Universidad de Valparaíso
CHILE
Instituto de Estadística
Facultad de Ciencias

Sistema de Recomendaciones para Netflix usando Técnicas de Minería de Datos.

Trabajo de titulación, para optar al título profesional de: Ingeniero Estadístico

Tomás Tapia Barría

Profesor Guía: Dr. Harvey Rosas Quintero.

Enero 2018

Agradecimientos

Agradecimientos a mi madre, hermanos y amigos, quienes fueron indispensables en este proceso.

También agradezco al profesor Harvey por su compromiso con este proyecto, su dedicación en las clases, por incentivarme a ser autodidacta y entender mis motivaciones.

Mis más sinceros agradecimientos a todos los académicos y funcionarios del Instituto de Estadística de la Facultad de Ciencias. Agradezco la posibilidad de haber formado parte de esta casa de estudios en esta bella ciudad llamada Valparaíso.

Resumen

Netflix es una empresa estadounidense dedicada al entretenimiento que proporciona mediante tarifa mensual, asociada a una cuenta de usuario, servicios de transmisión por Internet de películas y series de televisión. La empresa entre los años 2006 y 2009, formuló una competencia de análisis de datos en donde se dispuso un conjunto de archivos de prueba de 2,1 GB. El concurso consistía en premiar con un millón de dolares a aquellos competidores que pudiesen mejorar el sistema de recomendaciones de ese entonces, llamado Cinematch.

En este estudio se buscó realizar una propuesta para el algoritmo de recomendaciones de películas y series de las cuentas de los usuarios. Se incluyó información relevante sobre temas tales como: *Big Data*, Hadoop, Mapreduce y su implementación junto con el software R, así como también el uso de un modelo de sistemas de recomendación. Este modelo permitió realizar predicciones de las calificaciones a los ítems que puedan ser de interés para los distintos usuarios suscritos. Se utilizó uno de los algoritmos probados por el equipo BellKor's Pragmatic Chaos, donde se establecen previamente características y conceptos asociados a este.

Índice general

Índice de Imágenes y figuras

Índice de Tablas

Abreviaturas y simbología

A continuación son presentadas algunas abreviaturas y simbologías utilizadas en este estudio:

CPU: Central Processing Unit .

GB: Gigabyte.

HDFS: Hadoop Distributed File System .

IDEUV: Instituto de Estadística de la Universidad de Valparaíso.

RHIPE: R and Hadoop Integrated Programming Environment.

RMSE: Root Mean Squared Error.

SSH: Secure Shell.

Capítulo 1

Introducción

En los últimos años grandes compañías han aprovechado la oportunidad de crecer, gracias a las distintas técnicas usadas para aumentar las ventas en sus establecimientos o sitios web, ya sea impulsando soluciones de marketing o utilizando el análisis estadístico de datos. Las empresas, se esfuerzan por obtener y almacenar grandes cantidades de datos de las distintas operaciones que pudiesen realizar, analizando e identificando a la vez, patrones de comportamiento que permitan predecir las acciones de sus clientes y de la misma forma satisfacer las distintas necesidades. Con esta idea nace el concepto de sistemas de recomendaciones.

Los sistemas de recomendaciones, son métodos que buscan el poder auto-matizar la consulta de un usuario de acuerdo a la opinión qué otros puedan tener de un producto o servicio, y con estas opiniones poder tomar las decisiones para la adquisición de estos. Estas decisiones pueden estar ligadas a distintas acciones de la vida cotidiana, ya sea qué película elegir para ver antes de dormir, gustos de música en una familia o qué alimentos pudiesen ser del agrado de una persona qué frecuenta una tienda de comestibles de la zona (Jannach , Zanker, Felferning & Friedrich, 2010).

Las empresas en el ultimo tiempo, han considerado el análisis de datos para satisfacer las necesidades de sus usuarios, con esto han visto como

prioridades, el incorporar a sus equipos profesionales expertos, que tengan las competencias para realizar estos análisis. De la misma forma, cuando aún parece necesario avanzar en los estudios relacionados a la posibilidad de mejorar los algoritmos que son útiles en el filtrado de información, se han incorporado a las comunidades científicas, mediante competencias que abordan un desafío motivado por un premio monetario, premiando a aquellos que puedan superar alguna meta establecida. Motivado por lo anterior surgen competencias como el *Netflix Prize*.

La metodología expuesta en este análisis aborda la aplicación de un mode-lo de sistemas de recomendaciones, utilizando el *cluster* del IDEUV y la integración de R con Hadoop. Esta arquitectura permite abordar el desafío del *Netflix Prize*, dado que sin estos últimos recursos mencionados, sería imposible realizar dicho análisis dadas las dimensiones del conjunto de datos.

En este trabajo es empleado un algoritmo de sistemas de recomendaciones basado en el documento de Koren (2009), utilizando datos pertenecientes al desafío *Netflix Prize*. Se planteó qué el algoritmo permite predecir la calificación de algún usuario a una película, superando el registro del *RMSE* de Cinematch.

El objetivo de esta investigación considera el uso de un modelo para un sistema de recomendaciones, de las películas y series de los usuarios de la página de *Netflix*, estudiando conceptos asociados a sistemas de recomendaciones junto con el uso de una arquitectura de bajo costo para el procesamiento de datos de R con Hadoop y las respectivas librerías disponibles.

En el primer capítulo se aborda el estado del arte recabando literatura reciente relacionada al uso de sistemas de recomendaciones; en el segundo capítulo se aborda la fundamentación teórica; abordando el reglamento de la competencia del *Netflix Prize*; definiciones de algunos conceptos asociados a la implementación y utilización de la arquitectura de Hadoop y el software R, así como su integración. También son tratados conceptos ligados a la utilización de un modelo de sistemas de recomendaciones útil en este

estudio. En el tercer capítulo se presentan la metodología de la aplicación de sistemas de recomendación, así como también la sintaxis necesaria para el procesamiento con R y incorporando la arquitectura del *cluster* del IDEUV, en este capítulo también son presentados los resultados de los experimentos planteados y definidos en la fundamentación teórica. Finalmente se abordan las conclusiones y trabajos futuros relacionadas de esta investigación, en que posteriormente se detalla la bibliografía utilizada.

Capítulo 2

Estado del Arte

En los últimos años el procesamiento y análisis de datos ha tomado una importancia relevante en el mundo, ayudando a comprender la naturaleza de la información en las distintas áreas del conocimiento.

El presente estudio es un aporte en el tratamiento de datos en sistemas de recomendaciones, concepto que aún representa una novedad, pero de gran utilidad en la actualidad.

Los sistemas de recomendaciones son métodos que permiten establecer filtros para los usuarios de un determinado producto o servicio, mostrando para estos solo aquellos ítems que sean de su interés. En la actualidad estas técnicas han sido ampliamente utilizadas por empresas, que se han especializado en la venta de productos y servicios por Internet.

Es importante resaltar que la aplicación de técnicas estadísticas en el manejo de sistemas de recomendaciones, requiere de herramientas matemáticas, estadísticas y computacionales con el fin de implementar los distintos algoritmos. A continuación se presenta el estado del arte que permite consolidar las bases de la presente investigación.

2.1 Antecedentes del marco de investigación

La revisión bibliográfica permitió el reconocimiento de estudios que aplican distintos métodos en el uso de sistemas de recomendaciones. Autores como: Davidsson (2010), Pradel et al. (2011), Viljanac (2014), entre otros, evidencian la implementación de algoritmos de sistemas de recomendaciones en diversos casos, tales como ofertar productos y servicios mediante plataformas web, seguridad informática y mejora en sistemas de búsqueda de contenidos.

En el desarrollo de algoritmos útiles para este tipo de técnicas, es necesario realizar ciertas transformaciones correspondientes a la matrices de datos. Son considerados aspectos de cercanía o similitud entre usuarios e ítems, así como el uso de modelos híbridos que consideran a la vez el efecto que pueden causar los distintos usuarios activos en la predicción.

2.2 Aplicación de sistemas de recomendaciones en el mundo

En muchos casos, las empresas buscan satisfacer a sus clientes, de acuerdo a sus preferencias, y es en este caso, en que los sistemas de recomendaciones, toman un papel importante, orientando la toma de decisiones, para la adquisición de algún producto que pudiese ser de su interés. La idea fundamental de estos métodos es la automatización de sugerencias que ocurren entre las personas (Vinodhini, Rajalakshmi, & Govindarajalu, 2014).

Respecto al modelamiento de algoritmos útiles en sistemas de recomendaciones, es posible identificar distintos métodos, tales como vecinos más cercanos, descomposición en valores singulares, modelos basados en usuarios, en contenido, en conocimiento y recomendación híbrida (Vinodhini, Rajalakshmi, & Govindarajalu, 2014).

Algunos estudios recientes plantean distintas configuraciones para las aplicaciones de los sistemas de recomendaciones. Respecto a esto, un trabajo comparativo relacionado a la elección de los modelos de recomendaciones fue publicado por Pradel et al. (2011). En este se plantea una solución de un sistema automatizado de recomendaciones para una tienda de productos para decoración y mejora del hogar. El proyecto contempla la capacitación de los vendedores, para la entrega de respuestas más profesionales a la hora de satisfacer los requerimientos de los compradores, y de esta forma, ayudar a que estos no regresen a su trabajo con insumos faltantes. Los enfoques de filtrado colaborativo comparados en este caso son: basado en memoria, basado en modelo, aprendizaje mediante reglas de asociación, donde este último obtuvo mejores resultados en el proceso de validación. Se utilizó para este experimento un conjunto de datos de compras reales de la tienda.

Viljanac (2014) probó un motor de recomendaciones denominado App-Detective en un estudio relacionado con el uso de un sistema de recomendaciones para aplicaciones móviles. AppDetective se comparó con algunos sistemas de recomendaciones clásicos de uso común en la actualidad. En esta pro-puesta se afirma que AppDetective presento ventajas por sobre el resto de los algoritmos.

Un trabajo propuesto por Davidsson (2010), también enfocado al uso en plataformas móviles, plantea el uso de un prototipo de sistemas de recomendaciones, considerando la ubicación del usuario y aplicaciones populares cercanas en distintos momentos del día. En esta investigación se demostró que el uso de ciertas aplicaciones, depende del contexto en el que un usuario esta presente.

Es posible concluir que la implementación de sistemas de recomendaciones puede tener distintos enfoques, permitiendo integrar estos de distintas ma-neras y con distintos algoritmos, considerando tipos de variables que tienen relación con el usuario, ya sean evaluaciones anteriores, contextos de tiempo o espacio, entre otras. A través de esta herramienta se pueden personalizar sitios web de comercio electrónico impulsando las ventas minoristas y el crecimiento de los negocios, mejorando la posibilidad de adquisición de

los compradores según sus intereses.

La utilización de sistemas de recomendaciones principalmente se orienta a la venta de productos y servicios mediante el uso de plataformas web pero también es posible que estos puedan implementarse en otras situaciones. Complementando los estudios citados con anterioridad, son presentados algunos trabajos que aplican sistemas de recomendaciones en otros casos distintos a la venta de productos y servicios.

Ruotsalo (2010) aborda un tema respecto de mejoras de sistemas de recomendaciones basado en contenidos, específicamente en métodos que hacen uso de ontologías y recuperación de la información. Este trabajo es parte del desarrollo de un sistema de información del patrimonio cultural para el portal CULTURESAMPO y el sistema móvil SMARTMUSEUM en Finlandia, proponiendo métodos para el mejoramiento del análisis del contenido y recuperación precisa de este. Este trabajo se concentra en la contribución en las áreas de análisis, heterogeneidad y recuperación de contenido. Las aplicaciones de esta investigación se orientaron en la creación de un sistema de recomendaciones tanto para para el portal CULTURESAMPO y el sistema móvil SMARTMUSEUM, implementándose y siendo aplicado satisfactoriamente en los ensayos con los usuarios.

Gadepally et al.(2016), describen el uso e implementación de sistemas de recomendaciones desarrollado por el Lincoln Laboratory, en el departamento de defensa de EEUU para el combate del cibercrimen. En la descripción de estos sistemas se menciona el filtrado de imágenes, fechas, horas y palabras claves que permiten la retroalimentación y de esta forma adaptar el análisis de diversas necesidades.

Cada uno de los trabajos mencionados dan cuenta de la importancia del uso de sistemas de recomendaciones en distintos casos de uso cotidiano en el mundo moderno. El enfoque de los sistemas de recomendaciones para el filtrado de información aún esta en sus inicios y muchas de estas técnicas son aplicadas cada vez con mayor frecuencia en empresas y organismos del estado mejorando a medida que las necesidades aumentan.

Capítulo 3

Marco Teórico

En esta sección se realiza una definición de los conceptos principales que se abordan en esta investigación, exponiendo algunos temas como: una reseña de la historia del concurso del *Netflix Prize*, incluyendo referencias de arquitecturas utilizadas, se abordan ideas tales como *Big Data*, Hadoop, considerando sus componentes más importantes (HDFS+MapReduce) y su respectiva implementación en el servidor con el sistema operativo Ubuntu Server 12.04 LTS en funcionamiento con el software R. En cuanto al uso de técnicas para el análisis de los datos se abordan temas tales como: sistemas de recomendaciones, utilización y descripción de un modelo útil para este problema, especificando, en cada caso, los fundamentos básicos asociados a estos.

3.1 El desafío *Netflix Prize*

Netflix es una empresa estadounidense dedicada al entretenimiento disponible en más de 190 países. Es un servicio que proporciona mediante tarifa mensual, asociada a una cuenta de usuario, contenido de galardonadas producciones originales de *Netflix*, largometrajes, documentales, series y mu-

chos otros. El contenido ofrecido puede tener variaciones según la ubicación geográfica, así como también con el paso del tiempo. Es posible reproducir, pausar y volver a ver contenido en alta definición y sin ningún tipo de publicidad. *Netflix* permite elegir distintos tipos de planes de membresía, para que el cliente seleccione el más adecuado según sus preferencias (Netflix, Inc., 2017).

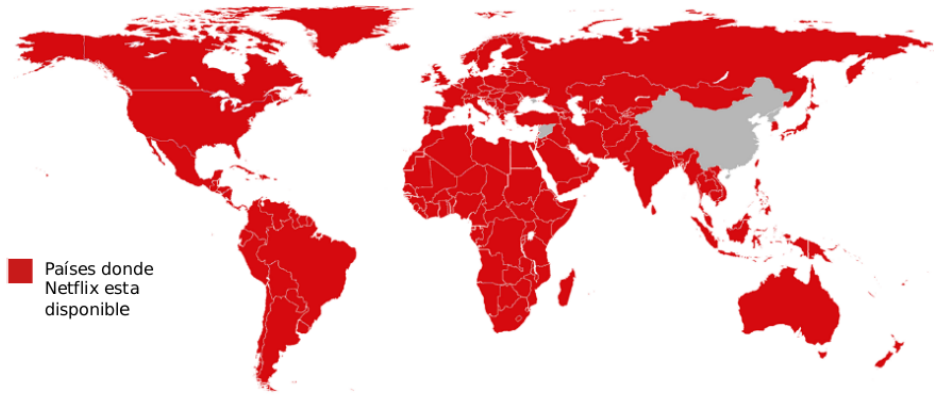

Figura 3.1: Países en donde el servicio de *Netflix* puede ser solicitado. (Adaptado de Netflix, Inc., 2017).

En octubre del año 2006, *Netflix* inicia una competencia para todos los interesados en participar, desafiando a los expertos en las áreas de aprendizaje de maquinas, minería de datos, estadística e informática, para superar al algoritmo de recomendaciones utilizado por la compañía, llamado Cinematch. En este desafío participaron 51.051 personas repartidas en 41.305 equipos distintos de 186 países (Pita, 2016).

Los participantes de este evento, podían competir ya sea de forma individual o colectivamente, en este ultimo caso, mediante la conformación de equipos, debiendo registrarse en el sitio oficial del *Netflix Prize*. Los equipos, tenían la misión de proporcionar el nombre del equipo, nombre completo de cada integrante, correo electrónico, afiliación, país y además elegir un líder, que sería el encargado de representar en aquellas instancias que fuese necesario. Enviar, recibir y comunicar las informaciones del concurso, era responsabilidad del líder de cada grupo. Estaba permitida la participación para

que los distintos participantes pudiesen formar parte de distintos equipos. No estaban permitidos los equipos con un conjunto idéntico de miembros. Cualquier dato del equipo que sea visible al público, podía cambiarse, si se consideraba inapropiado por los organizadores del evento. Era deber del equipo participante y no de los organizadores del evento, el definir con anticipación la forma en que serían distribuidos los premios entre los integrantes y organizaciones afiliadas. Considerando las decisiones de equipo, informadas por cada líder a la compañía, esta realizaría la entrega de los premios oportunamente (*The Netflix Prize Rules*, 1997-2009).

Netflix reglamentó que los participantes inscritos en la competencia tenían opción de representarse a si mismos, actuar como representantes de empresas o instituciones académicas. Era responsabilidad de los participantes revisar, comprender y cumplir con las políticas de su institución o empresa, verificando la viabilidad de participación y compatibilidad en dicha competencia. Si era descubierta y denunciada la violación de políticas de escuela o empresa, era causal para la descalificación del o los participantes, siendo negada la opción de seguir en la competencia, recibir o retener premios. La compañía definió que sus empleados con contrato en vigencia y anteriores, filiales, agentes de *Netflix*, junto con sus familiares inmediatos, cónyuges, y con quienes compartían el hogar, contratistas independientes, estaban excluidos de poder participar en la competencia, recibir o retener los premios concertados en el reglamento. Residentes en Cuba, Iran, Siria, Corea del Norte, Sudan y Myanmar , tenían negada la posibilidad de participación, y el concurso fue declarado nulo en estos países. *Netflix* definió renunciar a toda responsabilidad que surgieran por disputas entre estudiantes y su institución educacional, o empleados y su empresa, relacionados al concurso. La compañía reservó el derecho a limitar o restringir la participación a cualquier persona, en cualquier momento y por cualquier razón que se considerara pertinente (*The Netflix Prize Rules*, 1997-2009).

