ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
CENTRO DE INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA

Recomendaciones con Filtrado Colaborativo Basado en Usuario y en Ítem Aplicando el Paradigma Map-Reduce

Mervyn Macias[1] Freddy De la Rosa[2] Cristina Abad[3]
Facultad de Ingeniería Eléctrica y Computación
Escuela Superior Politécnica del Litoral
Campus "Gustavo Galindo V." Km. 30.5, Vía Perimetral, Apartado 09-01-5863, Guayaquil, Ecuador
www.espol.edu.ec
merv_mm@hotmail.com[1], freddyfdelarosa@gmail.com[2], cristina.abad@gmail.com[2]

Resumen

Un sistema de recomendaciones es un tipo específico de filtro de información que ayuda al usuario a seleccionar ítems de su interés tales como películas, músicas, páginas Web, revistas, libros, etc. Actualmente, los sitios Web que prestan estos servicios requieren que la gran cantidad de información recibida por todas las acciones implícitas o explícitas de millones de usuarios sobre millones de ítems, sea procesada de una manera rápida y con la menor infraestructura posible, esto con el fin de obtener rápidos y mejores índices de preferencias útiles y a menor costo El presente trabajo tiene como objetivo presentar dos alternativas de procesamiento de recomendaciones de ítems musicales basados en las preferencias implícitas de los usuarios y utilizando un modelo de programación masiva y escalable dentro del framework de Hadoop como un sistema de la ejecución de tareas en paralelo y tolerante a fallos.

Palabras claves*: Filtrado colaborativo, Sistema de archivos distribuidos Hadoop (HDFS), Mahout, coeficiente correlación de Pearson.*

Abstract

A system of recommendations is a specific type of filter of information that helps the user to select such articles of his (her, your) interest as movies, musical, web pages, magazines, books, etc. Nowadays, the web sites that give these services need that the great quantity of information got for all the implicit or explicit actions of million users on million articles, is tried in a rapid way and with the minor possible infrastructure, this in order to obtain rapid and better indexes of useful preferences and to minor cost The Present work has as aim to present two alternatives of processing recommendation of musical articles based on the implicit preferences of the users and using a model of massive and scalable programming inside Hadoop's framework as a system of the execution of tasks in parallel and tolerantly to failures.

Keywords*: Colaborative filtering, Hadoop Distributed File System (HDFS), Mahout, coefficient Pearson's correlation.***

1. Introducción

Antes de la implementación de los sistemas de recomendación, eran insuficientes los recursos a los que el usuario podía acceder para obtener sugerencias sobre temas de su interés. Por ejemplo, en el caso de recomendaciones musicales, la publicidad en radio era uno de los pocos medios a los cuales el usuario tenía acceso para conocer nuevos artistas y canciones. Ahora la situación se ha invertido. Se cuenta con extensos datasets en continuo crecimiento, que dificultan la clasificación de la información.

Aunque en el ámbito más general existen 5 paradigmas de recomendación [1], nos enfocaremos en dos de ellos: recomendación basada en contenido y recomendación colaborativa. En el primer caso las recomendaciones son hechas en función de los ítems que el usuario ha elegido en el pasado y, en el segundo, se identifica usuarios con características similares, y se recomiendan ítems que el usuario no haya elegido pero que sean del agrado de los usuarios similares. Un sistema que aplica recomendación colaborativa no se preocupa del contenido, es más, lo único que se conoce acerca de cada ítem es un identificador. Actualmente están emergiendo sistemas que aplican los dos paradigmas, por lo que reciben el nombre de sistemas híbridos, son ejemplos Fab [2] y Select [3].

Los sistemas de recomendación colaborativa juegan un rol importante en la actualidad. Con esta aproximación la comunidad de usuarios evalúa cada elemento y determina de forma implícita qué ítems son recomendables a qué usuario. Por otro lado, la implementación de estos sistemas aplicando paradigmas tradicionales de programación, para datasets extensos, no es la mejor opción debido a los tiempos de ejecución inherentes a los algoritmos correspondientes. En el presente trabajo mostraremos alternativas de implementación de algoritmos de recomendación colaborativa aplicando el paradigma Map-Reduce.

