Big Data: Preprocesamiento y calidad de datos

Salvador García, Sergio Ramírez-Gallego, Julián Luengo, Francisco Herrera
Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada (España)

<{salvagl, sramirez, julianlm, herrera}@decsal.ugr.es>

Resumen: *En los últimos años, el crecimiento masivo en la escala de los datos está siendo un factor clave en el actual escenario de procesamiento de datos. La eficacia de los algoritmos de extracción de conocimiento depende en gran medida de la calidad de los datos, la cual puede ser garantizada por los algoritmos de preprocesamiento. Sin embargo, en esta era de* Big Data*, los algoritmos de preprocesamiento tienen dificultades para trabajar con tal cantidad de datos, siendo necesario nuevos modelos que mejoren su capacidad de escalado. El objetivo de este trabajo es presentar la importancia del preprocesamiento de datos en* Big Data*, así como, estudiar las herramientas y técnicas de análisis de datos que dan soporte a la tarea del preprocesamiento de datos masivos.*

Palabras clave: Big Data*, calidad de datos, datos imperfectos, datos masivos, discretización, minería de datos, preprocesamiento de datos, selección de atributos, selección de instancias, transformación de datos.*

1. Introducción

Inmensas cantidades de información nos rodean en la actualidad. Tecnologías como Internet generan datos a un ritmo exponencial gracias al abaratamiento y gran desarrollo del almacenamiento y los recursos de red. El volumen actual de datos ha superado las capacidades de procesamiento de los sistemas clásicos de minería de datos [1]. Hemos entrado en la era del *Big Data* o datos masivos [2], que es definida con la presencia de gran volumen, velocidad y variedad en los datos, tres características que fueron introducidas por D. Laney en el año 2001 [3], con el requerimiento de nuevos sistemas de procesamiento de alto rendimiento, nuevos algoritmos escalables, etc.

Otros dos aspectos importantes que caracterizan los datos masivos son la veracidad de los datos y el valor intrínseco del conocimiento extraído. La **figura 1** muestra estas cinco características.

La calidad del conocimiento extraído depende en gran medida de la calidad de los datos. Desgraciadamente, estos datos se ven afectados por factores negativos como: ruido, valores perdidos, inconsistencias, datos superfluos y/o un tamaño demasiado grande en cualquier dimensión (número de atributos e instancias). Está demostrado que una baja calidad de los datos conduce en la mayoría de los casos a una baja calidad del conocimiento extraído.

Recientemente, ha emergido el término "*Smart Data*" que gira alrededor de dos importantes características, la veracidad y el valor de los datos, y cuyo objetivo es filtrar el ruido y mantener los datos valiosos, que pueden ser utilizados para la toma de decisiones inteligentes.

Tres características son asociadas a este nuevo paradigma de datos: exactos, procesables y ágiles (*accurate*, *actionable* y *agile*, en inglés). Una descripción breve de estos términos nos conduce a tres aspectos esenciales en el uso de los datos: a) los datos deben ser lo que se dice, es importante la calidad de datos; b) los datos deben ser escalables para su procesamiento; c) los datos deben estar disponibles y preparados para adaptarse al entorno cambiante de los negocios.

Autores

Salvador García es profesor titular en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada. Ha publicado más de 45 trabajos en revistas internacionales en diferentes áreas de la minería de datos. Es co-editor de la revista "*Progress in Artificial Intelligence*" (Springer). Es coautor del libro "*Data Preprocessing in Data Mining*" (Springer, 2015). Aparece en la lista de *Highly Cited Researchers* de Thomson Reuters de los años 2014 y 2015 <www.highlycited.com>. Sus temas de intereses incluyen: *Big Data*, minería de datos, preprocesamiento de datos, aprendizaje semi-supervisado, inferencia estadística, y aprendizaje evolutivo y biometría, entre otros temas.