Toda la información ingresada y presentada debía ser recolectada y evaluada en Estados Unidos, y estar en idioma inglés. Toda información referente a los equipos era confidencial, hasta el momento de que se definiera un ganador. *Netflix* no utilizaría la información del registro de los participantes

con fines comerciales. Cada participante debía dar su aprobación respecto del cumplimiento de estas reglas (*The Netflix Prize Rules*, 1997-2009).

Los participantes, una vez efectuado el registro, tenían la posibilidad de acceder a un conjunto de datos de prueba, entrenamiento y calificación de concurso. Los datos de entrenamiento, se componían de más de 100 millones de calificaciones, con aproximadamente 480.000 clientes anónimos suscritos, elegidos de forma aleatoria considerando un total de 17.770 títulos de películas, divididas en archivos de formato texto. La recolección de los datos fue realizada entre los años 1998 y 2005, donde los suscriptores calificaban las películas disponibles en una escala de 1 a 5 estrellas (Netflix-Prize, 1997-2009).

Cada archivo de película, se componía de una asignación con un correlativo de forma aleatoria a cada cliente, nombre de película vista, calificación a la película y fecha en que fueron realizadas las calificaciones. Adicionalmente al conjunto de entrenamiento, los concursantes tenían acceso a un conjunto de clasificación, que contenía 2,8 millones de registros pareados de clientes y películas, donde era retenida la calificación. Los participantes debían efectuar predicciones para todo el conjunto de clasificación. Donde el archivo de calificación solo contenía dos columnas a diferencia del conjunto de entrenamiento, el ID de cliente y fecha. Los concursantes debían entregar las predicciones del respectivo algoritmo en el formato especificado en las instrucciones. Un ejemplo para visualizar el formato del conjunto de clasificación y del conjunto de predicción se puede apreciar en la Tabla 3.1.

Tabla 3.1: Ejemplo formato de datos *Netflix Prize*

Conjunto de clasificación	Formato de la predicción
1:	1:
3245, 2005-12-19	2.5
5667,2005-12-24	4.4
225:	225:
1234,2005-06- 24	1.2
1245,2001-04-11	2
1111,2004-04-12	4.0

La predicción para el usuario 3245, el 19 de diciembre del año 2005, en la película 1 fue un 2.5. Del mismo modo se puede interpretar la predicción para el usuario 1245, la que fue un 1.2, el 11 de abril del año 2001 para la película 225. Los envíos que no cumplieran con los requisitos del formato estipulado, era una causal de rechazo de estos, por parte de los organizadores.

El conjunto de clasificación se dividió en dos conjuntos disjuntos que contenían pares de usuarios y películas seleccionados al azar, donde la asignación de pares para cada subconjunto era desconocida para los participantes. Se realizaría el cálculo del *RMSE* (Root Mean Square Error) para cada subconjunto. Esta medida es definida como un nivel de error asociado a la predicción, en donde un valor pequeño de este representa un algoritmo con un un mejor desempeño. Hahsler (2017) señala que el *RMSE* puede ser calculado utilizando la siguiente expresión:

$$RMSE = \sqrt{\frac{1}{S} \sum_{(u,i) \in S}^{S} (r_{ui} - b_{ui})^2}, \tag{1}$$

donde:

- S : es la cantidad de registros comparados del conjunto de predicción en el conjunto de prueba.
- r_{ui} : es la calificación del usuario u a un ítem i.
- b_{ui} : es la predicción para la calificación de un usuario u para un ítem i .

El *RMSE* para el primer subconjunto de prueba perteneciente al conjunto de clasificación, se informaba públicamente en el sitio, mientras que el el valor del *RMSE* para el segundo conjunto no se informaba el puntaje obtenido, y se utilizaba para identificar posibles ganadores. Los conjuntos de predicciones no serían públicos durante el plazo que durara la competen-

cia y pasarían a ser propiedad de *Netflix*, reservándose el derecho a efectuar evaluaciones adicionales (*The Netflix Prize* Rules, 1997-2009).

Para proteger la identidad de los subscriptores, se suprimió toda información de carácter personal. En los conjuntos de clasificación y entrenamiento, algunos de los datos fueron perturbados deliberadamente, ya sea, eliminando, modificando o insertando calificaciones y fechas. Sin embargo, la puntuación reflejada del *RMSE* con datos perturbados, no alteraría significativamente del *RMSE* del conjunto original. *Netflix* dio a conocer la marca desempeñada por el algoritmo de Cinematch entrenado en el conjunto de datos, donde su *RMSE* otorgaba una marca de 0.9525 (*The Netflix Prize* Rules, 1997-2009).

Desde el momento que comenzaba el concurso y después de 3 meses, si algún concursante, mejoraba más allá del *RMSE* de clasificación, un anuncio electrónico informaba a todos los participantes que dentro de un plazo de 30 días, debían presentar sus predicciones, para poder ser considerados y evaluados según los criterios de las reglas del concurso. No sería responsabilidad de la *Netflix*, la perdidas, trabajos no entregados, problemas con el proveedor de servicio de Internet, satélites, correo electrónico o postal, fallo de conexiones o disponibilidad, fallo en computadoras, distorsión en la transmisiones, etc. (*The Netflix Prize* Rules, 1997-2009).

Entradas con un ingreso más allá de la fecha limite de entrega no serían recepcionadas. En el caso de valores de *RMSE*, iguales, se verificaba cual de los participantes había enviado el conjunto de predicción de forma más temprana. Si no había envío de predicciones, ningún premio podía ser entregado y la competencia volvería a comenzar en un plazo fijado por la compañía. Las decisiones que emitirían los jueces de la competencia eran finales, sin excepciones(*The Netflix Prize* Rules, 1997-2009).

Un algoritmo que pudiese ser elegido, debía ser desarrollado originalmente, sin infringir derechos de autor, ser escrito en inglés y no haber requerido el uso de software privativos, ni de terceros (*The Netflix Prize* Rules, 1997-2009).

El concurso comenzaría el 2 de octubre del año 2006 y en un inicio, se tenía previsto que tuviera una continuidad al menos hasta el 2 de octubre del 2011. El *Grand Prize*, constaba de un premio de 1 millón de dolares que sería entregado a los participantes, que obtuvieran la mejor marca de al menos un 10% de mejora con respecto al algoritmo de Cinematch. Si la marca del 10% de mejora no se cumplía, la competencia entregaría un premio de 50.000 dolares cada año, llamado *Progress Prize*. Para que los participantes pudiesen calificar en el *Progress Prize*, la medida del *RMSE* del año correspondiente, debía ser menor o igual a la medida de precisión establecida por los jueces el año anterior. Para poder ganar y adjudicarse los premios, los participantes debían tener la mejor marca de precisión con respecto a sus competidores, además compartir su metodología y describir al mundo el por qué funcionaba. Posteriormente, los ganadores individuales o por equipos, debían conceder a *Netflix* (incluyendo filiales, subsidiarios, empleados, agentes y contratistas) una licencia no exclusiva, de libre uso del algoritmo, para distribuir, reproducir y crear trabajos derivados (*The Netflix Prize Rules*, 1997-2009).

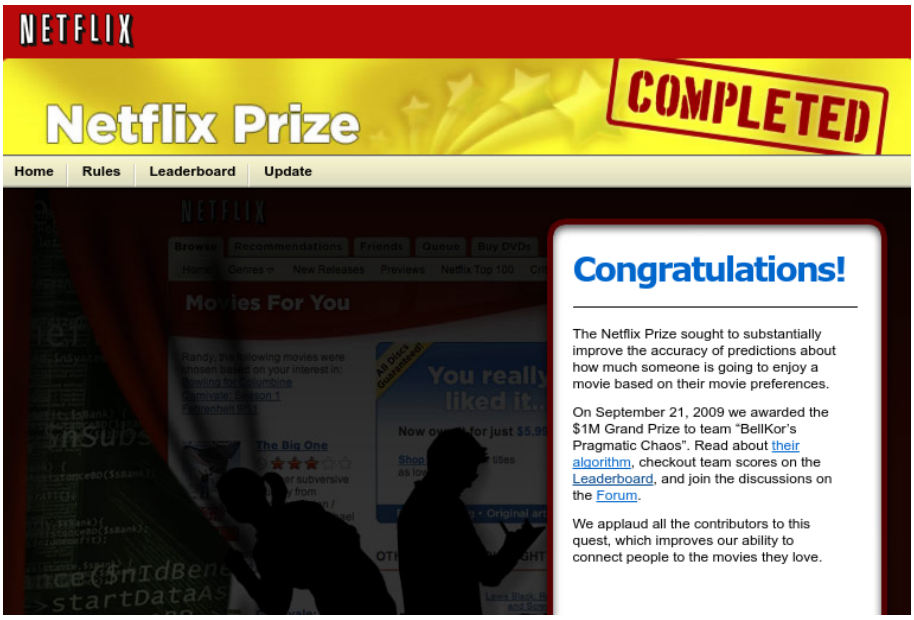

Figura 3.2: Página oficial *Netflix Prize*
(*Netflix Prize*, 1997-2009)

El primer *Progress Prize*, fue entregado al equipo KorBell (conocido también como BellKor), conformado por Yehuda Koren, Robert Bell y Chris Volinsky de T&T Research Labs, donde la marca desempeñada en su *RMSE* fue de 0.8723, el 1 de octubre del año 2007 a las 23:29:20 UTC, logrando una mejora de un 8,43%. respecto al algoritmo de Cinematch. La presentación del algoritmo fue realizada por Yehuda Koren, el 19 de noviembre del mismo año, en una charla publica. En ese instante, según el reglamento vigente, permanecían disponibles el *Progress Prize* y *Grand Prize*, para el año siguiente.

El segundo *Progress Prize* fue adjudicado al equipo BellKor in BigChaos, conformado por Andreas Töscher, Michael Jahrer de Commendo Research (originalmente pertenecientes al equipo BigChaos) y por Yehuda Koren, Robert Bell y Chris Volinsky de T&TResearch Labs. La marca obtenida de su *RMSE* fue de 0.8627, el 30 de septiembre, del año 2008 a las 21:17:40 UTC, estableciendo una mejora de 9,44% respecto del algoritmo de Cinematch. De acuerdo al reglamento vigente del concurso, Andreas Töscher y Michael Jahrer realizarían una presentación de su algoritmo, en charla pu-blica en las oficinas de *Netflix* el 17 de diciembre (2008). Según la actualización de la tabla de clasificación y el reglas del concurso, el único nivel restante, correspondía a la superación de la marca del 10% de mejora, sobre el nivel de la precisión del algoritmo de Cinematch.

El día 26 de julio del año 2009, a las 18:18:28 UTC, el equipo BellKor's Pragmatic Chaos, integrado por Bob Bell, Martin Chabbert, Michael Jahrer, Yehuda Koren, Martin Piotte, Andreas Töscher y Chris Volinsky, obtienen la puntuación ganadora con un *RMSE* en el conjunto de prueba de 0,8567, correspondiente a un 10,06% de mejora sobre el algoritmo de Cinematch, desde el inicio de la competencia. El premio fue entregado en la ciudad de Nueva York, Estados Unidos, el día 21 de diciembre del año 2009, otorgándose el *Grand Prize* por la suma de 1 millón de dolares.

Leaderboard

Showing Test Score. Click here to show quiz score

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Figura 3.3: Líderes del desafío *Netflix Prize*.
(*Netflix Prize*, 1997-2009)

Una anécdota que se dio al finalizar esta competencia, es que el algoritmo presentado por el equipo BellKor's Pragmatic Chaos, nunca llego a implementarse en los servidores de *Netflix* para generar recomendaciones, debido a su complejidad (Pita, 2016).

Figura 3.4: Fotografía ganadores del concurso *Netflix Prize*.
(Extraída de Irizarry & Hicks, 2016)

3.2 La filosofía del *Big Data*

En la sociedad actual, es posible ver los frutos del avance tecnológico, donde se puede lograr compartir o intercambiar información, estando a muchos kilómetros de distancia, en tan solo un instante. Cada casa cuenta con una o más computadoras, teléfonos fijos y móviles. Complejos sistemas computacionales en las oficinas y de modo discreto la información que se genera a diario comienza a ser de utilidad. Medio siglo después que las computadoras entraran en el medio social, los datos comenzaron a acumularse de forma cada vez más rápida y algo interesante comienza a darse como novedad en el mundo. Ciencias como la astronomía y la genómica crecieron de forma sorprendente desde el año 2000, y fueron estas quienes adoptaron en primera instancia la terminología de *Big Data*, en que esta aún no cuenta con una definición rigurosa. El concepto a migrado a distintos campos de investigación, abarcando cada vez más fuentes del estudio humano (Mayer-Schönberger & Cukier, 2013). Hoy en el mundo se hace necesario lograr el procesamiento de datos a gran escala. De esta manera es posible extraer información valiosa para quienes desean adelantarse a los requerimientos de clientes y usuarios, que solicitan el uso de servicios y el lograr adquirir los productos con disponibilidad en el mercado. Prajapati (2013) identifica tres características del *Big Data*, como las más importantes, en que estas pueden ser definidas de la siguiente forma:

- Volumen: Es el atributo tal vez más característico, enfocándose en el tamaño de los conjuntos de datos y la capacidad que tengan los usuarios o empresas para almacenarlos en un formato físico, ya que estos crecen día a día, generando un torrente de información conjuntamente con el avance tecnológico. Un ejemplo para este caso, puede ser la cantidad de registros de puntuaciones registrada por los usuarios a sus películas favoritas, en una determinada página web.

- Velocidad: Con la implementación de sistemas automatizados, aumenta la producción en masa de productos y servicios, aumentando la recolección de datos, pero el almacenamiento de estos durante un

tiempo muy extenso puede ser demasiado costoso en términos de infraestructura, por lo que es necesaria una optimización operacional de estos, para ser resumidos y analizados en un tiempo prudente. Un ejemplo de este caso puede ser la velocidad a la que se transmite información, mediante el uso de correo electrónico en todo el mundo.

- Variedad: Los datos pueden tener distintas categorías, dependiendo de cómo estén organizados, donde estos pueden ser estructurados, no estructurados o semi-estructurados. Esta característica se refiere a varios tipos de datos que pudiesen existir y ser parte de un estudio, por ejemplo música, videos, documentos de texto, etc.

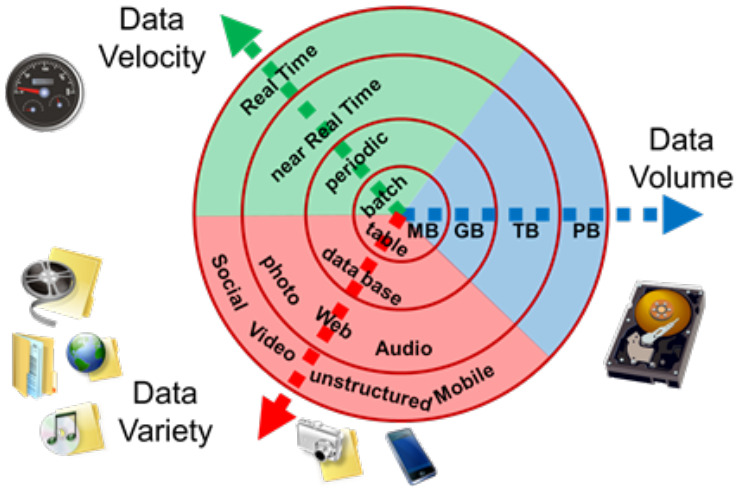

Figura 3.5: Paradigma de las 3V en Big Data.
(Extraído de Mishra, 2014).

La era del *Big Data* pretende instaurar una revolución en las técnicas, para el manejo de información, donde tecnologías de procesamiento tales como Hadoop, Spark, Storm y otras, permiten administrar muchos más datos que antes, en que estos no tienen que estar necesariamente asignados en formatos de tablas, ni de un formato estructurado.

El momento actual que vive el mundo, tiene que ver con la comprensión y percepción que se tiene con los datos disponibles, el familiarizarse, entender

la naturaleza de estos y su importancia en la toma de decisiones. Cuando los especialistas mencionan que los datos tienen la capacidad hablar, se refiere a que si se comprende el lenguaje de los datos es posible entender la dinámica de los tiempos modernos. La verdadera revolución no esta en las maquinas que realizan sorprendentes cálculos en un pequeño instante, sino en los datos mismos y en el uso que se les da a estos (Mayer & Cukier, 2013).

Considerando lo anterior, la importancia del análisis de datos a gran escala, hoy en día es de vital importancia en el mercado, ya que con esto es posible predecir y adelantarse a los requerimientos de los clientes y consumidores, para que estos busquen adquirir nuevos productos y servicios. Contar con especialistas en el procesamiento grandes cantidades de información, puede permitir además avances significativos a la hora de superar a los competidores y desplazarlos del mercado. El procesamiento masivo de datos marca a su vez, una importante transformación, y considerar este precedente puede ser una ventaja competitiva para aquellas empresas que tengan como opción el innovar en este nuevo universo digital en constante expansión.

3.3 Hadoop

Hadoop es una infraestructura digital de código abierto, desarrollada en lenguaje Java, por Doug Cutting, para la administración de *clusters* y procesamiento en paralelo a gran escala (Prajapati, 2013). Esta herramienta se fundamenta en un documento publicado en el año 2004 por Google, llamado "MapReduce: Simplified Data Processing on Large Clusters"(Dean & Ghemawat, 2004). En un principio, Hadoop fue desarrollado con el objetivo de apoyar el proyecto del motor de búsqueda web llamado "Nutch". Posteriormente Hadoop se separa de Nutch y emprende como proyecto independiente, bajo la licencia de fundación Apache (Wadkar, Madhu & Venner, 2014).

La implementación de esta arquitectura ha facilitando nuevas posibilidades en el campo de la ciencia de los datos (*Data Science*), abriendo el camino a explorar nuevas fuentes de información, en una amplia gama de áreas del conocimiento. Hadoop permite soluciones computacionales optimizando los recursos y minimizando los costos en el procesamiento de datos.