2. Recomendación Colaborativa

Se define la recomendación colaborativa como aquella en la cual las recomendaciones son hechas exclusivamente en base a usuarios con gustos similares. Utiliza tanto la base de ítems como la base de usuarios para generar predicciones. Primero emplea producto punto o correlación para encontrar usuarios con características similares, a quienes se denomina vecinos cercanos. Luego combina sus preferencias para generar una lista con los elementos más recomendables para el usuario.

Los algoritmos que se detallan más adelante requieren la puntuación que cada usuario ha asignado cada ítem. El sistema puede obtener las puntuaciones de forma explícita o implícita [4]. En el primer caso, el usuario califica los ítems que ha utilizado. En el segundo, el sistema obtiene retroalimentación implícita capturando la interacción del usuario sin que él lo note. Por ejemplo, en el caso de un sistema de recomendaciones musicales, si un usuario escucha con más frecuencia al artista A que al B, es sistema asigna mayor puntuación al artista A.

La siguiente tabla muestra una pequeña base de datos con 4 usuarios, 5 artistas y el número de veces que cada usuario ha escuchado a cada artista. Por ejemplo, el usuario 3 ha escuchado 5 veces al artista A.

	Ítems				
Usuario	**A**	**B**	**C**	**D**	**E**
1	4	5	0	0	0
2	5	6	0	5	0
3	5	0	3	0	8
4	0	0	5	0	0

Tabla I. Una base que muestra el número de veces que cada usuario ha escuchado cada ítem

El hecho que haya más similitudes entre los usuarios 1 y 2 que entre los usuarios 1 y el 3 indica que el usuario 1 es más similar al usuario 2 que al usuario 3. El sistema debe notar esto y recomendar al usuario 1 que escuche el ítem D, puesto que aún no lo ha visto y fue del agrado de un usuario con gustos similares a los de él.

Sólo se pueden recomendar artistas que el usuario no haya escuchado. Por ejemplo, el usuario 1 ha escuchado a los artistas A y B, por lo tanto sólo son posibles recomendaciones los artistas C, D y E.

3. Algoritmos

Los algoritmos que se muestran a continuación se basan en el producto punto entre dos vectores y en las fórmulas de correlación [5]. El grado de cercanía entre dos vectores lo podemos obtener calculando el coseno del ángulo que forman.

3.1 Basado en Usuario

En un sistema con m usuarios y n recomendaciones, se define el vector $\vec{u_i}$ como aquel cuyas coordenadas corresponden al número de veces que el usuario i escuchó cada canción. Por ejemplo, para la tabla I, el vector $\vec{u_1}$ está dado por la expresión <4, 5, 0, 0, 0>.

El sistema debe calcular la similitud entre cada par de usuarios para generar una matriz de similitud. La

similitud entre el usuario $\vec{u_i}$ y el usuario $\vec{u_j}$ viene dada por la fórmula:

$$Sim\left(\vec{u_i},\vec{u_j}\right)=\frac{u_i \bullet u_j}{|u_i||u_j|}$$

Para determinar qué usuarios son similares a un usuario particular se debe seleccionar un umbral. Los usuarios obtenidos se denominan vecinos cercanos.

Otra opción para determinar la similitud entre dos usuarios es aplicar correlación de Pearson [6]. El coeficiente de correlación de Pearson mide la relación lineal entre dos variables cuantitativas. Si u y v son dos usuarios, $r_{u,i}$ es la valoración del usuario u al ítem i, $\overline{r_u}$ es la valoración media del usuario u y σ_u es la desviación estándar de las valoraciones, la similitud entre u y v se obtiene a través de la expresión:

$$sim(u,v)=\frac{\sum_{i=1}^{m}\left(r_{a,i}-\overline{r_a}\right)\left(r_{u,i}-\overline{r_u}\right)}{\sigma_a \sigma_u}$$

Los ítems que no han sido escuchados por el usuario pero sí por sus vecinos cercanos son potenciales recomendaciones. Para cada uno de esos ítems se usa la siguiente expresión:

$$p_{a,i}=\frac{\sum_{k=1}^{n} r_{u_k,i} \bullet Sim\left(\vec{a},\vec{u_k}\right)}{\sum_{k=1}^{n} Sim\left(\vec{a},\vec{u_k}\right)}$$

Donde $P_{a,i}$ es el grado de recomendación del ítem i al usuario a y $r_{u_k,i}$ la valoración del usuario u_k al ítem i.