Sergio Ramírez-Gallego es estudiante de doctorado en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada. Entre sus temas de interés destacan: *Big Data*, minería de datos y preprocesamiento de datos.

Julián Luengo es profesor contratado en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada. Es coautor del libro "*Data Preprocessing in Data Mining*" (Springer, 2015). Ha recibido, entre otros, sendos premios a los trabajos de investigación de excelencia de la Universidad de Granada en los años 2013 y 2014. Sus temas de intereses incluyen: *Big Data*, minería de datos, datos imperfectos y preprocesamiento de datos.

Francisco Herrera es catedrático en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada. Ha dirigido 38 tesis doctorales y ha publicado más de 300 trabajos en revistas científicas. Es coautor de los libros: "*Genetic Fuzzy Systems*" (World Scientific, 2001), "*Inteligencia Artificial, Inteligencia Computacional y Big Data*". (Servicio. Pub. Univ. Jaen, 2014). "*Data Preprocessing in Data Mining*" (Springer, 2015), "*The 2-tuple Linguistic Model. Computing with Words in Decision Making*" (Springer, 2015), "*Multilabel Classification. Problem analysis, metrics and techniques*" (Springer, 2016) y "*Multiple Instance Learning: Foundations and Algorithms*" (Springer, 2016). Es editor jefe de las revistas "*Information Fusion*" (Elsevier) y "*Progress in Artificial Intelligence*", y participa en el comité editorial de una docena de revistas internacionales. Ha recibido, entre otros, el Premio Nacional de Informática ARITMEL 2010 de la Sociedad Científica Informática de España; el 2010 *International Award: International Cajastur "Mamdani Prize" for Soft Computing* (4ª edición, 2010, otorgado por el *European Center of Soft Computing*), *IEEE Transactions on Fuzzy System Outstanding 2008 and 2012 Paper Award* (*bestowed in* 2011 *and* 2015); 2011 *Lotfi A. Zadeh Prize Best Paper Award* 2009-10 (*International Fuzzy Systems Association*); Reconocimiento AEPIA 2013; Galardón "Natural de Jaén" 2014 de la Universidad de de Jaén; XV Premio Andalucía de Investigación "Maimónides" 2014 para el área de ciencias experimentales. Aparece en la lista de *Highly Cited Researchers* de Thomson Reuters de los años 2014 y 2015 <www.highlycited.com>. Sus temas de intereses incluyen: *Big Data*, preprocesamiento de datos, ciencia de datos, minería de datos, fusión de información y toma de decisiones, e inteligencia computacional (sistemas difusos y algoritmos evolutivos), entre otros temas.

> “Recientemente, ha emergido el término ‘*Smart Data*’ que gira alrededor de dos importantes características, la veracidad y el valor de los datos”

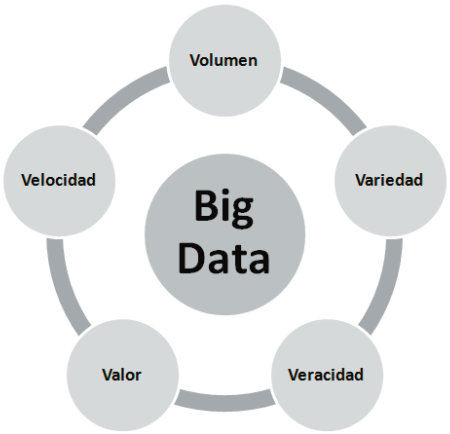

Figura 1. Características que definen los datos masivos (*Big Data*).

El preprocesamiento de datos [4] es una etapa fundamental en el proceso de extracción de conocimiento, cuyo objetivo principal es obtener un conjunto de datos final que sea de calidad y útil para la fase de extracción de conocimiento.

El preprocesamiento de datos se vislumbra como una herramienta muy importante en el paso de *Big Data* a *Smart Data*, esencial para convertir los datos almacenados (material en bruto) en datos de calidad (valga el símil del paso de un diamante en bruto sin pulir y sin tallar a la piedra preciosa tras su procesado).