En muchas situaciones hoy en día, hablar de Hadoop, es hablar de *Big Data*, puesto que esta herramienta a tomado un rol importante en el ultimo tiempo en el procesamiento de grandes cantidades de datos y donde se hace cada vez más común en el mundo. El desarrollo y mejoramiento de esta herramienta, se ha instaurado en empresas, instituciones de gobierno, universidades y centros de investigación científica, sumando nuevas y cada vez más sofisticadas herramientas tales como, aprendizaje automático, procesamiento de imágenes y del lenguaje natural. Hoy en día existen variados sustitutos, mutaciones y posibles sucesores de la arquitectura de Hadoop, donde este ve aún un largo recorrido por delante (Wadkar et al., 2014).

Hadoop es una arquitectura que en su núcleo es un marco MapReduce basado en Java. Sin embargo considerando el rápido interés que despertó esta arquitectura, nace la necesidad de apoyar a las comunidades de usuarios diferentes de Java. Con respecto a esto Hadoop evoluciona, realizando mejoras vinculadas a subproyectos, en apoyo a estas comunidades. Wadkar et al. (2014) indican a continuación algunos subproyectos de Hadoop:

- Hadoop Streaming: Esta plataforma permite el uso de MapReduce con cualquier script de lineas de comandos, permitiendo el uso por parte de programadores de script en Unix como Python, R y muchos otros.

- Hadoop Hive: Es una plataforma que permite simplificar la programación en MapReduce, dado que esta tarea puede ser muchas veces compleja y difícil de manejar. Hive evoluciona para poder proporcionar almacenamiento a grandes conjuntos de datos permitiendo a los usuarios enfocarse en el problema en lugar de implementaciones de bajo nivel. Los usuarios de Hive pueden realizar consultas mediante un lenguaje llamado *Hive Query Language*, muy similar a SQL. Esta

arquitectura permite que los usuarios más avanzados puedan realizar desarrollo de funciones mediante el uso del entorno de Java, también admitiendo controladores como ODBC (*Open Database Connectivity*) y JDBC (*Java Database Connectivity*).

- Hadoop Pig: Pig es un lenguaje que funciona bastante bien en escenarios en que es necesaria la administración de datos, apropiado para el uso de aplicaciones de extracción, carga y transformación. Esta plataforma puede ser atractiva para programadores dedicados al procesamiento de información, como por ejemplo usuarios SAS, R o Python.

- HBase: Plataforma para la búsqueda y almacenamiento de datos en tiempo real en Hadoop. Considérese el ejemplo de búsqueda del perfil de un usuario en la red social Facebook. Lo lógico debería ser encontrar el nombre del usuario de manera instantánea, no después de un prolongada carga de un gran conjunto de datos. Estas motivaciones permitieron desarrollar la plataforma HBase

El proyecto Apache Hadoop se estructura mediante dos componentes que son los que permiten el procesamiento de cálculo distribuido, estos son el HDFS (*Hadoop Distributed File System*) y MapReduce, descritos a continuación.

3.3.1 Descripción de HDFS

HDFS es un sistema de almacenamiento de archivos para Hadoop, basado en Unix y en el sistema de archivos de Google. Esta arquitectura fue pensada con el fin de almacenar grandes cantidades de datos y para poder ser utilizados en equipos de bajo costo (Wadkar et al., 2014).

En este sistema la información es dividida en bloques replicándose en los discos duros locales de los nodos pertenecientes al *cluster*, para realizar respaldos de seguridad en caso de fallos (Wadkar et al., 2014).

El sistema de archivos distribuidos de Hadoop se puede presentar como una arquitectura de tipo esclavo/maestro, capaz de almacenar de manera óptima grandes cantidades de información. Prajapati (2013), menciona tres componentes (service daemons) que son fundamentales para el adecuado funcionamiento de HDFS, los que se definen a continuación:

- *NameNode*: Nodo maestro del HDFS, encargado de la gestión de espacio de nombres y los ajustes de accesos a los archivos por parte del cliente (abrir, cerrar, renombrar y más). Su función es replicar los datos en bloques (que por defecto es de orden tres), administrar el almacenamiento y ubicación de estos bloques en DataNode. El tamaño de cada bloque HDFS, por defecto es 64 MB, en que si es necesario se puede aumentar.

- *DataNode*: Son los nodos esclavos desplegados en cada maquina, encargados de proporcionar almacenamiento real, además de la creación y eliminación de bloques.

- *Secundary NameNode*: Servicio (daemon) encargado de realizar chequeos periódicos y mantenimiento del NameNode. En caso de que pudiera fallar en algún momento el NameNode, este puede ser reemplazado con una imagen instantánea almacenada por los chequeos del Secundary NameNode.

Estos componentes han sido diseñados par el funcionamiento de manera desacoplada en máquinas genéricas mediante sistemas operativos heterogéneos. HDFS a sido escrito en el lenguaje de programación Java, por lo tanto en cualquier equipo que permita el uso de este lenguaje, puede ser implementado. Con respecto a una implementación clásica de HDFS, esta consta de un equipo en que se ejecuta NameNode, y en el resto de los equipos se ejecutara DataNode (Rodríguez, 2014).

Las aplicaciones de lectura de ficheros realizadas por el cliente, deben ser contactadas al NameNode en primera instancia, para establecer el lugar de almacenamiento de la información que se necesita recuperar. NameNode

en respuesta al cliente, retorna un identificador del bloque más relevante y el nodo en el cual esta almacenada la información solicitada. Posteriormente el cliente establece conexión con DataNode para establecer la lectura y recuperación de los datos. Los bloques son apilados en el sistema local del equipo, y HDFS se ubica en la parte superior de la pila de un determinado Sistema operativo. Una característica importante de HDFS es que la información nunca se mueve a NameNode. Toda transferencia de información es generada entre los distintos nodos de datos y los clientes, donde la comunicación con NameNode solo implica la de meta-datos (Rodríguez, 2014).

3.3.1.1 Algoritmo de escritura en HDFS

Wadkar et al. (2014), describen una operación de escritura en HDFS como la creación de archivos, en que los siguientes pasos permiten que un cliente pueda ejecutar esta acción:

- El cliente comienza la transmisión del contenido del archivo a un archivo temporal, en el sistema de archivos local. Este procedimiento lo realiza antes de ponerse en contacto con NameNode.

- Cuando el tamaño de los datos alcanza alcanza el tamaño de un bloque (por defecto 128 mb), el cliente contacta al NameNode.

- NameNode crea un archivo en HDFS y notifica al cliente sobre el identificador de bloque y la ubicación de los DataNodes, donde esta lista de DataNodes contiene los nodos replicados.

- El cliente utiliza la información del paso anterior para poder liberar el archivo temporal en una ubicación del bloque de datos recibida del NameNode. Esto resulta en la creación de un archivo real en el almacenamiento local en DataNode.

- Cuando el archivo se cierra (archivo visto por el cliente), NameNode

confirma el archivo y se hace visible en el sistema.

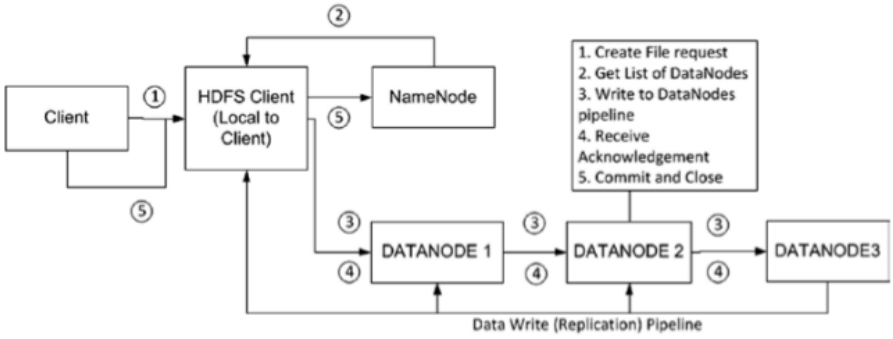

Figura 3.6: Diagrama de escritura de HDFS.
(Wadkar et al., 2014).

3.3.1.2 Algoritmo de lectura en HDFS

Segun Wadkar et al. (2014), los siguientes pasos permiten que un archivo pueda ser leído por un cliente desde HDFS:

- El cliente contacta con el NameNode realizando la devolución de la lista de bloques y sus replicas, con sus respectivas ubicaciones.

- Cuando el cliente inicia la lectura del bloque, esta se realiza directamente conectando con DataNode. Si el DataNode falla, el cliente establece contacto con aquel DataNode que aloja la replica.

- En la lectura del bloque, se realiza el cálculo de suma de comprobación (*checksum*), realizando una comparación con la suma de comprobación del momento de la escritura del archivo. Si la suma de comprobación falla, la recuperación del bloque es extraída desde la réplica.

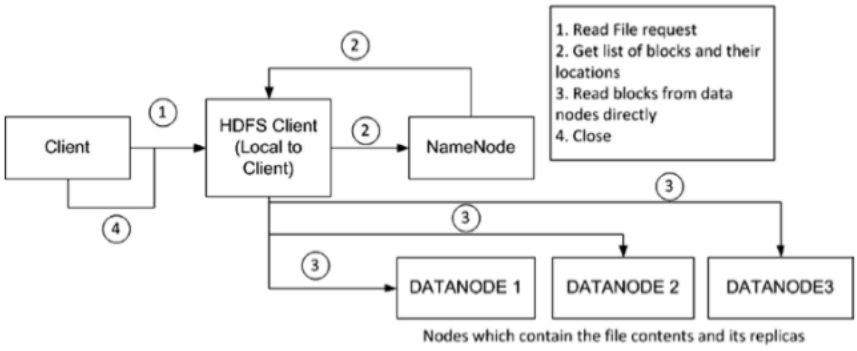

Figura 3.7: Diagrama para lectura en HDFS.
(Wadkar et al, 2014).

3.3.1.3 Algoritmo para suprimir archivos en HDFS

Segun Wadkar et al. (2014), los siguientes pasos permiten suprimir un archivo en HDFS por un cliente :

- NameNode solo se encarga de cambiar la ruta de almacenamiento del directorio a la papelera de Hadoop (*/trash*). El archivo eliminado permanecerá en la papelera, por medio de la actualización de metadatos vinculados al cambio de nombre de la ruta del archivo. El archivo antes de ser eliminado permanecerá en la papelera durante un intervalo de tiempo predefinido de 6 horas, en que durante este tiempo puede ser restaurado.

- Una vez cumplido el tiempo que permanece un archivo en la papelera de Hadoop, NameNode se encarga de suprimir su rutas de accesos y nombres (namespace).

- Los bloques que almacenaban el archivo son liberados y de esta forma el sistema tiene un mayor espacio disponible.

3.3.2 Descripción de la arquitectura MapReduce

MapReduce nace a principios de los años 80, en los inicios de la computación distribuida, en que empresas como Google, solicitaban gran capacidad de computo para el procesamiento de información, en los motores de búsqueda de su página web (*Page Rank*). Posteriormente se enfoca este trabajo al tratamiento de grandes volumenes de datos, mediante la creación de algoritmos capaces de llevar a cabo tareas de varios terabytes. De ahí la emergente popularidad de MapReduce como una potente herramienta de cálculo (Hernández & Hernández, 2015).

Dean & Ghemawat (2004), describen MapReduce como algoritmo para la simplificación de procedimientos a gran escala, en *clusters* de computadores básicos, utilizados para mejorar la eficiencia en el manejo de distintas operaciones. El funcionamiento de esta plataforma permite a los programadores facilitar y optimizar la implementación de tareas complejas, donde el sistema se encarga de distribuir los datos de entrada, procesarlos de forma paralela y combinar una salida en un archivo, según los requerimientos definidos. Esto puede ser de gran utilidad en términos de tiempo, ahorro y eficiencia para el análisis de grandes cantidades de datos.

El paradigma de programación de MapReduce esta dividido en dos fases, definidas por Prajapati (2013) de la siguiente forma:

- Fase Map: posterior a la fragmentación de los conjuntos de datos, se asignan a un rastreador de tareas, para realizar la transformación de estos, y ser procesados de forma paralela.

- Fase Reduce: fase de resumen, donde todos los registros asociados deben procesarse conjuntamente. El nodo maestro debe recoger las respuestas a los subproblemas y genera combinaciones para conformar la respuesta al problema inicialmente planteado.

Al igual que HDFS, MapReduce es una arquitectura de tipo esclavo/maestro. La clásica implementación de MapReduce contiene el envío e inicial-

ización de trabajos, asignación y ejecución de tareas, progreso y actualización de estados, y además las actividades relacionadas al termino de un determinado trabajo. La gestión de estos trabajos son realizados principalmente por el nodo JobTraker y ejecutados por el nodo TaskTracker (Prajapati, 2013).

Prajapati (2013) & Rodríguez (2014) definen JobTraker y TaskTracker como componente fundamentales para el adecuado funcionamiento de MapReduce, donde sus definiciones se presentan a continuación:

- JobTracker: en un *cluster* solo existe un JobTracker, en que este es el nodo maestro en la arquitectura de MapReduce. Encargado de la gestión de trabajos y recursos en el *cluster* (TaskTracker).

- TaskTracker: estos son los esclavos desplegados en cada maquina, responsables de la ejecución Map/Reduce de las tareas según las instrucciones de JobTracker. Cada TaskTracker es capaz de ejecutar varias tareas asignadas por el Jobtracker en paralelo, esto permite garantizar que en caso del fallo de ejecución de alguna tarea, no afecte la ejecución del resto de las tareas, ni tampoco al propio JobTracker.

MapReduce es un modelo de programación que funciona mediante el paradig-ma de computación distribuida, pero no es el único en este campo, donde es posible identificar competidores como por ejemplo MPI y Bulk Synchronous Parallel, que son capaces de realizar tareas similares. Existen *software* como R, Python y muchos otros, que pueden ser útiles hoy en día en métodos para el aprendizaje de maquinas y análisis estadístico, necesitando *hardware* especializados para su óptimo uso, donde la programación distribuida puede ser complementaria y una clave para el manejo de grandes cantidades de datos. Lógicamente la implementación en un *cluster* de computadores agrupados, puede maximizar la capacidad de cálculo de las tareas encomendadas, donde MapReduce responde de muy buena forma al manejo y distribución de información a gran escala (Prajapati, 2013).

Prajapati (2013), menciona cómo aspecto clave del funcionamiento en MapReduce, es que el sistema asume que cada Map y Reduce, son indepen-

dientes de todos los Map y Reduce que están en proceso en la red, donde estas ope-raciones se ejecutan en paralelo en diferentes claves y listas de datos . El nodo JobTracker, se encargara de todas las responsabilidades relacionadas a los trabajos operados en MapReduce, tales como la ejecución de trabajos, programación de mapeadores, reductores, combinadores y particionadores, monitoreo de éxitos, fallas y finalización del trabajo.

Entre las compañías que utilizan MapReduce hoy en día es posible mencionar: Amazon, eBay, Google, Linkedin, Trovit, Yahoo!, Youtube, en donde además de estas hay muchas otras que están actualmente utilizando esta plataforma para el procesamiento distribuido de información a gran escala.

3.3.3 Funcionamiento de MapReduce en Hadoop

Prajapati (2013), menciona cuatro etapas para el procesamiento de datos utilizando MapReduce en Hadoop, descritas a continuación:

- Cargas de datos en HDFS: El conjunto de datos de entrada debe estar cargado en el directorio de Hadoop para que pueda ser manipulado por los nodos de MapReduce. Posteriormente HDFS, divide los datos de entrada (splits) y almacena estos en DataNode en el *cluster*, efectuando el factor de replicación (por defecto de orden 3). Posteriormente TaskTracker realizara los procesos de Map y Reduce para todas las divisiones del conjunto de datos de forma paralela.

- Ejecución de la fase Map: Mediante la ejecución de la aplicación cliente, se inician los procesos de MapReduce en Hadoop. En esta fase se realizan las copias y almacenamiento de recursos en HDFS, efectuando la solicitud a JobTracker para la realización de un trabajo. JobTracker inicializa el trabajo, recupera las entradas, divide la información y crea una tarea Map para cada trabajo. JobTracker efectua un llamado a TaskTracker para la aplicación de una tarea Map a cada subconjunto de entrada (Map_1, Map_2,...,Map_n), efectuando asignación

de un bloque del fichero (split), donde cada bloque se divide posteriormente en pares de ordenados de tipo clave y valor. Se generará al menos una salida para cada par de entrada (<key, value>). La lista de pares <key, value> se genera, de modo que el atributo correspondiente a la clave se repita muchas veces, para ser utilizado en la fase Reduce. Referente al formato, los valores de salida de Mapper y valores de entrada en Reducer deben estar en cantidades iguales. Posterior a completar la fase Map, TaskTracker mantendrá los resultados almacenados el buffer y el espacio utilizado en el disco local.

- Combinando y ordenando: Esta es una etapa intermedia de MapReduce en Hadoop, útil para la optimización de recursos en el *cluster*, en que tan pronto como la fase Map fue concluida, esta fase sera llamada automáticamente. Se realiza un ordenamiento por clave de cada partición, clasificando en función de esta, al lado del Mapper. La salida de este ordenamiento se almacena en una memoria intermedia disponible en el TaskTracker. El combinador es en si el Reducer en el proceso, en que esta no es una compresión de archivos de tipo Gzip, Winzip u otra similar, sino una acción de combinar registros que tengan una misma clave. Los datos que han sido devueltos por el combinador, son barajados y enviados a los nodos reductores. Para acelerar la salida de los Mapper a las ranuras de los Reducer asignados en TaskTracker, es necesario comprimir esa salida con la función "combiner". De forma predeterminada la salida de un Mapper se almacena en el buffer y si la salida es mayor que el umbral de este, el almacenamiento se realiza en un disco local. La salida de estos datos se disponen mediante HTTP (Hypertext Transfer Protocol).

- Ejecución de la fase Reduce: En las salidas de los Mapper, TaskTracker efectuá la recuperación de los datos de salida, agrupando y fusionando en un único gran archivo asignado a un proceso con método Reducer. El método reductor recibe una lista de valores de entrada, desde un ingreso, donde recibe las claves y una lista de valores (key,list(value)) donde efectuá una salida de tipo <key, value>. Las salidas del método Reducer de la fase Reduce, son escritas en HDFS, según el formato

especificado de el trabajo MapReduce que fue programado.