La fórmula anterior es bastante intuitiva. Cada sumando es el producto entre la similitud del usuario con un vecino cercano y el número de veces que ese vecino escucho el ítem i.

3.2 Basado en Ítems

El principio es el mismo, la diferencia es que en este caso buscamos similitudes entre ítems en lugar de buscar similitudes entre usuarios. Definimos el vector $\vec{a_i}$ como aquel cuyas coordenadas son el número de veces que cada usuario lo ha escucho. Por ejemplo, para la tabla 2.1, el vector $\vec{a_1}$ está dado por <4, 5, 5, 0>. La fórmula es análoga a la utilizada para calcular similitudes entre usuarios.

3.3 Aproximación al Algoritmo Basado en Usuario

La aplicación de la expresión de similitud implica varias multiplicaciones y divisiones para cada comparación. Una aproximación a los métodos anteriores es aplicar directamente el producto punto para establecer el grado de similitud entre 2 usuarios. La ventaja es una reducción considerable de los tiempos de ejecución a cambio de la reducción en la calidad de las recomendaciones.

4. Esquema Map-Reduce

Map-reduce es un modelo de programación para el procesamiento de grandes cantidades de información en donde una función llamada "map" procesa pares clave/valor para generar un conjunto de pares intermedios clave/valor, y una función "reduce" que agrupa todos los valores intermedios asociados con la misma clave [7].

Para este trabajo se implementan dos variantes para el cálculo de vecinos cercanos: correlación de Pearson y producto punto.

A continuación se muestra el esquema map reduce para la aplicación del producto punto como medida de similitud entre usuarios [8].

La figura 4.1 muestra como se procesan los datos de entrada durante la ejecución del primer proceso. El map genera como salida los pares (usuario/[ítem valoración]). El reduce agrupa los pares con la misma clave genera

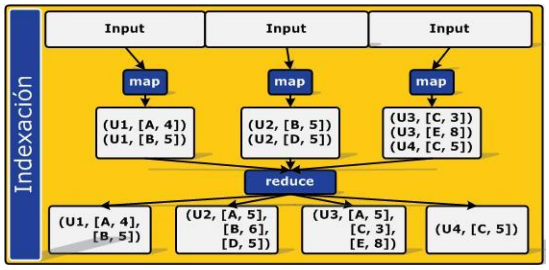

Figura 4.1: Esquema ***map-reduce*** que muestra el primer proceso map-reduce, indexación.

El segundo proceso toma como entradas las salidas del primero. Cada usuario constituye un vector. Las similitudes se hayan aplicando producto punto. La salida es una matriz que contiene las similitudes entre cada par de usuarios del sistema. El producto punto es conmutativo, así la similitud $Sim\left(\vec{u},\vec{v}\right)$ es igual a $Sim\left(\vec{v},\vec{u}\right)$. El map aprovecha esto para mejorar la eficiencia. En la figura 4.2 se muestra el esquema.

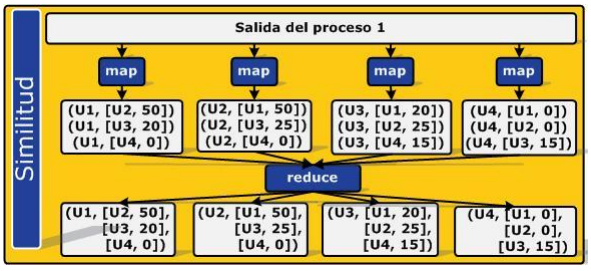

Figura 4.2: Esquema ***map-reduce*** que muestra el segundo proceso map-reduce, cálculo de similitudes.

El tercer proceso utiliza la matriz de similitudes y la valoración de los usuarios. Los artistas que se pueden recomendar son aquellos que no ha escuchado el usuario. Ver figura 4.3.