Para la mayoría de problemas actuales con datos masivos es necesario el uso de una solución distribuida escalable porque las soluciones secuenciales no son capaces de abordar tales magnitudes. Varias plataformas para el procesamiento a gran escala (como Spark o Hadoop) han intentado afrontar la problemática del *Big Data* en los últimos años [5]. Estas plataformas requieren algoritmos escalables que den soporte a las tareas más relevantes de la analítica de datos masivos.

Los algoritmos de preprocesamiento también están afectados por el problema de la escalabilidad, por lo tanto deben ser rediseñados para su uso con tecnologías *Big Data* si queremos preprocesar conjuntos de datos masivos en los diferentes escenarios de aplicación, aprendizaje supervisado y no supervisado, procesamiento en tiempo real (flujo masivo de datos), etc.

Este artículo tiene como objetivo introducir el preprocesamiento de datos para *Big Data*, así como enumerar y describir las tecnologías y herramientas de analítica de datos y las técnicas existentes para el preprocesamiento de datos para *Big Data*.

Para ello, el trabajo se organiza como sigue. En la **sección 2** describimos en qué consiste el preprocesamiento de datos y las dos áreas en las que podemos clasificar las técnicas de preprocesamiento. En la **sección 3** introducimos las tecnologías y herramientas para el procesamiento de datos masivos (*Big Data*). En la **sección 4** estudiamos las propuestas escalables para el procesamiento de datos masivos y presentamos un caso de uso para selección de atributos. Finalmente, en la **sección 5** presentamos unas breves conclusiones y discutimos sobre los retos futuros a los que se enfrenta el preprocesamiento en *Big Data*.

2. Preprocesamiento de datos

El preprocesamiento de datos es una etapa esencial del proceso de descubrimiento de información o KDD (*Knowledge Discovery in Databases*, en inglés) [6][7]. Esta etapa se encarga de la limpieza de datos, su integración, transformación y reducción para la siguiente fase de minería de datos [4].

La **figura 2** muestra las distintas etapas del proceso del KDD.

Debido a que normalmente el uso de datos de baja calidad implica un proceso de minería de datos con pobres resultados, se hace necesaria la aplicación de técnicas de preprocesamiento.

Después de la aplicación de la fase de preprocesamiento, el conjunto resultante puede ser visto como una fuente consistente y adecuada de datos de calidad para la aplicación de algoritmos de minería de datos. El preprocesamiento incluye un rango amplio de técnicas que podemos agrupar en dos áreas: preparación de datos y reducción de datos.

La preparación de datos está formada por una serie de técnicas que tienen el objetivo de inicializar correctamente los datos que servirán de entrada para los algoritmos de minería de datos. Este tipo de técnicas pueden clasificarse como de uso obligado, ya que sin ellas los algoritmos de extracción de conocimiento no podrían ejecutarse u ofrecerían resultados erróneos. En esta área se incluye la transformación de datos y normalización, integración, limpieza de ruido e imputación de valores perdidos (ver **figura 3**).

Las técnicas de reducción de datos se orientan a obtener una representación reducida de los datos originales, manteniendo en la mayor medida posible la integridad y la información existente en los datos.

Es por esta razón que la aplicación de técnicas de reducción no se considera estrictamente obligatoria. Sin embargo, cuando el tiempo de ejecución de un algoritmo o el tamaño de los datos son prohibitivos para los algoritmos de extracción, estas técnicas deben ser aplicadas para obtener conjuntos de datos más pequeños y de calidad.