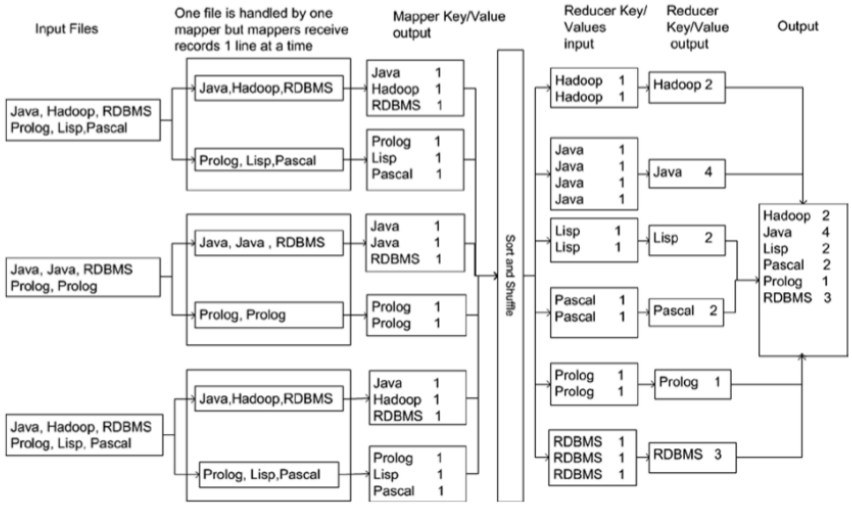

Figura 3.8: Diagrama de ejemplo de procesamiento en MapReduce (Wadkar et al., 2014).

3.3.4 Instalación de Hadoop

Según Prajapati (2013), el funcionamiento optimo de Hadoop, se realiza en sistemas operativos GNU/Linux. Si la instalación de Hadoop se requiere hacer en Windows, es necesario el uso de aplicaciones como VMware o Virtual Box para alojar algún sistema operativo GNU/Linux en una maquina virtual.

En este estudio la instalación es realizada en un ambiente Ubuntu 12.04, donde Prajapati (2013) describe a continuación tres modalidades para la instalación de Hadoop de la siguiente forma:

- Modalidad Autónoma: Esta es una modalidad, por defecto, que permite la instalación de Hadoop en un solo nodo, de manera no distribuida. En este caso, una operación Hadoop es ejecutado como un único proceso Java, junto con sus componentes: NameNode, Data-

Node, JobTracker y TaskTracker.

- Modalidad de procesamiento Pseudodistribuido: Al igual que en la modalidad autónoma, Hadoop es ejecutado en solo un nodo, simulando un proceso distribuido. Un JVM (Java Virtual Machine) es ejecutada para cada uno de los componentes o *demonios* por separado. El uso de Hadoop utilizando esta configuración es bastante limitada y en este caso solo puede ser utilizado con fines de prueba.

- Modalidad de procesamiento distribuido: Esta modalidad para la instalación de Hadoop permite una verdadera posibilidad de realizar procesamiento masivo de información. En este caso Hadoop es distribuido en múltiples maquinas, donde cada una de estas están configuradas para los distintos componentes con sus respectivos procesos JVM.

Para poder obtener un rendimiento máximo, de forma que Hadoop muestre toda su potencia en el procesamiento de información, es necesaria la implementación en un grupo de maquinas, en lugar de una sola.

En este estudio se hace necesaria la utilización e implementación de una modalidad de programación distribuida en un *cluster* de computadores, para poder procesar los datos pertenecientes al desafío del *Netflix Prize*.

3.3.4.1 Instalación de Hadoop modalidad de procesamiento pseudodistribuido

Kumar (2015), Anderson (2016) y Hong (2016) describen la instalación de Hadoop 2.6.5 para la modalidad de procesamiento pseudodistribuido en el sistema operativo Ubuntu, siguiendo los siguientes pasos:

- Instalación de Oracle Java 8

Se instala Java utilizando los siguientes comandos en la terminal de Ubuntu:

#Añadir el repositorio *webupd8team/java* al sistema
$ sudo add-apt-repository ppa:webupd8team/java

# Actualización de la lista de repositorios
$ sudo apt-get update

#Instalación Java Oracle 8
$ sudo apt-get install oracle-java8-installer

#Verificación de la versión de Java instalada
$ java -version

#Instalación por defecto de Java Oracle 8
$ sudo apt-get install oracle-java8-set-default

Es necesario editar el fichero de configuración de */etc/environment* para establecer las variables de entorno JAVA_HOME y JRE_HOME. Se puede realizar mediante el comando:

$ sudo pico /etc/environment

Una vez abierto el editor se requiere agregar las siguientes variables de entorno:

JAVA_HOME=/usr/lib/jvm/java-8-oracle
JRE_HOME=/usr/lib/jvm/java-8-oracle/jre

- Creación de un grupo llamado *Hadoop* generando un usuario *hduser*, donde este sera el propietario del directorio *hadoop*. Esto se puede realizar utilizando los siguientes comandos:

$ sudo addgroup hadoop
$ sudo adduser –ingroup hadoop hduser

- Instalación del servicio *ssh*

  Hadoop necesita acceso a *ssh* para la administración de sus nodos, permitiendo el uso de un modelo cliente-servidor para la comunicación entre sistemas remotos y el cifrado de datos entre ellos. Para ello es posible ejecutar los siguientes comandos:

  $ sudo apt-get install ssh
  $ which ssh
  $ which sshd

- Generación de clave *ssh* publica para la autenticación del usuario *hduser*

  $ su hduser
  Contraseña:
  $ ssh-keygen -t rsa -P ""

- Verificación de la configuración de conectividad *ssh* entre *hduser* y la maquina local. La ejecución de esta acción se realiza mediante el comando:

  $ ssh localhost

- La descarga de Hadoop se puede ejecutar vía terminal con los siguientes comandos:

  $ wget http://www-eu.apache.org/dist/hadoop/common/hadoop-2.6.5 /hadoop-2.6.5.tar.gz $ tar xvzf hadoop-2.6.0.tar.gz

- Es necesario agregar el usuario hduser al grupo sudo, se puede lograr

de la siguiente forma para el usuario local tomas:

$ su tomas
$ sudo adduser hduser sudo
[sudo] password:

- Es necesario volver al hduser, esto se logra de la siguiente forma:

  $ sudo su hduser

- Creación de un directorio llamado *hadoop*.

  $ sudo mkdir /usr/local/hadoop

- Ingreso al directorio *hadoop* − 2.6.5, esto se puede realizar utilizando el siguiente comando:

  $ cd hadoop-2.6.5

- Mover todos los ficheros del directorio hadoop-2.6.5 a hadoop de la siguiente forma:

  $ sudo mv * /usr/local/hadoop
  $ sudo chown -R hduser:hadoop /usr/local/hadoop

- Preparando la edición de archivos para configuración.

  Editar el archivo ∼/.bashrc, agregando al final de este documento las siguientes lineas para hacer permanente la definición de las variables de entorno:

  #HADOOP VARIABLES START
  export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
  export HADOOP_INSTALL=/usr/local/hadoop
  export PATH=$PATH:$HADOOP_INSTALL/bin

export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME= $HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_I
NSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INS
TALL/lib" #HADOOP VARIABLES END

# ejecutar el siguiente comando para abrir el archivo ∼/.bashrc
$ pico ∼/.bashrc
# Ejecutar el comando siguiente para guardar los cambios

$ source ∼/.bashrc

Es necesario establecer JAVA_HOME como ruta de acceso antes de /bin, de la siguiente forma:

$ which javac
/usr/bin/javac
$ readlink -f /usr/bin/javac

Es necesario establecer JAVA_HOME, para esto se debe editar el fichero hadoop-env.sh, agregando la siguiente linea:

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

Editar el archivo /usr/local/hadoop/etc/hadoop/core-site.xml

$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp

#Abrir el archivo y editar entre
<configuration></configuration> de la siguiente forma:

$ pico /usr/local/hadoop/etc/hadoop/core-site.xml

<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
<description>A base for other temporary
directories.</description>
</property>

<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value>
<description>
Nombre del sistema de archivos predeterminado
</description>
</property>
</configuration>

Copiar y renombrar el archivo mapred-site.xml.template y renombrar como mapred-site.xml de la siguiente manera:

$ cp /usr/local/hadoop/etc/hadoop/mapred-site
.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml

Editar el archivo mapred-site.xml:

$ pico /usr/local/hadoop/etc/hadoop/mapred-site.xml

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
<description>

</description>
</property>
</configuration>

Es necesario crear dos directorios que alojaran a DataNode y NameNode, ejecutando las siguientes lineas de comando:

$ sudo mkdir -p
/usr/local/hadoop_store/hdfs/namenode
$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
$ sudo chown -R hduser:hadoop /usr/local/hadoop_store

Editar el archivo hdfs-site.xml

$ pico /usr/local/hadoop/etc/hadoop/hdfs-site.xml

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
<description>

Default block replication.

</description>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>

- Es necesario formatear el sistema de archivos de Hadoop, esto se puede realizar utilizando la siguiente orden en la terminal:

  $ hadoop namenode -format

- Para iniciar Hadoop se puede correr el siguiente comando:

  $ start-all.sh

- Para verificar que Hadoop y sus componentes están funcionando se puede ejecutar el comando jps,de la siguiente forma:

  $ jps
  9026 NodeManager
  7348 NameNode
  9766 Jps
  8887 ResourceManager
  7507 DataNode

- Para detener el funcionamiento de Hadoop es posible lograrlo ejecutando la siguiente linea de comando:

  $ stop-all.sh

3.3.4.2 Instalación de Hadoop modalidad de procesamiento distribuido

Prajapati (2013), describe la instalación de Hadoop en modo distribuido utilizando dos equipos. Para esto es necesario configurar varios nodos en modalidad de *cluster* con un solo nodo.

- Después de realizar la instalación del *cluster* con nodo único, es necesario seguir los siguientes pasos:

1. En la fase de red, se utilizaran dos nodos para la configuración de Hadoop en modalidad completamente distribuida. Para establecer comunicación, los nodos deben estar en la misma red, en términos de la configuración de *software* y *hardware*.

2. Entre los dos equipos, debe considerarse un nodo maestro y su esclavo. Por lo tanto para poder realizar las operaciones de procesamiento con Hadoop, es necesario que el maestro este conectado al nodo esclavo. El ingreso para la maquina maestra es 192.168.0.1, mientras que 192.168.0.2 sera la maquina esclava.

3. Es necesario actualizar el directorio /etc/hosts en ambas maquinas. Después de esto debería poder visualizarse como 192.168.0.1 maestro y 192.168.0.2 esclavo.

4. Es necesario editar todos los archivos conf/*-site.xml en todos los nodos del *cluster*. En esta configuración se debe cambiar la etiqueta de la maquina local por la etiqueta del nodo maestro, además es necesario actualizar el factor de replicación a 2, fijado en la configuración de nodo único en 1.

5. Se debe actualizar realizar el formateo de HDFS, esto puede realizarse corriendo el siguiente comando desde el nodo maestro:

   bin/hadoop namenode -format

- Para probar la configuración del *cluster* de Hadoop, es necesario seguir los siguientes pasos:

  1. Inicializar todos los componentes de Hadoop

     $ bin/start-all.sh

  2. Detener el funcionamiento del *cluster* de Hadoop

```
$ bin/stop-all.sh
```

3.4 El software R

R es un entorno de computación enfocado al análisis estadístico, de amplia colaboración de usuarios en todo el mundo. Este software funciona bajo la licencia *GPL* (General Public Licence), donde su funcionalidad es multiplataforma, disponible para sistemas operativos *Unix*, *Gnu/Linux*, *Windows* y *Macintosh*. Fue desarrollado en la universidad de Aukcland en Nueva Zelanda, inicialmente por Robert Gentleman y Ross Ihaka, encontrándose actualmente a cargo del *Development Core Team* (Albert & Rizzo, 2012).

Este lenguaje de programación interpreta los comandos ingresados por un usuario en la ventana de la terminal o aquellos que envía desde un script y esos se evalúan cuando es presionada la tecla Enter (Albert & Rizzo, 2012).

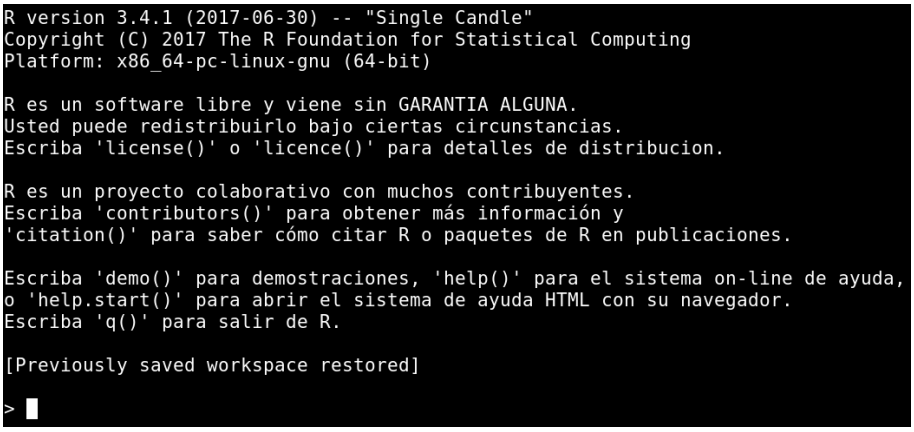

Figura 3.9: Interfaz gráfica *software R* en ambiente *Ubuntu 16.04 LTS*.

R cuenta con una gran cantidad de funciones y librerías integradas para distintas de las técnicas que permiten el análisis de datos. En la actualidad existen cerca de 11.000 librerías, donde la lista crece diariamente, más allá del alcance de cualquier libro. Es posible encontrar una lista completa de estas librerías en la página *http://cran.r-project.org*.

La utilización de este lenguaje se realiza mediante la linea de comandos, en que un lenguaje de programación asociado a este, permite realizar las tareas que sean requeridas en el procesamiento y análisis de los datos.

Es posible mencionar algunas de las implementaciones de técnicas que están disponibles en las distintas librerías de *R*, como por ejemplo:

- Operaciones de manejo de matrices y vectores de datos
- *Machine Learninig*
- *Clustering*
- Regresión Lineal
- Simulación
- Sistemas de Recomendación
- Big Data
- *Text Mining*
- Exploración de datos

R es un software que funciona de manera colaborativa, es decir que una de las bases de su éxito se lo debe al apoyo mutuo entre desarrolladores y la posibilidad de que la comunidad pueda realizar aportes para el mejoramiento de su código fuente. Al ser un *software* libre quien lo adquiera tiene la libertad de distribuir, copiar, mejorar y utilizar con el fin que estime conveniente.

Prajapati (2013, p.17) menciona algunas fuente populares que pueden ser útiles para el aprendizaje colaborativo del *software* estadístico *R*:

- *R mailing list*: Este es un grupo oficial de R, creado por los desarrolladores del proyecto.

- *R blogs*: *R* cuenta con una gran cantidad de blogs en que se comparte y desarrollan gran cantidad de aplicaciones del código. Uno de los bloggs más conocidos en la web por los usuarios es *http://www.r-bloggers.com/* donde todos los usuarios pueden realizar preguntas, responder a las inquietudes de otros usuarios, mostrar el desarrollo de algún tema, etc.

- *Stack overflow*: Esta es una plataforma multilenguaje de intercambio de conocimientos técnicos en que los programadores pueden plantear sus preguntas y junto con otros usuarios pueden llegar a una solución. Para más información puede visitar la página web: *http://stats.stackexchange.com*.

3.4.1 Instalación de R

La descarga de R puede realizarse ingresando a la página oficial del proyecto, en donde es posible elegir distintos repositorios de manera confiable, alrededor de todo el mundo. La instalación de R es multiplataforma, es decir, puede realizarse en distintos sistemas operativos. A continuación Prajapati, 2013; Anderson, 2016 describen de manera sencilla la instalación en sistemas operativos Ubuntu y Windows:

- Para instalar R en Ubuntu, es necesario seguir los siguientes pasos:

    1. Cambio de repositorios: R es un proyecto que esta en constantes actualizaciones de su código fuente, por lo que posiblemente en los repositorios de Ubuntu no estén disponibles la ultimas versiones. Es por esto que es recomendable agregar el repositorio de la página oficial de CRAN. Se deben ingresar los siguientes comandos vía terminal para agregar los repositorios de la página de CRAN.

$ sudo apt-key adv –keyserver keyserver.ubuntu.com –recv-keys E298A3A825C0D65DFD57CBB651716619E084DAB9

$ sudo add-apt-repository 'deb [arch=amd64,i386] https://cran.rstudio.com/bin/linux/ubuntu xenial/

2. Es posible realizar la actualización de la lista para paquetes disponibles, utilizando el siguiente comando:

   $ sudo apt-get update

3. Para instalar la ultima versión de R, se ingresa el siguiente comando desde la terminal:

   $ sudo apt-get install r-base

- Es posible realizar la instalación de R en Windows de la siguiente manera:

  1. Ingresar a la página *www.r-project.org*
  2. Click en la sección *CRAN*
  3. Elegir uno de los servidores disponibles
  4. Seleccionar la descarga de R para el sistema operativo Windows
  5. Ejecutar el archivo *.exe* para la instalación de R

3.5 Integración de R y Hadoop

Con anterioridad se definieron por separado distintas herramientas como *R* y Hadoop, con sus características y funcionamiento, para dar solución a al-

gunas tareas. La integración de R y Hadoop permite dar solución al análisis de datos a gran escala, de modo que Hadoop puede ser utilizado para la organización y almacenamiento de los datos, mientras que R se encarga de la parte operacional correspondiente al análisis. El funcionamiento conjunto de esta arquitectura de Hadoop, permite la construcción de un sistema escalable con características de ambas herramientas (Prajapati, 2013).

En este estudio se abordan dos formas de vinculación de R y Hadoop, mencionados a continuación:

- RHIPE
- RHadoop

Estas integraciones de R y Hadoop son definidas en los secciones 3.5.1 y 3.5.2.