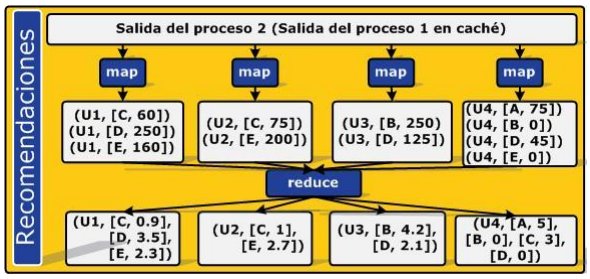

Figura 4.3: Esquema ***map-reduce*** que muestra el tercer proceso map-reduce, recomendaciones.

5. Análisis de Algoritmos

Con respecto a la tabla I, al aplicar el algoritmo de recomendación basado en usuario, cada usuario se compara con el resto, así, para cada usuario se realizan 3 comparaciones. Por lo tanto es necesario un total de 12 comparaciones. En el peor de los casos, cada comparación implica 5 operaciones, por consiguiente el tiempo de ejecución es del orden 12*5=60.

En general, para un dataset con *n* usuarios y *m* ítems, para cada usuario se deben realizar *n(n-1)* comparaciones. En el peor de los casos cada comparación implica *m* operaciones. Así, el tiempo de ejecución es del orden de mn^2.

El dataset utilizado en este proyecto tiene aproximadamente 150000 usuarios y 2000000 de ítems. El tiempo de ejecución es del orden de 10^{16}.

Si aplicamos algoritmo basado en ítem, las comparaciones se realizan entre ítems, en cuyo caso el tiempo de ejecución es del orden de nm^2, 10^{17}, 10 veces mayor al algoritmo basado en usuarios.

Si el dataset tiene más usuarios que ítems lo ideal es aplicar el algoritmo basado en ítems. En caso contrario, la mejor decisión depende de qué tan grande sea la diferencia de cantidad de usuarios y de ítems.

6. Implementación con Mahout

Mahout es un proyecto de Apache Lucene cuyo propósito es construir librerías escalables de máquinas de aprendizaje. Mahout implementa el algoritmo basado en ítems y el algoritmo basado en contenido [9].

Para la implementación de los algoritmos utilizaremos el dataset de Audioscrobbler. Audioscrobbler es un dataset de música que entre otros datos contiene una base de usuarios, de artistas y el número de veces que cada usuario ha escuchado a cada artista.

Para trabajar con los algoritmos de Mahout se necesita una gran cantidad de memoria RAM. El modelo de datos se implementa mediante la interfaz "DataModel". Mahout crea un objeto usuario por cada usuario del sistema y recomienda un dataset con máximo de diez mil usuarios. Nos vemos en la necesidad de manipular el código fuente para utilizar el dataset completo.

En la figura 6.1 explica el proceso de implementación del sistema de recomendaciones utilizando el paradigma map-reduce y la librería Mahout para la obtención de clave/valor para las comparaciones, el framework de Hadoop [10] para el procesamiento síncrono y programado de tareas, y las herramientas de Amazon EC2 que proveen el ambiente virtual necesario para un computo masivo y escalable de datos en la nube.

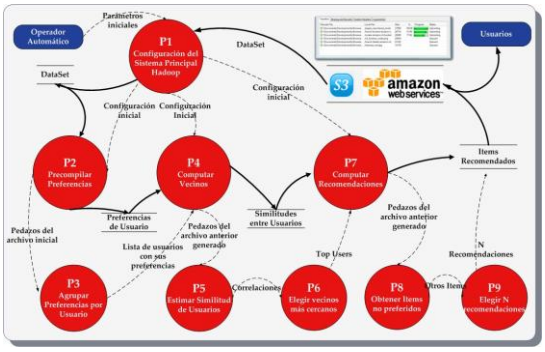

Figura 6.1: Diagrama de flujo de datos del sistema en general.

En este diagrama se representa a las entidades (Operador, Usuario), los procesos (P1, P2, ...etc) y los almacenes (S3, HDFS).

El sistema inicia con el ingreso de los parámetros tales como el archivo de entrada, el archivo de salida, los N vecinos más cercanos y las N recomendaciones para cada usuario.