En esta área las técnicas de reducción más relevantes son: la selección de atributos

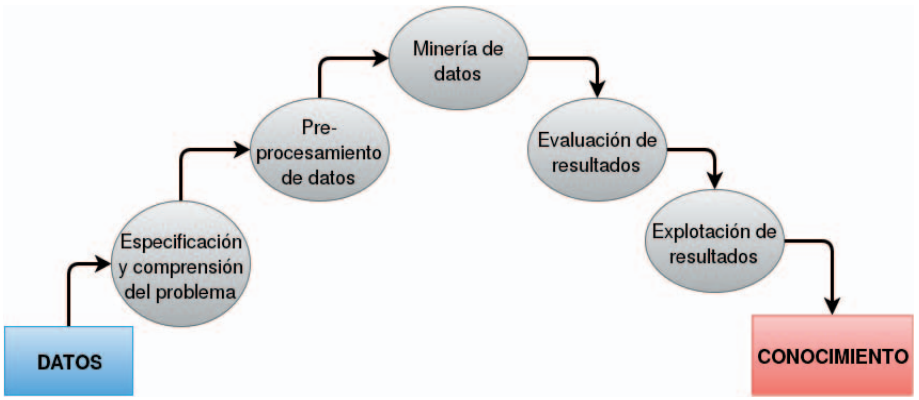

Figura 2. Esquema general del proceso de descubrimiento de información en bases de datos.

> “La preparación de datos está formada por una serie de técnicas que tienen el objetivo de inicializar correctamente los datos que servirán de entrada para los algoritmos de minería de datos”

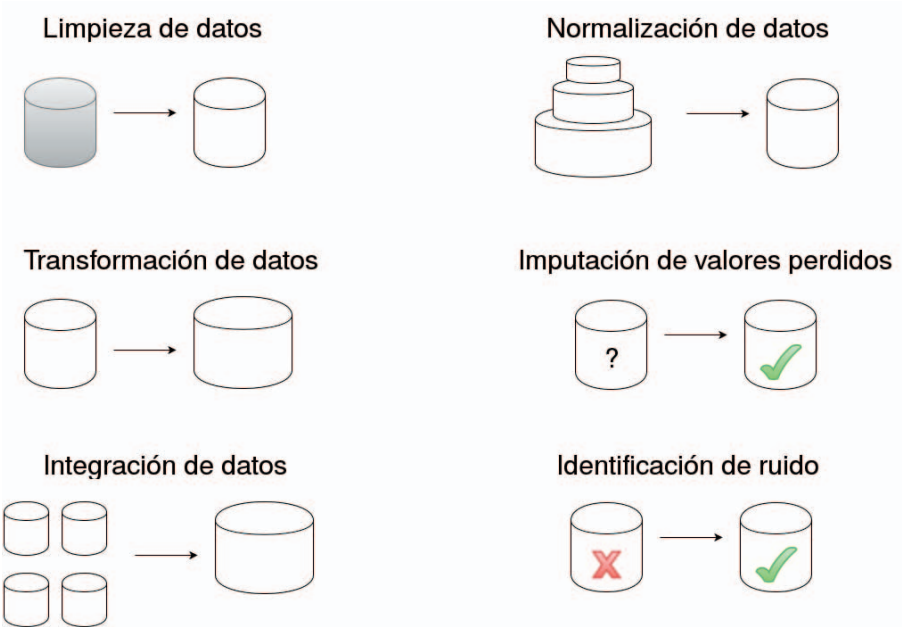

Figura 3. Familias de técnicas en preparación de datos.