3.5.1 RHIPE

Esta integración fue implementada por primera vez por Saptarshi Guha, en el Departamento de Estadística de la Universidad de Purdue el año 2012, para su tesis de doctoral. RHIPE actualmente continua siendo desarrollado en esta universidad por intermedio del Departamento de Estadística y algunos grupos de discusión en *Google* (Prajapati, 2013, p.64).

El funcionamiento de RHIPE se realiza por medio de las técnicas *Divide* and *Recombine*, para el procesamiento de datos en *Big Data*. Este procedimiento divide el conjunto de datos, y los cálculos son aplicados en los subconjuntos de este, en que posteriormente se combinan las salidas. Prajapati (2013) menciona que los objetivos por los que RHIPE fue implementado son las siguientes:

- Procesamiento y análisis tanto de grandes como de pequeños conjuntos de datos.

- Dar la posibilidad a usuarios de R para realizar distintas operaciones para el análisis de datos, utilizando un lenguaje sencillo. RHIPE esta diseñado con funciones que facilitan y ayudan a Hadoop, para realizar las labores de HDFS y MapReduce mediante la linea de comandos de la consola en R.

La ultima versión disponible en la actualidad es RHIPE 0.75.1.7, con fecha de modificación 11 de noviembre de 2015.

3.5.1.1 Instalación de RHIPE

Para la instalación de RHIPE es necesaria la instalación de R y Hadoop en un equipo de escritorio o *cluster*, donde estas han sido descritas en secciones anteriores. Prajapati (2013, p.65) describe a continuación los pasos que deben seguirse para la instalación de *RHIPE*:

- Instalación de Protocol Buffer: Estos son utilizados para poder serializar la información, donde RHIPE depende de los protocol buffer 2.4.1, para lograr esto. Para la instalación de estos se puede usar la terminal introduciendo los siguientes comandos:

  # descarga de *Protocol Buffer* 2.4.1
  $ wget http://protobuf.googlecode.com/files/protobuf-2.4.1.tar.gz

  # Descomprimiendo archivo
  $ tar -xzf protobuf-2.4.1.tar.gz

  # Para ingresar en el directorio donde se realiza la extracción
  $ cd protobuf-2.4.1

  # Instalación de *Protocol Buffer*

$ ./configure # –prefix=...
$ make
$make install

- Configuración de variables de entorno: para el correcto funcionamiento de *RHIPE* es necesario asegurar que las variables de entorno estén configuradas adecuadamente. Para configurar las librerías de Hadoop, es necesario establecer las variables PKG_CONFIG_PATH y LD_LIBRARY_PATH en el archivo ./bashrc de huser (Hadoop user) para establecer automáticamente cuando el usuario inicia sesión en el sistema.

  PKG_CONFIG_PATH es una variable de entorno que contiene el *script* de la ruta para la recuperación de librerías instaladas en el sistema, mientras que LD_LIBRARY_PATH contiene la ruta de acceso a librerías compartidas. Se puede realizar esta configuración de la siguiente forma agregando las siguientes lineas en el fichero ./bashrc vía terminal:

  export PKG_CONFIG_PATH = /usr/local/lib
  export LD_LIBRARY_PATH = /usr/local/lib

- Instalación de librería rJava: RHIPE es un paquete java, de esta forma este actúa como un puente Java de la integración R y Hadoop. Es posible instalar esta librería en R de la siguiente forma:

  > install.packages("rJava")

- Instalación de RHIPE: la instalación de efectuá desde la terminal, de la siguiente forma:

  #Descarga de la ultima versión de RHIPE desde los repositorios
  $ wget http://ml.stat.purdue.edu/rhipebin/Rhipe_0.75.1.7.tar.gz

  #Instalación de *RHIPE*
  $ R CMD INSTALL Rhipe_0.75.1.7.tar.gz

Después de esto es posible utilizar RHIPE en la integración entre R y Hadoop

3.5.1.2 RHIPE y su arquitectura

Para entender de mejor forma el funcionamiento de RHIPE, para el procesamiento masivo de información, se debe saber que existen una serie de componentes que permiten y optimizan su funcionamiento.

Los componentes de RHIPE son los siguientes:

- *Rclient*
- *JobTracker*
- *TaskTracker*
- HDFS

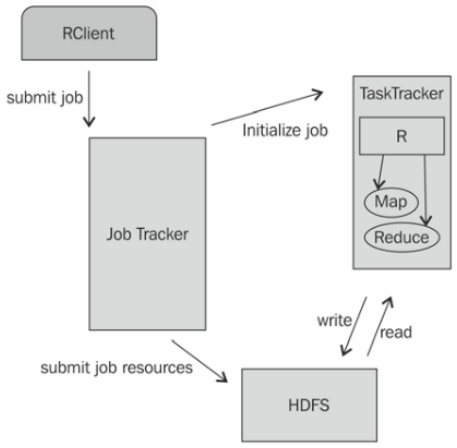

Figura 3.10: Componentes de RHIPE
(Prajapati, 2013).

Donde Rclient es una aplicación de R que permite llamar a JobTracker y de esta forma realizar tareas, indicando el uso de recursos como MapReduce, Mapper, Reducer, formatos de entrada y salida, así como otros parámetros

que se pudiesen manejar para el procesamiento de un conjunto de datos (Prajapati,2013, p.68).

Figura 3.11: Entorno gráfico de RHIPE

3.5.2 RHadoop

RHadoop es una plataforma de código abierto, diseñada para el análisis de datos a gran escala, utilizando Hadoop por medio de funciones y librerías de R. Esta plataforma ha sido desarrollada por Revolution Analitics, que es uno de los lideres comerciales basados en el proyecto de código abierto en R. Esta integración contempla el utilizar 3 librerías que son útiles para el funcionamiento conjunto con HDFS y MapReduce, herramientas que son funcionales a la arquitectura de Hadoop. A continuación Prajapati (2013) describe las características de estos paquetes:

- *rhdfs*: Se presenta como una interfaz que entrega la posibilidad de utilizar HDFS desde la terminal o script de R, donde el usuario puede realizar operaciones de lectura y escritura en los archivos de datos distribuidos (p.61).

- *rmr*: Interfaz útil para la funcionalidad de MapReduce en el entorno de *R*, donde el programador solo debe ajustar su algoritmo a la lógica de las fases *Map* y *Reduce* (p.61).

- *rhbase*: se define como una interfaz de R para la el manejo de datos en *Hadoop Hbase* para el almacenamiento en la red distribuida, utilizando *Thrift server*. Esta librería esta diseñada con el objetivo de

realizar operaciones relacionadas a la inicialización, lectura, escritura y manipulación de tablas (p.61).

En el funcionamiento de RHadoop no es necesaria la instalación de los tres paquetes, en que el paquete *rhbase* solo necesario instalar si se ha realizara el almacenamiento de datos en la fuente de datos de Hbase. Para la ejecución de Hadoop en la consola de R, en un *cluster* o equipo, bastara con la instalación de *rmr* y *rhdfs*. La librería *rhdfs* entregara soporte a R respecto de HDFS, mientras que *rmr* lo entrega respecto de MapReduce. Además Rhadoop también cuenta con un paquete llamado quick check encargado de depurar el trabajo de MapReduce del paquete *rmr* (Prajapati, 2013, p.76).

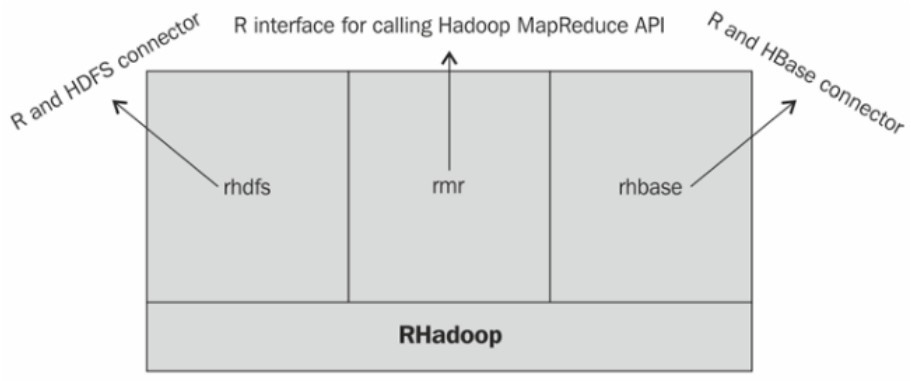

Figura 3.12: Ecosistema RHadoop
(Prajapati, 2013).

3.5.2.1 Instalación de RHadoop

Para utilizar esta integración, obviamente se necesita la instalación previa de R y Hadoop en un equipo personal o un *cluster*. Sera necesario el escalamiento de un *cluster*, si la cantidad de datos es demasiado grande, aumentando el numero de nodos. Considerando lo anterior RHadoop puede ser instalado en uno o múltiples nodos dependiendo la complejidad de una tarea y el volumen de información.

Prajapati (2013) menciona que para la integración RHadoop es necesario seguir los siguientes pasos:

- instalación de librerías de R:

    1. rJava: Interfaz de bajo nivel similar a Java

    2. RJSONIO: Librería que permite el manejo de datos desde formato JSON (JavaScript Object Notation).

    3. itertools: Conjunto de herramientas para la creación de iteraciones.

    4. digest: Implementación de una función digest() para el manejo arbitrario de objetos R.

    5. Rcpp: Paquete para la integración de R y C++.

    6. httr: Librería para el manejo de protocolos HTTP.

    7. functional: Conjunto de herramientas para el manejo de datos.

    8. devtools: Colección de paquetes para el desarrollo de herramientas.

    9. plyr: Herramientas para el manejo de datos para su simplificación.

    10. reshape2: Manejo avanzado para problemas asociados a operaciones con matrices.

    Es posible instalar estos paquetes en R, utilizando la siguiente función en R:

    >install.packages(c("rJava", "RJSONIO, "itertools", "digest",

"Rcpp", "httr", "functional", "devtools", "plyr", "reshape2"))

- Configuración de variables de entorno: para esto es posible utilizar los siguiente comandos en R:

  ##Setting HADOOP_CMD

  >Sys.setenv(HADOOP_CMD="/usr/local/hadoop/bin/hadoop")

  ##Setting up HADOOP_STREAMING

  >Sys.setenv(HADOOP_STREAMING="/usr/local/hadoop/contr ibstreaming/ hadoop-streaming-1.0.3.jar")

- Descarga e instalación de librerías de RHadoop

  1. Es posible descargar las librerías rmr, rhdfs y rmr desde los repositorios de Revolution Analytics en GitHub: https://github.com/RevolutionAnalytics/RHadoop

     rmr: [rmr-2.2.2.tar.gz]

     rhdfs: [rhdfs-1.6.0.tar.gz]

     rhbase: [rhbase-1.2.0.tar.gz]

  2. Utilizando la terminal de Ubuntu, es posible instalar estos paquetes de la siguiente forma:

     rmr: **R CMD INSTALL rmr-2.2.2.tar.gz**

     rhdfs: **R CMD INSTALL rmr-2.2.2.tar.gz**

     rhbase: **R CMD INSTALL rhbase-1.2.0.tar.gz**

Prajapati(2013) describe un ejemplo de una simulación, para la utilización de RHadoop, muy útil para comprender su funcionamiento. En este ejemplo se realiza una operación para obtener el doble de los valores de un vector de datos. La codificación de este ejemplo se describe a continuación de la siguiente forma:

```
# Es necesario iniciar R
$R

#limpiar memoria de objetos rm(list=ls(all=TRUE))

#Configuración de variables de entorno
>Sys.setenv(HADOOP_HOME="/opt/hadoop/hadoop")
>Sys.setenv(HADOOP_CMD="$HADOOP_HOME/bin/hadoop")
>Sys.setenv(HADOOP_STREAMING="/opt/hadoop/hadoop/share
/hadoop/tools/lib/hadoop-streaming-2.7.2.jar")

>library(rmr2)
>library(rhdfs)
>hdfs.init()

#reparte un vector en el dfs (disco duro de hadoop)
>a= 1:100000
>ints = to.dfs(a)

#reparte un cálculo (v,2*v) en hadoop (procesador de hadoop)
>calc = mapreduce(input = ints,map = function(k, v) cbind(v, 2*v))

#recoge el cálculo de cada uno de los nodos
>f=from.dfs(calc)
>f
```

3.6 Sistemas de Recomendación

Los sistemas de recomendación son herramientas y técnicas que intentan predecir el comportamiento de un usuario y establecer sugerencias, de acuerdo a la preferencia que han emitido con anterioridad respecto de un ítem. En la actualidad estos ítem pueden ser productos o servicios ofrecidos por una empresa, donde estas se enfocan en los procesos de tomas de decisiones como por ejemplo preferencias de música , recomendación de libros o noticias, etc. Ítem es el termino utilizado en la teoría de sistemas de recomendaciones para referirse a aquello que el sistema recomienda a los usuarios (Ricci, Rokach, Shapira & Kantor, 2011).

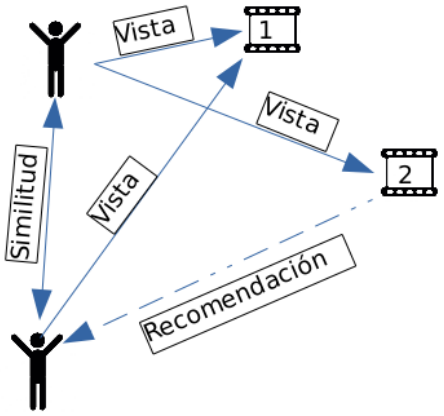

Figura 3.13: Ejemplo básico de una recomendación

Los sistemas de recomendaciones, nacen a partir de la idea de automatización de las opiniones que pudiese dar algún conocido, amigo o familiar respecto de algo de lo que fueron consumidores. Por ejemplo, a la hora de adquirir un libro, si la opinión de un amigo es positiva, es más fácil confiar de que este puede ser del gusto del usuario y este mismo buscara conseguirlo. En estas automatizaciones de las recomendaciones, el enfoque se centra en recomendar uno o varios ítem entre usuarios que tienen gustos similares, este enfoque se denomina filtrado colaborativo. De esta forma y a medida que los sitios web de comercio electrónico crecían, aumentando la posibilidad de adquirir nuevos productos, surge la necesidad de realizar recomendaciones de acuerdo a los distintos perfiles de usuarios. El creci-

miento explosivo de variedad de productos y servicios disponibles en la web frecuentemente llevaban a los usuarios a confusiones y a tomar malas decisiones, y en lugar de producir un beneficio comenzó a afectar el bienestar de los usuarios. Con respecto a esto, los sistemas de recomendaciones han demostrado ser un importante avance en términos de amortiguar la sobrecarga de información, donde el usuario puede examinar las recomendaciones, aceptarlas o desecharlas. Esto permite proporcionar una retroalimentación con el sistema pudiendo ser almacenadas en una base de datos, y ser utilizadas para generar nuevas recomendaciones en las próximas interacciones del usuario (Ricci et al., 2011).

El desarrollo de los sistemas de recomendaciones es relativamente nuevo comparado al uso de otras herramientas. Estos tienen sus inicios como área de investigación independiente en los años 90, donde el interés en los últimos años ha aumentado de manera significativa (Ricci et al., 2011).

El primer sistema de recomendación fue llamado Tapestry, el cual fue diseñado con el fin de recomendar documentos extraídos de un grupo de noticias a una colección de usuarios. En éste se aprovechaba la colaboración social, para evitar que los usuarios fuesen inundados por una gran cantidad de documentos que pudiesen no ser de su interés. En la actualidad el área de los sistemas de recomendaciones esta activa con conferencias ACM (Association for Computing Machinery) en varios lugares en el mundo, siendo parte de subdisciplinas en estadística, aprendizaje de maquinas, minería de datos y recuperación de información. La aplicaciones de estas técnicas han sido aplicadas en diversos ámbitos como recomendaciones de música, libros y muchos otros (Melville & Sindhwani, 2010).

Este análisis busca proponer un modelo que permita minimizar el error al predecir las calificaciones de usuarios suscritos a la página de *Netflix*, abordando el desafío *Netflix Prize*. A continuación se expone un modelo que permitiría realizar predicciones de calificaciones de los usuarios para las distintas películas.

3.6.1 Modelo para sistemas de recomendación

En esta sección se describe uno de los modelos probados por el equipo "BellKor's Pragmatic Chaos", para ganar el gran premio de 1 millón de dolares del desafío *Netflix Prize*. Algunas características presentadas a continuación son necesarias para comprender la notación utilizada.

De acuerdo a lo que afirma Koren (2009), es necesario reservar la asignación de letras especiales, para poder distinguir usuarios de las películas. Para los usuarios son asignadas las letras u, v y en el caso de las películas i, j. La calificación de un usuario u a una película i es denotado por $r_{u,i}$, en que los valores posibles son entre 1 y 5 estrellas.

En el desarrollo de este trabajo los datos originales del conjunto de entrenamiento del *Netflix Prize*, serán divididos en dos subconjuntos de entrenamiento y prueba, de forma aleatoria. Los pares (u,i) para los cuales $r_{u,i}$ son conocidos, se almacenan en el conjunto de entrenamiento denotado por $K = \{(u, i) \mid r_{u,i} \text{ es conocido}\}$. En el conjunto de entrenamiento K, el modelo realiza aprendizaje, ajustando estas calificaciones para poder realizar las predicciones de los datos que se encuentran en el conjunto de prueba. El conjunto de prueba se define por aquellos pares (u, i) con calificación $r_{u,i}$ omitidos desde el conjunto de entrenamiento original, suponiendo esta cali-ficación como inexistente, para poder predecirla. El largo del conjunto de prueba es denotado por S. $R(i)$ representa el largo del vector de usuarios que calificaron una película i en el conjunto de entrenamiento. Cada usuario u es asociado a un conjunto de ítems (películas), en que $R(u)$ representa el largo del vector de películas calificadas por un usuario u en el conjunto de entrenamiento (Koren, 2009).

Irizarry & Hicks (2016), denotan un modelo de recomendaciones de la siguiente manera:

$$Y_{ui} = \mu + b_u + b_i + \varepsilon, \tag{2}$$

donde:

- μ: Media global de todos las calificaciones disponibles.
- b_u: Desviación observada de un usuario u.
- b_i: Desviación observada de un ítem i.
- ε: Representa un termino de error aleatorio asociado.