Luego pasa al proceso de pre-indexación de usuarios el cual genera un archivo de texto plano donde cada línea representa las preferencias por usuario. Este archivo será almacenado en el HDFS el cual sirve como entrada para el siguiente proceso que tiene como objetivo el cómputo de los N vecinos más cercanos. En la salida del cómputo mencionado se graban los N usuarios más similares con su valor de similitud por cada usuario activo.

Por último, el proceso de cómputo de las recomendaciones recibe como entrada el archivo de las similitudes para obtener los ítems nunca antes vistos por cada usuario y generar las N recomendaciones por medio de colas de prioridad, el

cual dará como resultado un archivo en texto plano que luego será comprimido y subido al S3 para el almacenamiento respectivo.

7. Análisis de Resultados

Nuestro primer inconveniente radicaba en la gran cantidad de memoria que utiliza la librería de Mahout para el almacenamiento de las características de todos los usuarios y en el tiempo de respuesta que llevaría la comparación de todos los ítems para todos los 150.000 usuarios del dataset en una máquina local.

Para los resultados del presente trabajo nos enfocamos en el tiempo de respuesta y costo de cada proceso ejecutado en EC2. Se muestra la comparación del tiempo de ejecución del proceso de los vecinos más cercanos utilizando dos variantes en la obtención de los valores de similitud entre usuarios.

Para la ejecución de cada una de las dos variantes se levantaron 10 nodos con la misma configuración en la plataforma Linux y Hadoop versión 0.18.3-14. La figura 7.1 muestra los procesos ejecutados para las dos variaciones.

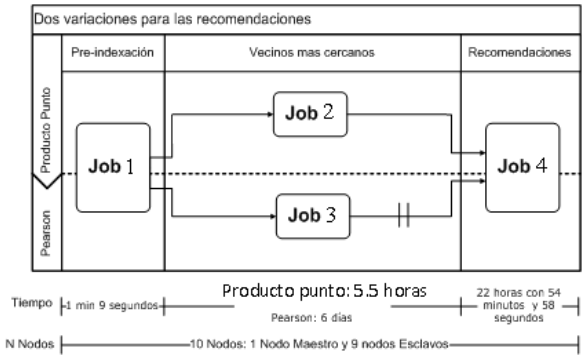

Figura 7.1. Diagrama de flujo de ejecución de los Jobs en 10 nodos.

Para el job 1, el cual es aplicado en las dos variaciones, se levantaron 4 tareas map y 10 tareas ***reduce***, las cuales presentaron una duración total 1 minuto con 9 segundos.

Para el job 4, el cual es aplicado también para las dos variaciones, se levantaron 10 tareas map y 10 tareas ***reduce***, los cuales presentaron una duración total de ejecución de aproximadamente 22 horas con 54 minutos y 58 segundos.

Para el tiempo de respuesta en la obtención de los vecinos más cercanos se tuvieron dos alternativas expuestas a continuación:

Con el coeficiente de correlación de Pearson se tuvo un tiempo de respuesta de aproximadamente 144 horas (6 días). Esta duración de tiempo fue estimada y obtenida de los tiempos de respuestas de los primeros porcentajes de avance en las tareas map. Por tanto el proceso fue terminado de forma manual ya que la ejecución de todo el proceso habría implicado tener levantados los 10 nodos durante aproximadamente 6 días, lo cual habría causado un exceso en el presupuesto.

Con el producto punto el problema anterior se tomó en consideración un procesamiento de los valores de similitud utilizando el agrupamiento de tuplas de usuarios similares y multiplicando las preferencias de sus ítems (Producto punto). El tiempo de respuesta con respecto a la alternativa anterior se reduce en un 99.93% (ver figura 7.2).

Este método no asegura que todos los usuarios tengan la vecindad requerida ni tampoco que se contemplen todos los ítems del data set porque en el proceso de agrupamiento de tuplas existen ítems que han sido escuchados por un sólo usuario [11]. Como consecuencia, este usuario no debería participar como candidato a ser vecino.