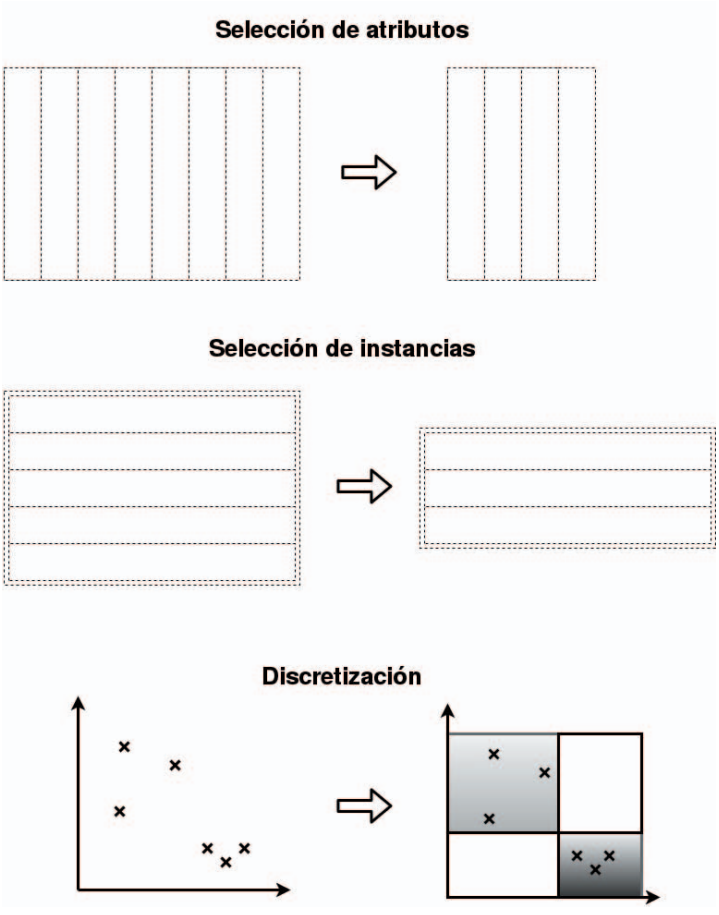

Figura 4. Familias de técnicas en reducción de datos.

(*Feature Selection* (FS) en inglés), la selección de instancias (*Instance Selection* (IS) en inglés) o la discretización (ver **figura 4**).

3. *Big Data*

El ritmo actual de generación de datos está sobrepasando las capacidades de procesamiento de los sistemas actuales en compañías y organismos públicos. Las redes sociales, el Internet de las Cosas y la industria 4.0 son algunos de los nuevos escenarios con presencia de datos masivos.

La necesidad de procesar y extraer conocimiento valioso de tal inmensidad de datos se ha convertido en un desafío considerable para científicos de datos y expertos en la materia. El valor del conocimiento extraído es uno de los aspectos esenciales de *Big Data*, como ya hemos comentado en la introducción.

A continuación describimos brevemente las tecnologías y herramientas para la analítica de datos masivos.

3.1. Tecnologías para *Big Data*

Las tecnologías y algoritmos sofisticados y novedosos son necesarios para procesar eficientemente lo que se conoce como *Big Data*. Estos nuevos esquemas de procesamiento han de ser diseñados para procesar conjuntos de datos grandes, datos masivos, dentro de tiempo de cómputo razonable y en un rango de precisión adecuado.

Desde el punto de vista del aprendizaje automático, esta problemática ha causado que muchos algoritmos estándar se conviertan en obsoletos en el paradigma *Big Data*. Como resultado surge la necesidad de diseñar nuevos métodos escalables capaces de manejar grandes volúmenes de datos, manteniendo a su vez su comportamiento en términos de efectividad.

Google diseñó MapReduce en 2003 [8] que es considerada como la plataforma pionera para el procesamiento de datos masivos, así como un paradigma para el procesamiento de datos mediante el particionamiento de ficheros de datos. MapReduce es capaz de procesar grandes conjuntos de datos, a la vez que proporciona al usuario un manejo fácil y transparente de los recursos del *clúster* subyacente.

> “ Spark se ha convertido en una de las herramientas más potentes y populares en el ecosistema del *Big Data*, habiendo demostrado ser más eficiente que Hadoop en muchos casos de uso ”

En el paradigma MapReduce, existen dos fases: Map y Reduce. En la fase Map, el sistema procesa parejas clave-valor, leídas directamente del sistema de ficheros distribuido, y transforma estos pares en otros intermedios usando una función definida por el usuario. Cada nodo se encarga de leer y transformar los pares de una o más particiones. En la fase Reduce, los pares con claves coincidentes son enviadas al mismo nodo y finalmente fusionados usando otra función definida por el usuario.