Un estimador que es útil para realizar predicciones de los rating desconocidos, es denotado por Koren (2009), con una variable de respuesta b_{ui}, descritos en la siguiente ecuación:

$$b_{ui} = \hat{\mu} + \hat{b}_u + \hat{b}_i \tag{3}$$

Para la estimación de los parámetros μ, b_i y b_u, Koren (2009) denota las siguientes fórmulas:

$$\hat{\mu} = \frac{1}{K} \sum_{(u,i) \in K}^{K} r_{ui} \tag{4}$$

$$\hat{b}_i(\lambda_1) = \frac{1}{(\lambda_1 + \mid R(i) \mid)} \sum_{u \in R(i)} (r_{ui} - \hat{\mu}) \tag{5}$$

$$\hat{b}_u(\lambda_2) = \frac{1}{(\lambda_2 + \mid R(u) \mid)} \sum_{i \in R(u)} (r_{ui} - \hat{\mu} - \hat{b}_i) \tag{6}$$

En este caso los parámetros λ_1 y λ_2 se definen como parámetros de regulari-zación, donde su valor se determina mediante ensayos del modelo validado en el conjunto de prueba.

Un ejemplo de una aplicación simplificada del modelo anterior, puede ser intentar predecir la calificación del usuario 5 a la película 1 y del usuario

1 a la película 3 (ver Tabla 3.3). Para esto se utilizan los datos de la Tabla 3.2.

ID user	rating	movie
1	3	2
5	2	3
2	3	1
3	4	1
1	4	1
5	5	4
2	1	2
5	2	2
3	4	4
1	2	4
3	3	3
1	4	3
5	4	1

Tabla 3.2: Ejemplo datos *Netflix Prize*

A partir de la Tabla 3.2 se extraen de forma aleatoria los conjuntos de entrenamiento y prueba. En el conjunto de entrenamiento se construye el modelo para predecir las calificaciones en el conjunto de prueba.

user	movie1	movie2	movie3	movie4
1	4	4	?	2
2	3	1		
3	4		3	4
5	?	2	2	5

Tabla 3.3: Ejemplo conjunto de entrenamiento

user	movie1	movie3
1		4
5	4	

Tabla 3.4: Ejemplo conjunto de prueba

Los valores que aparecen incógnitos con el símbolo de interrogación (?) en la Tabla 3.3 esta disponibles en el conjunto de prueba (ver Tabla 3.4), entonces a partir de los datos que son conocidos en el conjunto de entrenamiento, se utiliza el modelo planteado anteriormente (2). A continuación se realiza el desarrollo para este ejemplo:

- Es necesario estimar el valor de μ, para esto se emplean la formula anteriormente descrita en (3) y todas las calificaciones disponibles en el conjunto de entrenamiento.

$$\hat{\mu} = \frac{4+3+4+1+2+2+4+5}{8} = 3,1$$

- Se debe realizar la asignación de valores a los parámetros de regularización. Para este ejemplo se consideran los mismos valores definidos por Koren (2009), en que: λ_1=25 y λ_2=10.

- Es necesario calcular el valor de b_i, en que i=$\{1, 3\}$ considerando las calificaciones realizadas por usuarios activos en las películas 1 y 3.

  $\hat{b}_{i=1}= \frac{1}{25+3}[(4-3,1)+(3-3,1)+(4-3,1)]$=0,832

  $\hat{b}_{i=3}= \frac{1}{25+2}[(3-3,1)+(2-3,1)]$=-0,044

- El valor de $\hat{b}_u$ depende de $\hat{b}_i$, por lo que el paso previo era necesario para realizar este cálculo. En el primer caso u=5, y son útiles todos las calificaciones realizadas por el usuario 5 en el conjunto de entrenamiento.

$\hat{b}_{u=5}= \frac{1}{10+3}[(2-3,1-0,026)+(2-3,1+0,026)+(5-3,1-0,026)]$

$\hat{b}_{u=5} = -0,025$

- Por lo tanto la estimación para calificación del usuario 5 en la película 1, considerando lo anterior se puede realizar utilizando la formula (2) de la siguiente forma

  $\hat{b}_{51} = \hat{\mu}+\hat{b}_{u=5}+\hat{b}_{i=1}$

  $\hat{b}_{51} = 3{,}1+\text{-}0{,}025+ 0{,}832= 3{,}907$

  En que el valor 3,907 es el valor predicho de la calificación del usuario 5 a la película 1.

- El cálculo de b_u en que u=1 se realiza de igual forma utilizando la formula (5):

  $\hat{b}_{u=1} = \frac{1}{10+3}[(4-3,1-0,044)+(4-3,1-0,044)+(4-3,1-0,044)]$

  $\hat{b}_{u=1} = 0,198$

- Entonces el cálculo para el usuario 1 en la película 3 puede realizarse utilizando la formula (2)

  $\hat{b}_{1,3} = \hat{\mu} + b_{u=1} + b_{i=3}$

  $\hat{b}_{1,3} = 3,1 + 0,198 - 0,044 = 3,254$

  En que el valor 3,907 es el valor predicho de la calificación del usuario 1 a la película 3.

- Para el cálculo del *RMSE* y comprobar las estimaciones del modelo de recomendaciones es posible utilizar la formula (1) disponible en la

sección 2.1. En este ejemplo el cálculo del *RMSE* es el siguiente:

$$RMSE = \sqrt{\frac{1}{2}(r_{1,3} - b_{1,3})^2 + (r_{5,1} - b_{5,1})^2}$$

$$RMSE = \sqrt{\frac{1}{2}(4 - 3,254)^2 + (4 - 3,907)^2}$$

$$RMSE = 0,532$$

3.6.2 Proceso de recomendación

En este estudio se plantea que mediante los valores predichos, sea posible efectuar una recomendación. Los datos correspondientes a los valores predichos y los *rating*, son binarizados, en que un valor mayor a 3 es denotado por 1 y en caso contrario 0. En el caso de los valores predichos, los ítem denotados con 1, se consideran como una recomendación. El siguiente ejemplo denotado en las Tablas 3.5 y 3.6 permite ilustrar esta idea:

Tabla 3.5: Ejemplos rating v/s valores predichos

rating	valores predichos
3	3,2
4	3,5
5	4,6
3	2,1
2	4
4	3,2
5	3,1
1	3,5
2	4,4

Tabla 3.6: Ejemplos rating y valores predichos binarizados

rating	valores predichos
1	1
1	1
1	1
1	0
0	1
1	1
1	1
0	1
0	1

En este caso es posible realizar el cálculo de la matriz de confusión para verificar las recomendaciones realizadas. Para efectos de comprender esta idea, es necesario abordar las siguientes definiciones:

- Verdaderos positivos(VP): Son aquellos casos en que es realizada una recomendación y esta es del gusto del usuario.

- Verdaderos negativos(VN): Los casos en que la recomendación no fue realizada y esta no es del gusto del usuario.

- Falsos positivos(FP): Se refiere a los casos en que una recomendación fue efectuada y esta no es del gusto del usuario.

- Falsos Negativos(FN): Es el caso en que un usuario gusta de una película, pero no es realizada la recomendación.

Tabla 3.7: Matriz de confusión para sistemas de recomendaciones

	Le gusta	No es del gusto
Recomendación	Verdaderos positivos	Falsos positivos
No hay recomendación	Falsos negativo	Verdaderos Negativos

Según Gorakala & Usuelli (2015) es posible definir las siguientes medidas de

diagnóstico:

- Sensitividad: Probabilidad de recomendar correctamente.

$$Sensitividad = \frac{VP}{VP + FN} \tag{7}$$

- Especificidad: Es la probabilidad de no realizar recomendaciones correctamente.

$$Especificidad = \frac{VN}{VN + FP} \tag{8}$$

- Precisión: Probabilidad de que las predicciones clasificadas como verdaderas, sean correctamente clasificadas.

$$Precision = \frac{VP}{VP + FP} \tag{9}$$

Considerando los datos de la Tabla 3.6 es posible generar la matriz de confusión de la siguiente forma:

Tabla 3.8: Matriz de confusión para sistemas de recomendaciones

	Le gusta	No es del gusto
Recomendación	5	3
No hay recomendación	1	0

El cálculo de las medidas de sensibilidad, precisión y especificidad para este ejemplo, son presentados a continuación utilizando las fórmulas anteriores:

Sensitividad=$\frac{5}{5+1}$=0,8333

Precisión= $\frac{5}{5+3}$= 0,625

Especificidad=0

Capítulo 4

Metodología

En el presente capítulo se describe la metodología que fundamenta este trabajo, utilizando las distintas técnicas abordadas en el marco teórico. Son aplicadas herramientas tales como la conexión y manejo del cluster del IDEUV, utilización de R y la integración RHadoop en la codificación del algoritmo de sistemas de recomendaciones.

4.1 Conexión al *cluster* del IDEUV

Para la carga de los datos y su respectivo análisis, es necesaria la implementación y utilización de un *cluster* de computadores, que permitan llevar este estudio. La conexión al *cluster* del instituto de estadística se realiza mediante la terminal de Ubuntu.

Existen dos formas de conexión a este medio descritas a continuación:

Conexión Intranet:

- Para la conexión dentro de la Universidad de Valparaíso vía Intranet,

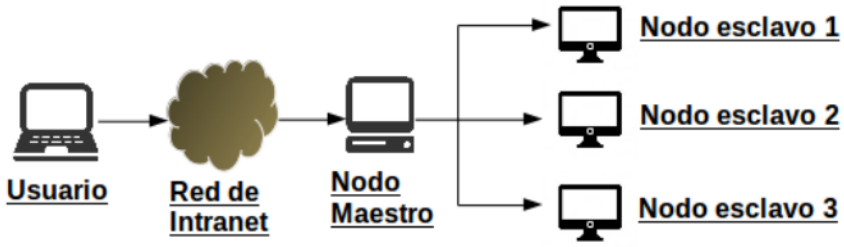

Figura 4.1: Diagrama conexión cluster IDEUV

desde un computador registrado por el admi-nistrador de la red, es posible conectarse utilizando el siguiente
comando:

$ ssh hadoop@<ip_servidor_del_cluster>

Una vez aquí se puede llamar a R desde la terminal
$ R

- Conexión fuera de la universidad usando Internet:

  En el caso de establecer conexión a Internet, esto se puede realizar de la siguiente forma:

  # Conexión al servidor del IDEUV
  $ ssh ttapia@<ip_servidor_del_IDEUV>

  # Conexión al cluster del IDEUV
  $ ssh hadoop@<ip_servidor_del_cluster>

  Una vez aquí se puede llamar a R desde el *cluster* de Hadoop
  $ R

Las características de esta arquitectura pueden ser obtenidas utilizando dis-

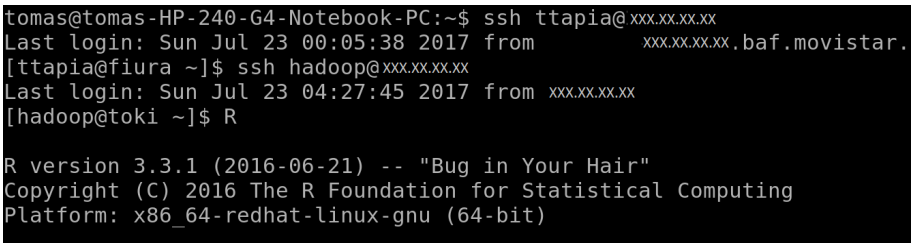

Figura 4.2: Conexión al cluster del IDEUV en Internet

tintos comandos vía terminal de Ubuntu, como se describe a continuación:

número de procesadores presentes en el cluster
$nproc –all

# modelo de CPU de equipos que componen el cluster
$ cat /proc/cpuinfo | grep `"model name"`

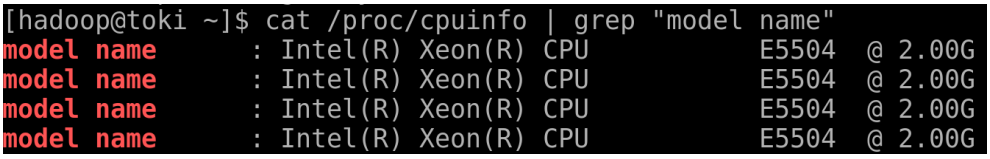

Figura 4.3: Modelos de CPU del *cluster* del IDEUV

# número de nucleos o cores
$ cat /proc/cpuinfo | grep `"cpu cores"`

4.2 Descarga de los datos

Los datos de entrenamiento de la competencia del *Netflix Prize*, útiles para este estudio, es posible descargarlos, utilizando el siguiente comando vía terminal en Ubuntu desde los servidores de archive.org/:

$ wget https://archive.org/download/nf_prize_dataset.tar/nf_prize_dataset.tar.gz

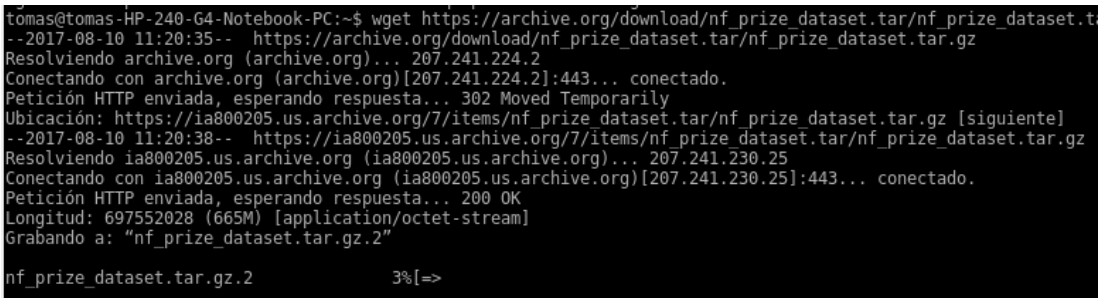

Figura 4.4: Descarga de Archivos del *Netflix Prize*

# Este comando descomprime el archivo descargado anteriormente
$ tar -xzvf nf_prize_dataset.tar.gz

Después del termino de la descarga y descompresión se habrá generado un directorio con 17.000 archivos de texto de 2,1 GB, en la carpeta \home\hadoop.

4.3 Administración de datos

En esta sección se describe la fase relacionada con a la administración de los datos. Los siguientes pasos permiten unificar los 17.000 archivos de texto disponibles del conjunto de entrenamiento, dejando un solo archivo de texto de filas m=100.480.507 y columnas n=3.

Las variables utilizadas para esta investigación serán las siguientes:

- Id de usuario
- Rating otorgado por un usuario a una película
- Identificador de película

Los pasos enumerados a continuación permiten realizar el proceso de administración de los datos utilizados en este análisis:

1. El Numero identificador de película: Un problema que podría parecer bastante trivial y tal vez insignificante para el analista de datos, es que en el caso de los datos del conjunto de entrenamiento del *Netflix Prize*, cada uno de los archivos de texto descargados anteriormente, existe un numero identificador correspondiente a la película en la parte superior izquierda de este.

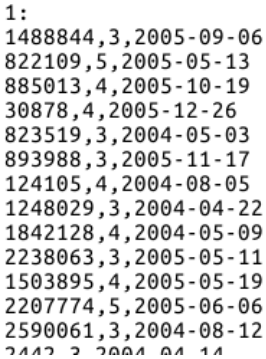

Figura 4.5: Archivo Nº1 *Netflix Prize*.

   En el intento de cargar los datos al software R, este arrojaría un error correspondiente a que esta carga de datos, no es posible realizar debido a este identificador. Una solución que pudiese parecer bastante obvia, sería intentar borrar cada uno de estos identificadores manualmente, pero esto podría ser una solución efectiva, pero bastante lenta, que podría durar meses o tal vez años. Afortunadamente este proceso puede realizarse automáticamente utilizando el siguiente comando en la terminal en Ubuntu:

   $ for FILE in *; do echo `"$(tail -n +2 $FILE)"` > $FILE; done

   Este comando permite borrar la primera fila de todos los archivos de texto almacenados en un directorio.

2. Numero identificador como variable: En el paso anterior el numero

identificador fue eliminado de la parte superior de cada uno de los archivos de texto, pero ahora es necesario colocar estos identificadores como una columna adicional en cada uno de estos. Este procedimiento puede realizarse utilizando el siguiente comando:

$ for f in mv_*; do sed -i "s/$/,$f/" $f; done

Este comando registra el nombre del archivo en cada fila, asociando este a un registro de usuario (ver Figura 4.6).

```
1488844,3,2005-09-06,mv_0000001.txt
822109,5,2005-05-13,mv_0000001.txt
885013,4,2005-10-19,mv_0000001.txt
30878,4,2005-12-26,mv_0000001.txt
823519,3,2004-05-03,mv_0000001.txt
893988,3,2005-11-17,mv_0000001.txt
124105,4,2004-08-05,mv_0000001.txt
1248029,3,2004-04-22,mv_0000001.txt
1842128,4,2004-05-09,mv_0000001.txt
2238063,3,2005-05-11,mv_0000001.txt
1503895,4,2005-05-19,mv_0000001.txt
2207774,5,2005-06-06,mv_0000001.txt
```

Figura 4.6: Primer Archivo de texto editado.

3. Unificar todos los archivos de texto en uno solo gran archivo: Este procedimiento se puede lograr utilizando el siguiente comando en la terminal de Ubuntu:

   $ cat mv_> mvt.txt

   En este caso identifica todos los archivos dentro de un directorio que contengan el prefijo "mv_" y los apila en un solo nuevo archivo de texto llamado "mvt.txt".

