomendadas y la puntuación real para todos los métodos implementados.

En la gráfica se observa como la función característica 1 y el método de factorización de matrices son los que más porcentaje de aciertos tendrían a la hora de asignar puntuaciones discretas. Es sorprendente el hecho de que el método de kNN sea el que menos aciertos tendría y aún así haya obtenido medidas de errores similares a los de la función 1. De hecho, la explicación a que cada uno de estos dos métodos obtenga mejores resultados que el otro según la medida de error utilizada puede encontrarse aquí. La función 1 tiene mayor porcentaje de aciertos, pero la suma del porcentaje de aciertos con el de recomendaciones a distancia uno es mayor con el método de kNN (95.68% de kNN por 94.33% de la función característica 1).

De este análisis podemos concluir que los resultados obtenidos con el método propuesto, aunque son aceptables para tratarse de un primer acercamiento, se encuentran por debajo de los métodos tradicionales y, por lo tanto, el método debe refinarse antes de poder ser una opción a considerar a la hora de construir sistemas de recomendación. Dicho refinamiento puede consistir en la automatización de la construcción de funciones características de forma que sea el propio motor del sistema de recomendación quien interprete los resultados y cambie el valor los parámetros para intentar mejor la tasa de aciertos, o, como se propuso anteriormente, la modificación o eliminación de alguna fase.

5. CONCLUSIONES Y TRABAJOS FUTUROS

El Trabajo Fin de Grado realizado ha tenido sendas componentes de investigación y desarrollo en las que se han trabajo algunas de las competencias adquiridas en diferentes asignaturas cursadas a lo largo del doble Grado en Ingeniería Informática y Matemáticas, del que este trabajo es el colofón.

En lo que se refiere a investigación, este trabajo plantea la resolución de un problema ligado a la recomendación de productos a grupos de usuarios. Para su resolución se ha propuesto la utilización de una nueva metodología basada en el uso de la Teoría de Juegos Cooperativos de Utilidad Transferible, combinada con técnicas de aprendizaje automático (creación de *clusters*).

Para poder llevar a la práctica la solución propuesta, ha sido necesario emplear técnicas de muy diversa índole como por ejemplo el acceso a bases de datos, transmisión de la información entre herramientas, programación de algoritmos, desarrollo de funciones en lenguaje R, elaboración de gráficos y tablas para la visualización de resultados, etc. Así mismo, se han llevado a cabo gran cantidad de operaciones de análisis y manipulación de los datos de que disponíamos habiendo realizado procesos de filtrado, combinación o descripción estadística entre otros.

De los resultados obtenidos, y que han sido recogidos en esta memoria, pueden extraerse importantes conclusiones, algunas de las cuales puedes resultar de gran utilidad para futuros trabajos de investigación. Entre ellas destacan las siguientes:

- **Similitudes entre los sistemas de recomendación colaborativos y la teoría de juegos cooperativos.** Al estudiar ambas áreas por separado se encontraron puntos en común entre ellas que han permitido el uso de conceptos y herramientas propias de la teoría de juegos para desarrollar un sistema de recomendación.
- **Teoría de juegos cooperativos, terreno por explorar.** Durante el proceso de estudio de los conceptos básicos de la teoría de juegos, comprobamos que la gran mayoría de resultados disponibles correspondían a juegos superaditivos y se centraban en el problema de reparto. Además, a la hora de buscar tecnologías y lenguajes de programación que dispusieran de librerías especializadas en este campo de las matemáticas, el número de resultados fue bastante corto lo que hace pensar que su uso no está generalizado. Por todo esto, creemos que existe un amplio campo de estudio en el área de la teoría de juegos tanto a nivel teórico como a nivel de desarrollo práctico.
- **Desarrollo de funciones en R.** Al no haber encontrado ninguna tecnología que se adaptase a los requisitos del sistema que se iba a implementar, escogimos R como lenguaje para desarrollar nuestras propias funciones. La potencia de este lenguaje y su orientación al trabajo con conjuntos de datos de gran tamaño han facilitado sobremanera la construcción de una librería que apoyase nuestro trabajo. Por lo tanto, consideramos un gran acierto la decisión de haber utilizado R para el desarrollo de la solución propuesta.