La **figura 5** muestra un esquema general del proceso completo MapReduce.

La función Map tiene como entrada una serie de pares <clave, valor> y produce una lista de pares intermedios como salida. La función Map, que internamente procesa los datos en cada proceso, es definida por el usuario siguiendo el esquema clave-valor. El esquema general para dicha función es el siguiente:

Map(<clave1, valor1>) -> lista(<clave2, valor2>)

En la segunda fase, el nodo maestro agrupa pares por clave y distribuye los resultados combinados a los procesos Reduce en cada nodo. La función de reducción es aplicada a a la lista de valores asociada a cada clave y genera un valor de salida. Dicho proceso es esquematizado a continuación:

Reduce(<clave2,lista(valor2)>)→<clave3,valor3>

A pesar de su popularidad, MapReduce y su versión de código abierto Hadoop [9] han mostrado limitaciones en ciertos escenarios, especialmente en aquellos que implican la reutilización de datos como los procesos iterativos, el procesamiento de grafos, etc. [10].

Apache Spark [11] nace como una alternativa que intenta dar solución a las limitaciones de MapReduce/Hadoop. Spark se ha convertido en una de las herramientas más potentes y populares en el ecosistema del *Big Data*, habiendo demostrado ser más eficiente que Hadoop en muchos casos de uso (entre 10 y 100 veces trabajando en memoria). Gracias a sus operaciones de uso intensivo de memoria, esta plataforma es capaz de cargar datos en memoria y consultarlos rápidamente.

Por otra parte, Spark es perfecto para procesos iterativos donde un mismo dato es reutilizado varias veces para el procesamiento de algoritmos sobre grafos, etc.

Merece destacar otra plataforma emergente, como es Apache Flink [12]. Esta plataforma intenta llenar el hueco entre el procesamiento en tiempo real y el secuencial dejado por Spark. Flink es una plataforma distribuida para flujos de datos que también puede trabajar con datos secuenciales, mientras que Spark emula el procesamiento en tiempo real usando mini-lotes de datos. Flink muestra un buen rendimiento en el procesamiento de datos en sistemas de baja latencia.

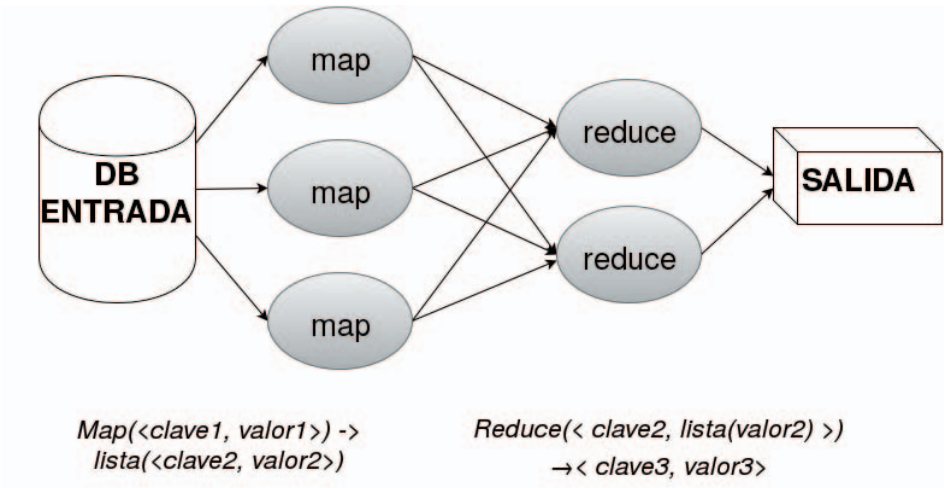

Figura 5. Esquema general del proceso MapReduce.