4. Suprimir columna de Fechas: En este análisis no se considera la variable de fechas, dado que el modelo implementado no requiere la utilización de esta columna. Para crear un nuevo archivo considerando solo las columnas de Id de usuario, Id de película y rating, se puede utilizar el siguiente comando vía terminal de Ubuntu:

$ awk -F“,” ‘print $1,$2,$4’ mvt.txt > mv.txt

En este caso desde el archivo "mvt.txt" creado en el paso 3, se genera un nuevo archivo llamado “mv.txt” (ver Figura 4,7).

```
1488844 3 mv_0000001.txt
822109 5 mv_0000001.txt
885013 4 mv_0000001.txt
30878 4 mv_0000001.txt
823519 3 mv_0000001.txt
893988 3 mv_0000001.txt
124105 4 mv_0000001.txt
1248029 3 mv_0000001.txt
1842128 4 mv_0000001.txt
2238063 3 mv_0000001.txt
1503895 4 mv_0000001.txt
2207774 5 mv_0000001.txt
```

Figura 4.7: Archivo "mv.txt" editado

5. Extraer todos los elementos no numéricos de la columna 3: En este caso se utiliza R para extraer “mv_”, y que el Id de película quede enumerado de 1 a 17.770. Esto se puede realizar utilizando la siguiente codificación:

```
# Leer el archivo mv.txt en R
> mvtotal <- read.csv("mv.txt", header=FALSE,sep=",")
> attach(mvtotal)

# Suprimir mv_ de la columna 3 del archivo mv.txt
> group<-data.frame(group=V3)
> group$group.no.e <- gsub("mv_", "", group$group)
> attach(group)
> a<-data.frame(group.no.e)
> names(a) <-"l"
> attach(a)
> a2<-data.frame(sub('\\..*', '', l))
> names(a2) <-"c"
> attach(a2)
```

Convertir de cadena a numérico el vector c
> g<-as.numeric(as.character(c))

Reemplazar la antigua variable de película por la que fue editada
> s<-cbind(mvtotal[, -3],g)
> attach(s)

Renombrar variables
> colnames(s)<- c("id","rating","movie")

Escribir en formato texto el nuevo archivo
>write.table(s, "mvtotal.txt", sep="\t")

Ejecutando el código anterior se obtiene un archivo de texto llamado "mvtotal.txt". Este archivo es posible visualizar sus dimensiones utilizando los siguientes comandos vía terminal de Ubuntu:

# Este comando permite ver cuantas lineas tiene un archivo de texto
$ wc -l mvtotal.txt
100480507 mvtotal.txt

6. Conjuntos de Entrenamiento y prueba: Para probar los distintos modelos para el sistema recomendación es necesario particionar este archivo en un conjunto de entrenamiento (70%) y prueba (30%). En el conjunto de entrenamiento se construyen los modelos y estos son validados en el conjunto de prueba. Esta división puede realizarse utilizando el siguiente comando en la terminal de Ubuntu:

$ shuf mvtotal.txt | split -l $(($(wc -l <mvtotal.txt) * 70/100))–addi tional-suffix=.txt

De la misma forma, es posible visualizar la cantidad de filas de estos archivos en el directorio que son almacenados, utilizando el siguiente comando en la terminal de Ubuntu:

$ wc -l test.txt
20096102 test.txt

$ wc -l train.txt
80384405 train.txt

También es posible visualizar el tamaño en el disco duro de todos los archivos alojados en un directorio de la siguiente forma:

$ du -sh *
1.5G mvtotal.txt
307M test.txt
1.2G train.txt

7. Fragmentación del conjunto de entrenamiento

   En el paso anterior se realiza la partición de los datos en conjunto de prueba y entrenamiento, pero ahora es necesario dividir este ultimo en varias partes. Ese procedimiento permite la carga total de los datos, dado que R no puede cargar un archivo tan grande de inmediato, pero si puede cargar progresivamente varios archivos de menor tamaño. El comando utilizado en este caso es el siguiente:

   $ split -dl 10000000 –additional-suffix=.txt train.txt

   Este comando divide el archivo “train.txt” en 8 archivos de textos de 10.000.000 lineas y uno de 384.405.

4.4 Sistemas de recomendaciones utilizando el cluster del IDEUV

En esta sección son probados dos modelos para la aplicación e implementación de un algoritmo de sistemas de recomendación útiles para esta investigación los cuales serán implementados en el *cluster* del IDEUV. En el primer modelo, se utilizara la integración RHadoop, describiendo los pasos efectuados en la codificación, utilizando un modelo ingenuo para la estimación de los *rating*. El segundo modelo es el modelo propuesto en la sección 2.6 planteado por Koren (2009), utilizando un algoritmo en R.

En los siguientes casos empleados, el conjunto de entrenamiento corresponde a un 70%, mientras que el conjunto de prueba a un 30%.

4.4.1 Modelo basado en la media

En un primer ensayo, es aplicado un modelo simple, basado en la media, propuesto por Irizarry & Hicks (2016), en un ejemplo de sistemas de recomendaciones en el sofware R. Este primer modelo es denominado modelo ingenuo. Es realizada una estimación del rating, en que todos estos, corresponden al valor de la media. El modelo es detallado a continuación:

$$Y_{u,i} = \mu + \varepsilon, \tag{10}$$

Donde:

- $Y_{u,i}$ Variable de respuesta de un modelo de recomendaciones.
- μ: Es el promedio de todas las calificaciones disponibles realizadas por usuarios activos en el conjunto de entrenamiento del *Netflix Prize*.

- ε: Corresponde a un error aleatorio asociado al modelo

El modelo estimado para este caso es el siguiente:

$$b_{u,i} = \hat{\mu}, \tag{11}$$

- $\hat{\mu}$ corresponde al promedio calculado en el conjunto de entrenamiento, descrito en la ecuación (4) de la sección 2.6.

4.4.1.1 Codificación en RHadoop

Para este procedimiento fue necesario dividir el conjunto de entrenamiento en nueve subconjuntos. (ver Figura 4.8)

```
hadoop@toki copiadeseguridad]$ ls
vtotal.txt  train.txt     x01train.txt  x03train.txt  x05train.txt  x07train.txt
est.txt     x00train.txt  x02train.txt  x04train.txt  x06train.txt  x08train.txt
hadoop@toki copiadeseguridad]$
```

Figura 4.8: Vista previa de archivos útiles para el análisis

A continuación se realiza la codificación de la carga de los datos del *Netflix Prize* con su análisis y comentarios de su aplicación en el cluster RHadoop.

#Conexión al servidor del IDEUV
$ ssh ttapia@<ip_servidor_del_IDEUV>

#Conexión al cluster hadoop del IDEUV
$ ssh hadoop@<ip_servidor_del_cluster>

#Llamado de la consola de R
$ R

#configuración de Variables de entorno

```
>rm(list=ls(all=TRUE))
>Sys.setenv(HADOOP_HOME="/opt/hadoop/hadoop")
>Sys.setenv(HADOOP_CMD="$HADOOP_HOME/bin/hadoop")
>Sys.setenv(HADOOP_STREAMING="/opt/hadoop/hadoop/share
/hadoop/tools/lib/hadoop-streaming-2.7.2.jar")

# Carga de librería útiles para el análisis
>library(rmr2)
>library(rhdfs)
>library(ggplot2)
>library(readr)
>library(tidyr)
>library(dplyr)
>library(curl)
>hdfs.init()

#cargando primeras 5 partes.
>mv1<-read.csv("x00train.txt", header=FALSE,sep="")
>mv2<-read.csv("x01train.txt", header=FALSE,sep="")

>a<-rbind(mv1,mv2)
>rm(mv1,mv2)
>mv3<-read.csv("x02train.txt", header=FALSE,sep="")
>a<-rbind(a,mv3)
>rm(mv3)
>mv4<-read.csv("x03train.txt"", header=FALSE,sep="")
>a<-rbind(a,mv4)
>rm(mv4)
>mv5<- read.csv("x04train.txt", header=FALSE,sep="")
>mvtrain1<-rbind(a,mv5)
>rm(mv5)

attach(mvtrain1)
# Carga de los datos al sistema de archivos distribuido
>ints <- to.dfs(mvtrain1$V2)
```

#Procesamiento en MapReduce
>countdata = mapreduce(input = ints,map = function(k, v)length(v))
>sumdata = mapreduce(input = ints,map = function(k, v) sum(v))

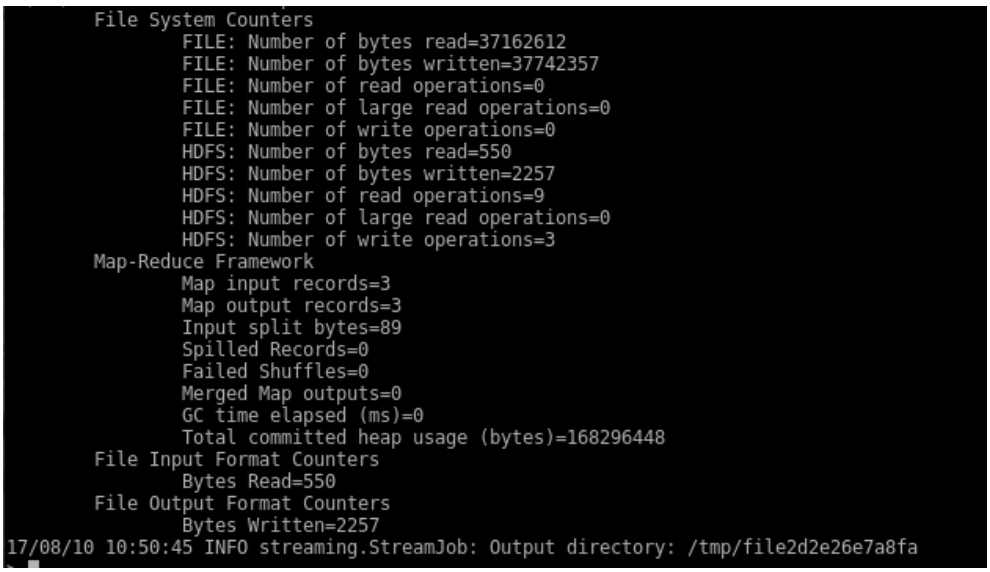

Figura 4.9: Ventana de tareas del procesamiento en MapReduce

Recoge el cálculo de los nodos
>f1<-from.dfs(countdata)
>f2<-from.dfs(sumdata)

#largo de primer conjunto de datos
>calc <- sum(as.numeric(unlist(f1)))

#suma del primer conjunto de datos
>sum1<-sum(as.numeric(unlist(f2)))
>calc
>sum1
>rm(mvtrain1)

```
#cargando segunda parte de los datos
>mv6<-read.csv("x05train.txt", header=FALSE,sep="")
>mv7<-read.csv("x06train.txt", header=FALSE,sep="")
>a<-rbind(mv6,mv7)
>rm(mv7,mv6)
>mv8<-read.csv("x07train.txt", header=FALSE,sep="")
>a<-rbind(a,mv8)
>rm(mv8)
>mv9<-read.csv("x08train.txt", header=FALSE,sep="")
>mvtrain2<-rbind(a,mv9)
>rm(mv9,a)

>attach(mvtrain2)
# Carga de los datos al sistema de archivos distribuido
>ints2 <- to.dfs(mvtrain2$V2)

#Procesamiento en MapReduce
>countdata2 = mapreduce(input = ints2,map = function(k, v)length(v))
>sumdata2 = mapreduce(input = ints2,map = function(k, v) sum(v))

# Recoge el cálculo de los nodos
>f12<-from.dfs(countdata2)
>f22<-from.dfs(sumdata2)



#largo de segundo conjunto de datos
>calc2<-sum(as.numeric(unlist(f12)))

# suma del segundo conjunto de datos
>sum2<-sum(as.numeric(unlist(f22)))
>calc2
>sum2

#total de registros
>totalregistrostrain<-calc+calc2
```

```
>totalregistrostrain

# Valor media general de rating en el conjunto de entrenamiento
average_rating<-(sum1+sum2)/totalregistros

#Carga del conjunto de prueba
>mvtest<-read.csv("test.txt", header=FALSE,sep="")

# Nombrando variables
>colnames(mvtest)<-c("id","rating","movie")
>nrowtest<-20096102

#Codificación de modelo
predictions <- rep(average_rating,nrowtest)

#Función cálculo RMSE
>RMSE <-function(true_ratings, predicted_ratings) {

sqrt(mean((true_ratings - predicted_ratings)^ 2))

}.

# RMSE con respecto al conjunto de prueba
> RMSE <- print(RMSE(mvtest$rating, predictions))
```

4.4.2 Modelo propuesto de sistemas de Recomendaciones

En la Sección 2.6, se abordó un modelo útil para un sistemas de recomendaciones, en que se estudian los conceptos básicos y una aplicación simplificada, utilizando un conjunto de datos de ejemplo. En esta sección se aborda la aplicación de este modelo en el conjunto de datos de *Netflix Prize*, utilizando R en el *cluster* del IDEUV. A continuación es descrita la rutina del código

para la carga de los datos e implementación del modelo propuesto.

```
#Conexión al servidor fiura del IDEUV
$ ssh ttapia@<ip_servidor_del_IDEUV>

#Conexión al cluster toki hadoop del IDEUV
$ ssh hadoop@<ip_servidor_del_cluster>

#Llamado de la consola de R
$ R

#configuración de Variables de entorno
>rm(list=ls(all=TRUE))
>Sys.setenv(HADOOP_HOME="/opt/hadoop/hadoop")
>Sys.setenv(HADOOP_CMD="$HADOOP_HOME/bin/hadoop")
>Sys.setenv(HADOOP_STREAMING="/opt/hadoop/hadoop/share/hado
op/tools/lib/hadoop-streaming-2.7.2.jar")

# Carga de librerías útiles para el análisis
>library(rmr2)
>library(rhdfs)
>library(ggplot2)
>library(readr)
>library(tidyr)
>library(dplyr)
>library(curl)
>hdfs.init()

#Carga de los datos
>mv1<-read.csv("x00train.txt", header=FALSE,sep="")
>mv2<-read.csv("x01train.txt", header=FALSE,sep="")
>mv3<-read.csv("x02train.txt", header=FALSE,sep="")

>mvtotal<-rbind(mv1,mv2,mv3)
>rm(mv1,mv2,mv3)
```

```
>mv4<-read.csv("x03train.txt", header=FALSE,sep="")
>mvtotal<-rbind(mv4,mvtotal)
>rm(mv4)

>mv5<-read.csv("x04train.txt", header=FALSE,sep="")
>mvtotal<-rbind(mvtotal,mv5)
>rm(mv5)

>mv6<-read.csv("x05train.txt", header=FALSE,sep="")
>mvtotal<-rbind(mvtotal,mv6)
>rm(mv6)

>mv7<-read.csv("x06train.txt", header=FALSE,sep="")
>mvtotal<-rbind(mvtotal,mv7)
>rm(mv7)

>mv8<-read.csv("x07train.txt", header=FALSE,sep="")
>mvtotal<-rbind(mvtotal,mv8)
>rm(mv8)

>mv9<-read.csv("x09train.txt", header=FALSE,sep="")
>train<-rbind(mvtotal,mv9)
>rm(mv9)

#Nombrando variables del conjunto de entrenamiento
>colnames(train)<-c("user","rating","movie")

#Función cálculo RMSE
>RMSE <-function(true_ratings, predicted_ratings) {
sqrt(mean((true_ratings - predicted_ratings)^2))
}.

#Carga del conjunto de prueba
>test<-read.csv(test.txt", header=FALSE,sep="")
```

```
#Asignando nombres a las variables
>colnames(test)<-c("user","rating","movie")

#Media de los rating en el conjunto de entrenamiento
>mu <-mean(train$rating)

#Ajustando parámetros de regularización
>lambda1 <- 5
>lambda2 <- 5

#Cálculo b_is
>movie_reg_means <- train
>group_by(movie) %>%
summarize(b_i = sum(rating - mu)/(n()+lambda1))

#Cálculo b_us
user_reg_means = train%>%
mutate(resids = rating - mu - b_i) %>% group_by(user) %>%
summarize(b_u = sum(resids)/(n()+lambda2))

> joined <- test %>%
left_join(movie_reg_means, by='movie') %>%
left_join(user_reg_means, by='user') %>%
replace_na(list(b_i=0, b_u=0))

#Modelo propuesto
>predicted_ratings = mu + joined$b_i + joined$b_u

#Cálculo RMSE
>print(RMSE(predicted_ratings, test$rating))

#Cálculo valores predichos
>joined<-data.frame(joined$user,joined$movie,joined$rating,
joined$b_i+joined$b_u+mu)
```

>colnames(joined)<-c("user","rating","movie","predict")

#Binarización rating y predict
>joined[,c(3,4)] <-ifelse(joined[,c(3,4)] >= 3,1,0)

#Son denotados por 1 los valores mayores o iguales a 3

En caso contrario los valores menores a 3 se denotan con 0

matriz de confusión
>library(caret)
>xtab<-table(joined$predict, joined$rating)
>confusionMatrix(xtab,positive ="1")

El tiempo estimado de la carga y procesamiento de los datos para la obtención de los resultados, utilizando esta codificación puede variar entre 30 y 45 minutos, dependiendo de la conexión de red.

Capítulo 5

Resultados

En este capítulo son presentados los resultados de los dos modelos codificados anteriormente, validando la metodología propuesta. Son comparados los modelos construidos con los datos cargados en el cluster del IDEUV, versus los resultados obtenidos por algunos de los equipos participantes en el *Netflix Prize*, indicando el porcentaje de mejora con respecto al algoritmo de recomendaciones de Cinematch.

Los resultados del *RMSE* del algoritmo propuesto en este estudio, puede variar según los parámetros de regularización ajustados. En este trabajo son probados algunos casos adjuntos en la Tabla 5.3.

En este caso para $Lamda1 = 5$ y $lambda2 = 5$ permiten obtener el resultado de un valor menor en el *RMSE* del modelo propuesto para este estudio, en que la matriz de confusión se muestra en la Tabla 5.1. Son denotados la cantidad de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.

Los puntajes presentados en la Tabla 5.2, estructuran los *RMSE* y las mejoras obtenidos en esta investigación comparándolos con algunos resultados de los equipos líderes de la competencia, junto con el algoritmo de Cinematch. Los reglones en amarillo representan los algoritmos presentados en

Tabla 5.1: Matriz de confusión para lambda1=5 y Lamda2=5

	Le gusta	No es del gusto
Recomendación	15565845	1805276
No hay recomendación	1582095	1142886

este análisis, en que **Modelo Propuesto** es el algoritmo presentado para batir la marca efectuada por Cinematch. Mientras que el **Modelo Ingenuo** es el algoritmo basado en la media. En el caso de los lideres de la competencia el equipo de BelKor's Pragmatic Chaos es el ganador del premio, quedando en el segundo lugar The Ensemble, este último entregando resultados con aproximadamente 20 minutos de diferencia.