- **La función de similitud como piedra angular.** Tanto en el proceso de documentación como a la hora de diseñar e implementar el modelo se ha comprobado que la clave para obtener buenos resultados es la definición de una función de similitud que se adapte lo mejor posible a la realidad. Y es precisamente para este proceso para el que no existe ningún algoritmo universal. Valga como ejemplo el caso tratado que, tras construir quince funciones distintas, la función que mejores resultados ha obtenido es la que primero se generó. Los coeficientes utilizados en esta función se determinaron aplicando cierta lógica pero sin tener ningún resultado a priori en el que justificarse, más allá del análisis previo del conjunto de datos.
- **Resultados aceptables pero mejorables.** Las tasas de error obtenidas por los sistemas de recomendación implementados están cerca de las usadas como *baseline,* correspondientes a técnicas de recomendación tradicionales. Además, la implementación práctica del método propuesto ha tenido que adaptarse a las limitaciones propias de un primer acercamiento, por lo que cabe pensar que los resultados podrían mejorar con un mayor número de recursos.

5. 1. TRABAJOS FUTUROS

Los resultados de este trabajo han planteado nuevas incógnitas y posibles líneas de trabajo futuras que quedan resumidas a continuación:

- **Búsqueda de nuevos puntos en común.** El trabajo llevado a cabo debe servir de punto de partida para seguir explorando formas de aplicar las técnicas propias de la teoría de juegos, incluyendo el análisis de los resultados para juegos no cooperativos, para desarrollar herramientas relacionadas no solo con los sistemas de recomendación sino también con el resto de técnicas de manipulación de datos (clasificación, detección de comunidades, optimización, etc.).
- **Refinamiento del modelo propuesto.** El análisis de los resultados obtenidos ha sugerido algunos cambios en la solución propuesta que podrían estudiarse como, por ejemplo, la modificación o sustitución de alguna de las fases.
- **Ampliación de la librería en lenguaje R.** Para la realización de este estudio se desarrollaron una serie de funciones necesarias para la implementación del método propuesto. Sin embargo, el refinamiento del modelo junto con el descubrimiento de nuevos aspectos en común, deben desembocar en la ampliación y la generalización de las funcionalidades provistas por la librería. De hecho, si se generase un paquete de funciones completo podría plantearse su integración dentro de las librerías descargables desde los entornos de desarrollo.
- **Implementación del modelo general.** Debido a las restricciones computacionales del proyecto ha sido necesaria la implementación de un sistema de recomendación *ad hoc* incluyendo procesos auxiliares para reducir las operaciones realizadas por el sistema principal. Sería interesante comprobar qué resultados se obtendrían trabajando con el conjunto de datos completo y sin llevar a cabo el proceso de clustering previo.

- **Generación de funciones características automáticamente.** En la misma línea del punto anterior, uno de los aspectos en los que deberían centrarse trabajos futuros es en la automatización del proceso de generación de funciones características. Para ello, podrían utilizarse métodos de aprendizaje automático que permitiesen al propio sistema interpretar los resultados obtenidos y modificar los parámetros de las funciones en consonancia. Hay que tener en cuenta que para poder llevar a cabo esto, es necesaria una cantidad de recursos bastante considerable.
- **Empleo de datos sociales:** en el trabajo se ha prescindido del uso de datos sociales presentes en los perfiles de los usuarios del dataset empleado. El uso de dichos datos (relaciones de amistad, comentarios, valoraciones de los comentarios, etc.) pueden aportar información importante al proceso de recomendación. Además, en el concurso organizado por la ACM se proporcionaban tres conjuntos de datos adicionales que contienen mayor cantidad de datos sociales. Por lo tanto sería importante abrir una línea de investigación en esta dirección.

BIBLIOGRAFÍA

Blashfield, R. K. (1976). Mixture model tests of cluster analysis: Accuracy of four agglomerative. *The Psychological Bulletin* (83), 377–388.

Burke, R. (2007). Hybrid Web Recommender Systems. *The Adaptative Web*, 377-408.

Community, R. (n.d.). *The ACM Conference Series on RecommenderSystems*. Retrieved from http://recsys.acm.org/

Cuadras, C. M. (1986). *Métodos de Análisis Multivariante.* Editorial Universitaria de Barcelona.