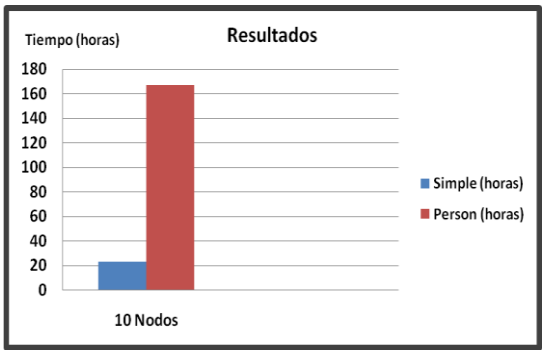

Figura 7.2. Gráfico de barras mostrando el tiempo de respuesta de los dos procesos de recomendaciones utilizando dos tipos de correlación.

8. Conclusiones y Recomendaciones

Los tiempos de ejecución de los algoritmos de recomendación colaborativa basados en correlación y en producto punto aumentan con el cuadrado de la cantidad de usuarios y por ello presentan problemas de escalabilidad. En el caso del algoritmo la correlación de Pearson, el costo computacional es tan alto que en términos monetarios supera 8 a 1 al algoritmo basado en producto punto. La poca cantidad de nodos compartidos disponibles es una limitante. La aplicación de producto punto es una alternativa viable para el procesamiento de las similitudes. Aunque con el producto punto disminuye en gran proporción el tiempo de ejecución, no se obtienen recomendaciones para usuarios que han calificado ítems que otros usuarios no han calificado, Pearson resuelve este problema aplicando inferencias. Una mejora puede ser la aplicación de una variante de producto punto que utiliza inferencias.

9. Referencias

[1]. Aniceto Saboya Vargas Estudiante de doctorado en Inteligencia Artificial, "Uso de Recomendadores, Asistentes y Ayudantes en sistemas Tutores," *Universidad Politécnica de Cataluña*, pág. 6.

[2]. Marko Balabanović, "Content-based, collaborative recommendation," ACM New York, NY, USA, vol. 40, Mar. 1997, pág. 72.

[3]. David M. Nichols, "Implicit Rating and Filtering," Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering, Nov. 1997, pág. 5.

[4]. Roland Alton-Scheidl, Jesper Ekhall, Olivier van Geloven, Laszlo Kovacs & Andras Micsik, Christopher Lue, Richard Messnarz, David Nichols, Jacob Palme & Torgny Tholerus, Dave Mason & Rob Procter, Enrico Stupazzini & Massimo Vassali, y Richard Wheeler, "Social and Collaborative Filtering of Web Documents and News," Dagstuhl, Germany: Alfred Kobsa, GMD and Constantine Stephanidis, 1999, pág. 14.

[5]. Oswaldo Velez-Langs y Carlos Santos, "Sistemas Recomendadores: Un enfoque desde los algoritmos genéticos," *Instituto de Investigación de Ingeniería Industrial de la UNMSM. (Peru)*, vol. 9, Jun. 2006, pág. 31.

[6]. Sergio Manuel Galán Nieto, *Filtrado Colaborativo y Sistemas de Recomendación*, España - Madrid: Universidad Carlos II de Madrid, 2007.

[7]. Jeffrey Dean and Sanjay Ghemawat. Google, Inc., "Google Research Publication: MapReduce," Research Publications Google, Dic. 2004, pág. 13.

[8]. Elsayed, Douglas W. Oard, y Jimmy Lin Tamer, Pairwise Document Similarity in Large Collections with MapReduce, University of Maryland: Human Language Technology Center of Excellence and UMIACS Laboratory for Computational Linguistics and Information Processing, 2008.

[9]. Andre Vellino, "PageRank Effect on Collaborative Filtering," *National Research Council (Ottawa, Ontario K1A 0R6)*, Jul. 2008, pág. 4.

[10]. "The Hadoop Distributed File System: Architecture and Design," *The Hadoop Distributed File System: Architecture and Design*, May. 2008.

[11]. Valeria Molina, Rodrigo Machado, y Leticia Betarte, "Recomendación de música basada en filtrado colaborativo," Grado, UdelaR - Carrera de Ingeniería en Computación.