Para el algoritmo del **Modelo Propuesto**, con respecto a los pasos descritos en la Sección 3.6, los datos de los rating y valores predichos deben ser binarizados, entendiendo que en este ultimo caso un valor 1 representa una recomendación.

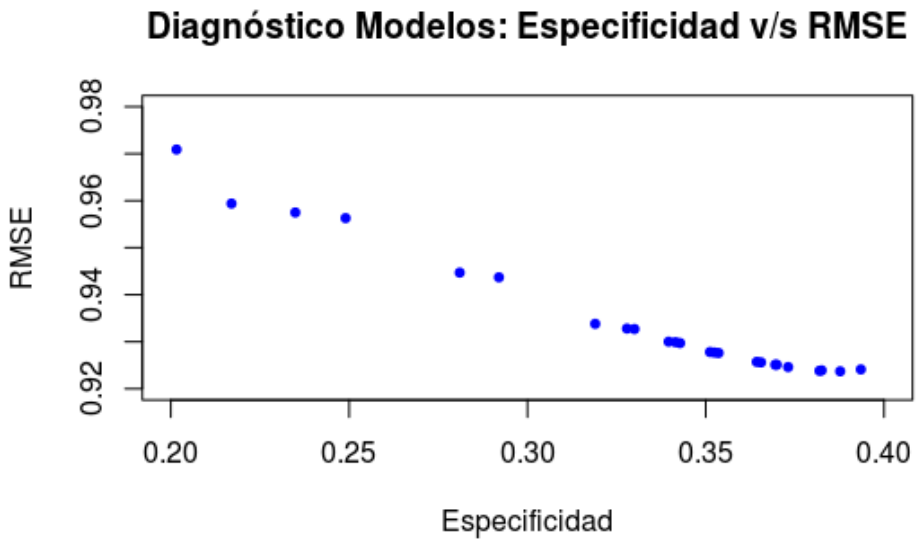

Figura 5.1: Visualización del RMSE respecto a la Especificidad, considerando datos de Tabla 5.3

Observando los gráficos se puede apreciar que el valor del $RMSE$ se mini-

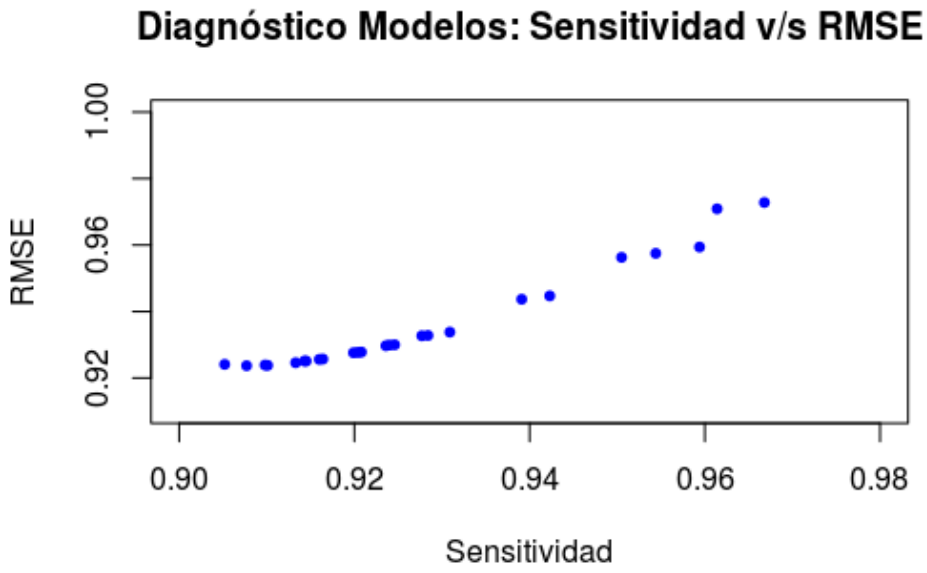

Figura 5.2: Visualización del RMSE respecto a la Sensitividad, considerando datos de Tabla 5.3

miza en general para valores bajos en las medidas de sensitividad y precisión. Distinto es el caso de la especificidad en que los valores mayores son inversamente proporcionales a los valores minimos obtenidos en el $RMSE$. Otro punto a considerar es que los valores pequeños de *Lamda* y *Alpha* se obtienen valores menores para el $RMSE$.

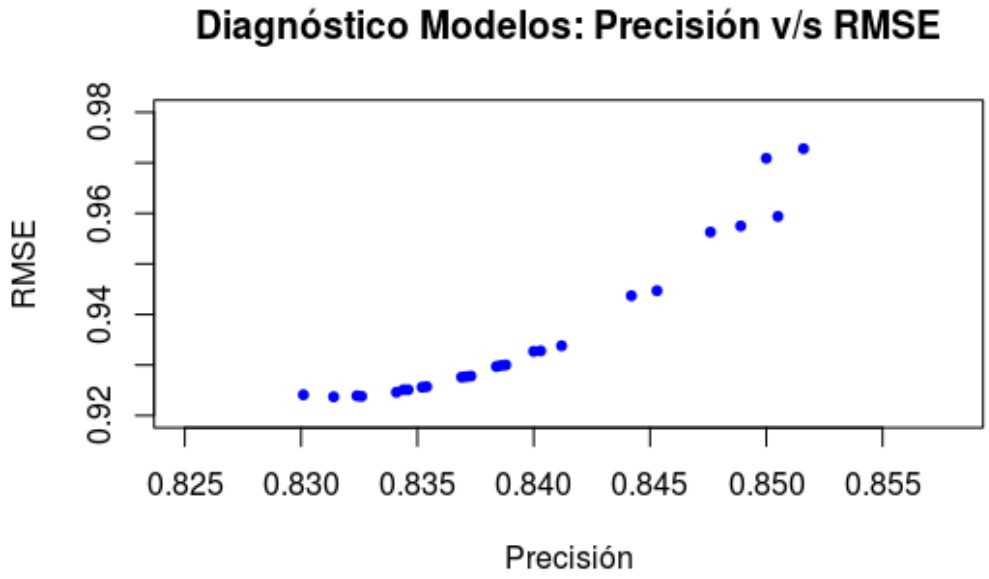

Figura 5.3: Visualización del RMSE respecto de la Precisión, considerando datos de Tabla 5.3

Tabla 5.2: Competidores *Netflix Prize*

Premio	Participantes	RMSE	Mejora
Grand Prize	BellKor's Pragmatic Chaos	0,8567	10,06%
2º lugar Grand Prize	The Ensemble	0,8567	10,06%
Progress Prize 2008	BellKor in BigChaos	0,8627	9,44%
Progress Prize 2007	KorBell	0,8723	8,43%
Trabajo Titulación 2017	Modelo Propuesto	0.9237	3,02%
Algoritmo Netflix	Cinematch	0,9525	0%
Trabajo Titulación 2017	Modelo Ingenuo	1,0851	-13,92%

Tabla 5.3: Medidas diagnósticas del modelo de sistemas de recomendaciones, mediante el ajuste de parámetros de regularización

Lamda	Alpha	RMSE	Sens.	Espec.	Prec.
0	0	0.9241	0.9052	0.3935	0.8301
5	5	0.9237	0.9077	0.3877	0.8314
10	10	0.9239	0.9098	0.3825	0.8324
25	10	0.9238	0.9101	0.3819	0.8326
10	25	0.9251	0.9143	0.3699	0.8344
25	20	0.9246	0.9133	0.3731	0.8341
20	25	0.9251	0.9145	0.3695	0.8346
30	30	0.9256	0.9160	0.3655	0.8352
50	30	0.9257	0.9164	0.3643	0.8354
30	50	0.9276	0.9199	0.3536	0.8369
50	50	0.9277	0.9204	0.3524	0.8371
50	70	0.9297	0.9236	0.3428	0.8384
70	50	0.9278	0.9208	0.3512	0.8373
70	70	0.9299	0.9240	0.3415	0.8386
100	70	0.9300	0.9246	0.3396	0.8388
70	100	0.9327	0.9277	0.3300	0.8400
100	100	0.9328	0.9284	0.3279	0.8403
100	250	0.9437	0.9391	0.2920	0.8442
250	100	0.9338	0.9309	0.3190	0.8412
250	250	0.9447	0.9423	0.2810	0.8453
250	500	0.9563	0.9505	0.2490	0.8476
500	500	0.9575	0.9544	0.2349	0.8489
500	1000	0.9709	0.9614	0.2016	0.8500
1000	500	0.9594	0.9594	0.2170	0.8505
1000	1000	0.9728	0.9668	0.1817	0.8516

Capítulo 6

Conclusiones y Trabajo futuro

En la aplicación de modelos para sistemas de recomendaciones es posible realizar predicciones considerando las opiniones de otros usuarios; aspecto fundamental para lograr establecer patrones de comportamiento de estos.

La implementación de sistemas de recomendaciones permite establecer filtros de búsqueda y con esto, no colapsar a los usuarios con una sobrecarga de información. La ventaja que proporcionan los sistemas de recomendaciones es que son métodos que permiten automatizar las opiniones que pudiese dar un amigo, familiar o conocido respecto de recomendar algún producto o servicio.

En este trabajo se encontró la utilidad de operar en R, el conjunto de datos del *Netflix Prize* en el cluster del IDEUV, mediante la carga gradual de estos. Así mismo RHadoop como herramienta que permite efectuar programación en paralelo, es aplicado en un **Modelo Ingenuo** a modo de ejemplificar su uso, pero en este estudio no fue posible aplicarlo en el **Modelo Propuesto**, dado que los datos del conjunto de entrenamiento del *Netflix Prize* no pudieron almacenarse en HDFS, dadas las dificultades para realizar procesamiento distribuido con algoritmo.

Adicionalmente, se comprueba la mejora en la aplicación del algoritmo propuesto, con respecto al algoritmo de Cinematch, motor de recomendación de *Netflix*.

Queda abierta la posibilidad de implementar algoritmos en sistemas de recomendaciones utilizando otras técnicas, explorando la posibilidad de superar los métodos aplicados en este análisis.

Referencias

Albert A.& Rizzo M. (2012). *R by Example*, New York, USA: Springer.

Anderson, M. (2016). *How To Install R on Ubuntu 16.04*. Recuperado el 1 de Abril de 2017 de https://www.digitalocean.com/community/tutorials /how-to-install-r-on-ubuntu-16-04-2

Anderson, M. (2016). *How to Install Hadoop in Stand Alone Mode on Ubuntu 16.04*. Recuperado el 1 de Abril de 2017 de https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-16-04

Badrinarayanan, S. (2014). *Hadoop Fully Distributed Mode Cluster*. Recuperado de http://hadoopfullydistributedmode.blogspot.cl/

Barranco, R.(2012). *¿Qué es Big Data?*. Recuperado el 30 de Julio de 2017 de https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

Calca, J., Lorenzi, D. & Lish, N. (2009). *Hadoop- Netflix Prize*. Recuperado el 14 de agosto de 2017 de http://hackedexistence.com/project-netflix.html

Cancela, J.(2017). *Gradiente descendente estocástico*. Recuperado el 6 de diciembre de 2017 de https://www.javiercancela.com/2017/04/09/python-machine-learning-iv/

Chapman, C. & McDonnell, F. E. (2015) *R for Marketing Research and*

Analytics. Cham, Suiza: Springer International Publishing

Chiu, D. (2015). *Machine Learning with R Cookbook*, Birmingham, Uk: Packt Publishing.

Davidsson, C.(2010). *Mobile Application Recommender System* (Master of Science Programme in Information Technology Engineering). Uppsala University, Sweden.

Dean J. & Ghemawat S. (2004). *MapReduce: Simplified Data Processing on Large Clusters*. Vol.1. Recuperado el 15 de agosto, 2017 de https://static.googleusercontent.com/media/research.google.com/es//archive/mapreduce-osdi04.pdf.

Forte, R.M. (2015).*Mastering Predictive Analytics with R.*Birmingham, UK: Packt Publishing Ltd

Gadepally, V., Hancock, B., Greenfield, K., Campbell, J., Campbell, W., Reuther, A. (2016). Recommender Systems for the Department of Defense and Intelligence Community.*Lincoln Laboratory Journal*, 22(1). 74-89.

Gorakala S. & Usuelli M. (2015). *Building Recommendation System with R.* Birmingham, Uk: Packt Publishing.

Hadoop Installation. (2014). Recuperado el 15 de mayo, 2017 de http://doctuts.readthedocs.io/en/latest/hadoop.html

Hadoop Team. (2015). *Hadoop Modes*. Recuperado el 15 de mayo, 2017 de http://hadoopinrealworld.com/hadoop-modes/

Hernández, D. A. & Hernández, Y. A. (2015). Acerca de la aplicación de MapReduce + Hadoop en el tratamiento de Big Data. *Revista Cubana de Ciencias Informáticas*. Recuperado el 15 de mayo, 2017 de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992015000300004

Hong, K. (2016). *HADOOP 2.6 INSTALLING ON UBUNTU 14.04* . Recuperado el 11 de Febrero, 2017 de http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_cluster.php

Index of /7/items/nf_prize_dataset.tar/ .(2016).[Archivo de datos]. Recuperado el 1 de Febrero, 2017 de https://archive.org/download/nf_prize_dataset.tar

Irizarry, R. & Hicks, S. (2016). Netflix Prize Challenge. Recuperado 19 de Febrero, 2017 de http://datasciencelabs.github.io/pages/html/hw6-solution.html

Jannach ,D., Zanker, M., Felferning, A.& Friedrich, G. (2010). *Recommender System an introduction*. New York. USA: Cambridge University Press.

Koren, Y. (2009). *The Belkor Solution to the Netflix Prize*. Recuperado 19 de Junio, 2017 de https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf.

Kumar, R. (2015). *How to Setup Hadoop 2.6.5*. Recuperado 4 de Febrero, 2017 de https://tecadmin.net/setup-hadoop-2-4-single-node-cluster-on-linux/

Kumar, R. (2015). *How to Install Oracle JAVA 8*. Recuperado 19 de Mayo, 2017 de https://tecadmin.net/install-oracle-java-8-ubuntu-via-ppa/

López, A., D. (2012). *Cómo saber cuántos procesadores y núcleos tiene una máquina Linux*. Recuperado 19 de Julio, 2017 de http://www.daniloaz.com/es/como-saber-cuantos-procesadores-y-nucleos-tiene-una-maquina-linux/

Manual of the American Psychological Association (6a.ed.). (2010). Washington, USA: American Psychological Association.

Melville, P. & Sindhwani, V. (2010). *Recommender Systems.* Recuperado de http://vikas.sindhwani.org/recommender.pdf

Mishra, P. (2014). *What is big data and Hadoop?.* [Ilustración] Recuperado 19 de Febrero, 2017 de https://www.quora.com/What-is-big-data-and-Hadoop

Mayer-Schönberger, V.& Cukier, K.(2013). *Big data*, Rio de Janeiro, Brasil: Elsevier Editora Ltda.

Netflix, Inc. (2017). [online].
Disponible en https://help.netflix.com/es/node/14164

Noll, M. (2012). *Running Hadoop on Ubuntu Linux.* Recuperado 1 de Abril, 2017 de
http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

Pradel, B., Sean, S., Delporte, J., Guérif, S., Rouveirol, C., Usunier, N.,..., Dufau-Joel, F.(2011). *A Case Study in a Recommender System Based on Purchase Data.* California, USA.

Prajapati, V. (2013). *Big Data Analytics with R and Hadoop*, Birmingham, UK: Packt Publishing.

Pita, P., L. (2016). *Netflix, Data Science sin fronteras.* Recuperado 9 de Marzo, 2017 de https://antoniopitablog.wordpress.com/2016/03/13/netflix-data-science-sin-fronteras/

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ricci, F.,Rokach, L., Shapira, B. & Kantor, P.(2011). *Recommender System Handbook.* New York, USA: Springer.

Rodríguez,F. (2014). *Herramientas para Big Data: Entorno Hadoop* (Trabajo fin de grado), Escuela técnica superior de ingeniería de telecomunicación, Murcia, España.

Ruotsalo, T. (2010). *Methods and Applications for Ontology-Based Recommender System.* (Memoria Doctorado, School of Science and Technology, Aalto University, Helsinki, Finlandia.). Recuperado 22 de Mayo, 2017 de http://lib.tkk.fi/Diss/2010/isbn9789526031514/isbn9789526031514.pdf

Seber, G. & Lee A. (2003). *Linear Regression Analysis.* New Yersey, USA: Wiley Interscience.

Smola, A. (2015). *8 Recommender Systems - Machine Learning Class 10-701* [Archivo de vídeo]. Recuperado 15 de Abril, 2017 de https://www.youtube.com/watch?v=gCaOa3W9kM0

Sutheebanjard,P. (2015). *Install a single-node Hadoop on Ubuntu.* Recuperado 28 de Mayo, 2017de https://blog.phaisarn.com/node/1237

The Netflix Prize Rules (1997-2009). Recuperado 1 de Febrero, 2017 de http://www.netflixprize.com/assets/rules.pdf.

Torres J. (2013). *Servicios y Demonios en Linux.* Recuperado 11 de Agosto, 2017 de https://medium.com/jmtorres/servicios-y-demonios-en-linux-a424366336ac

Vadkar, S.,Siddalingaiah & Venner, J. (2014). *Pro Apache Hadoop*, Estados Unidos: Apress.

Viljanac, V. (2014). *Recommender System for Movile Applications.* (Master Thesis, Univerisity of Zagreb, Croacia). Recuperado de https://www.fer.unizg.hr/_download/repository/Master_Thesis_-_Vanda_Viljanac.pdf

Vinodhini, S., Rajalakshmi, V. & Govindarajalu, B. (Abril, 2014). Building

Personalised Recommendation System With Big Data and Hadoop Mapreduce, 3(4) 2310-2316.

Zhang, D. (2016). *R for Programmers*. New York, USA: CRC Press