Gillies, D. B. (1959). Solutions to general non-zero-sum games. *Contributions to the Theory of Games IV. (Annals of Mathematics Studies 40).*, 47-85.

Golbeck, J. (2006). Generating predictive movie recommendations from trust in social networks. *Trust Management, 4th International Conference iTrust*, (pp. 93–104). Pisa, Italia.

Hamming, R. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal, XXIX*(2).

Hands, S., & Everitt, B. (1987). A Monte Carlo study of the recovery of cluster structure. *Multivariate Behavioral Research*(22), 235-243.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*(22), 5-53.

Kogan, J. (2007). *Introduction to Clustering Large and High-Dimensional Data.* Cambridge University Press.

Kuiper, F., & Fisher, L. (1975). A Montercarlo comparison of six clustering procedures. *Biometrics*(31), 777-783.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 707-710.

Littlechild, S., & Thompson, G. (1977). Aircraft Landing Fees: A Game Theory Approach. *The Bell Journal of Economics, 8*(1), 186-204.

Mirás Calvo, M. A., & Sánchez Rodríguez, E. (2009, Abril). *Transferable utility game theory Matlab toolboxes*. Retrieved from http://mmiras.webs.uvigo.es/TUGlab/

Owen, G. (1968). *Game Theory* (Segunda edición ed.). W.B. Saunders Company.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (Eds.). (2011). *Recommender Systems Handbook.* Springer.

Rich, E. (1979). User Modeling via Stereotypes. *Cognitive Science*, 329-354.

Shapley, L. S. (1953). A Value for n-person Games. (H. Kuhn, & A. Tucker, Eds.) *Annals of Mathematical Studies, II*, 307-317.

Torkaman, A., Moghaddam, N., Aghaeipour, M., & Hajati, E. (2009). A Recommender System for Detection of Leukemia. *17th Mediterranean Conference on Control & Automation.* Makedonia Palace, Thessaloniki, Greece.

Von Neumann, J., & Morgenstern, O. (1944). *Theory of Games and Economic Behavior.* Estados Unidos: Princeton University Press.

Wang, Y., & Tseng, M. M. (2011). Adaptive attribute selection for configurator design. (C. U. Press, Ed.) *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 185–195.

Ward, J. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association, 58*, 236-244.

GLOSARIO

ACM: Association for Computing Machinery

Arranque en frío: problema al que deben hacer frente los sistemas de recomendación cuando aparece un nuevo usuario o un nuevo producto. Al no tener información sobre él deben buscarse métodos alternativos para generar recomendaciones.

Beneficio Mínimo: valor mínimo de similitud entre participantes que debe alcanzarse en cada iteración del modelo propuesto para que se forme una coalición. Sirve como condición de parada para el algoritmo.

CAMRa: Challenge on Context-Aware Movie Recommendation.

Coalición: subconjunto del conjunto de jugadores participantes en un juego que deciden cooperar para obtener beneficios comunes.

Contribución marginal: dado un juego cooperativo (N, v), y suponiendo la coalición total, la contribución marginal del jugador i al beneficio total $v(N)$ viene determinado por $v(N) - v(N\backslash\{i\})$.

Dendrograma: tipo de diagrama en forma de árbol que muestra las relaciones de agrupación o división entre los datos de partida. Es la forma más visual de mostrar la evolución de algoritmos de clustering jerárquicos.

Función característica: expresión matemática que define la estructura de un juego cooperativo asignando un valor de beneficio a cada posible coalición.

kNN: algoritmo de k vecinos más cercanos.

Juego Cooperativo de Utilidad Transferible: juego cooperativo en el que los beneficios obtenidos pueden ser repartidos entre los jugadores sin ningún tipo de restricción.

MAE: error medio absoluto.

MF: algoritmo de factorización de matrices.

Participante: usuario o coalición que permanece activo en el sistema de recomendación.

Problema de Reparto: problema propio de la teoría de juegos cooperativos consistente en determinar la forma más justa de distribuir los beneficios obtenidos por cada coalición entre los jugadores que la forman.

RMSE: error cuadrático medio.