3.2. Herramientas para la analítica de datos masivos

En los últimos años, ha surgido un gran abanico de herramientas de analítica de datos escalables asociadas a las plataformas anteriores, con el objetivo de dar soporte al proceso de análisis de datos. A continuación, describimos brevemente algunas:

- Mahout [13]: Esta biblioteca ofrece implementaciones basadas en Hadoop MapReduce para varias tareas de analítica de datos como el agrupamiento, la clasificación o el filtrado colaborativo. La versión actual denominada Mahout Samsara (0.12.2) está evolucionando para ser compatible sobre otras plataformas del ecosistema de Hadoop, como Spark o Flink.
- MLlib [14]: Nacida junto al proyecto de Spark; es una biblioteca de aprendizaje automático que contiene varias utilidades estadísticas y algoritmos de aprendizaje. Esta biblioteca contiene algoritmos que dan soporte a tareas del proceso de extracción del conocimiento como clasificación, optimización, regresión, agrupamiento, y preprocesamiento.
- FlinkML [15]: Es la biblioteca nativa para análisis distribuido de datos de Flink. FlinkML incluye algoritmos escalables para tareas como la clasificación, el agrupamiento, el preprocesamiento de datos y la recomendación. Aunque está lejos de ofrecer la variedad de otras bibliotecas como MLlib, ofrece algunas técnicas que se proponen mejorar a las de MLlib, como por ejemplo la implementación del algoritmo de *Support Vector Machines* (SVM), que está basada en un algoritmo de comunicación extremadamente eficiente.
- H2O [16]: Es una plataforma de código abierto para análisis *Big Data*. H2O destaca por su aproximación al *deep learning*, y por sus implementaciones iterativas. Estas últimas permiten que el usuario decida si obtener la solución más óptima o interrumpir la computación y obtener una solución aproximada. H2O puede ser ejecutada en sistemas tradicionales (Windows, Linux, etc.), así como en plataformas *Big Data* (como Spark).

4. Preprocesamiento para *Big Data*

En esta sección presentamos las diferentes soluciones que existen actualmente para preprocesar conjuntos de datos masivos. En

> “ MLlib destaca como la herramienta que ofrece una mayor variedad de métodos de preprocesamiento ”

la **sección 4.1** describimos las propuestas procedentes de herramientas para analítica de datos, así como las presentadas en la literatura. En la **sección 4.2** mostramos un caso de uso dónde se aplica un algoritmo de selección de atributos escalable sobre un problema de alta dimensionalidad.

4.1. Algoritmos de preprocesamiento de datos masivos

En esta subsección mostramos brevemente los algoritmos de preprocesamiento disponibles en las herramientas de analítica de datos previamente descritas, así como las propuestas que encontramos en la literatura especializada.

MLlib destaca como la herramienta que ofrece una mayor variedad de métodos de preprocesamiento. Esta biblioteca de software ofrece algoritmos para diversas tareas:

- Discretización y normalización: Transforma atributos continuos usando intervalos discretos, mientras que la normalización realiza un ajuste en la distribución. Algoritmos: binarizador, discretizador manual, normalizadores basados en min-max o media-varianza, etc.
- Extracción de atributos: Combina el conjunto original de atributos para obtener un nuevo conjunto de atributos menos redundante, usando proyecciones, por ejemplo. Algoritmos: *Principal Component Analysis* (PCA) o *Single Value Decomposition* (SVD).
- Selección de atributos: Selecciona subconjuntos de atributos, minimizando la pérdida de información. Algoritmos: Chi-cuadrado, RFormula, etc.
- Conversores para atributos: Transforman atributos de un tipo a otro usando técnicas de indexación o codificación. Algoritmos: *OneHotEncoder*.
- Técnicas para el preprocesamiento de texto: tienen como objetivo estructurar la entrada de texto, produciendo patrones de información estructurados. Algoritmos: *Term Frequency-Inverse Document Frequency* (TF-IDF), *n*-gram, etc.) [17].

En la biblioteca FlinkML encontramos tres métodos de preprocesamiento actualmente (versión 1.1 de Flink): Un algoritmo que transforma un conjunto de atributos a un espacio polinomial, y dos algoritmos para normalización (uno basado en min-max y otro en media-varianza). Sin embargo, los creadores de FlinkML planean desarrollar varios algoritmos de reducción de dimensionalidad, extracción de atributos en datos textuales y *hashing* de atributos.

H2O ofrece algunos algoritmos de extracción de atributos como PCA, así como algunos métodos para normalización y anonimización; mientras que Mahout actualmente sólo ofrece algoritmos para reducción de dimensionalidad como: SVD, QR *Decomposition* [18] o versiones estocásticas de SVD y PCA.

En cuanto a la literatura especializada, en la **tabla 1** se enumeran y referencian los estudios y algoritmos propuestos. Se clasifican acorde a la familia de técnicas de preprocesamiento (selección de atributos -FS-, tratamiento de datos incompletos y desbalanceados, discretización, o selección de instancias -IS), el número de atributos, número de instancias y tamaño máximo de datos (en GB) que maneja cada algoritmo en sus experimentos, y la plataforma sobre la que trabajan. El tamaño se calcula multiplicando el número de atributos por el número de instancias, 8 bytes por dato.

Como se puede observar en la **tabla 1**, la mayoría de los métodos están implementados bajo el paradigma Hadoop MapReduce (Hadoop MR), un paradigma que presenta limitaciones para su uso eficiente en procesos

MÉTODO (VER “REFERENCIAS”)	CATEGORÍA	# ATRIBUTOS	# INSTANCIAS	TAMAÑO (GB)	PLATAFORMA
[19]	FS	630	65.003.913	305,12	Hadoop MR
[20]	FS	630	65.003.913	305,12	Apache Spark
[21]	FS	1.156	5.670.000	48,84	MPI
[22]	FS	29.890.095	19.264.097	4,16	C++/ MatLab
[23]	FS	100.000	10.000.000	1,49	MR
[24]	FS	100	1.600.000	1,19	Apache Spark
[25]	FS	54.675	2.096	0,85	Hadoop MR
[26]	FS	54	581.012	0,23	Hadoop MR
[27]	FS	20	1.000.000	0,15	MR
[28]	FS	-	-	0,10	Hadoop MR
[29]	FS	256	38.232	0,07	Hadoop MR
[30]	FS	52	5.253	0,00	Hadoop MR
[31]	FS	-	-	0,00	Hadoop MR
[32]	FS	-	-	0,00	Hadoop MR
[33]	FS	-	-	0,00	Hadoop MR
[34]	Desbalanceados	630	32.000.000	150,20	Hadoop MR
[35]	Desbalanceados	41	4.856.151	1,48	Hadoop MR
[36]	Desbalanceados	6	29.887.416	1,34	Hadoop MR
[37]	Desbalanceados	14	1.432.941	0,15	Hadoop MR
[38]	Desbalanceados	18	492.796	0,07	Hadoop MR
[39]	Desbalanceados	36	95.048	0,03	Hadoop MR
[40]	Incompletos	625	4.096.000	19,07	MR (Twister)
[41]	Incompletos	481	191.779	0,69	Hadoop MR
[42]	Discretización	630	65.003.913	305,12	Apache Spark
[43]	Discretización	-	-	4,00	Hadoop MR
[44]	IS	41	4.856.151	1,48	Hadoop MR

Tabla 1. Listado de métodos para preprocesamiento en *Big Data*[1].

> “Una contribución reseñable para el preprocesamiento de datos masivos, y en particular, para selección de atributos, es el algoritmo fast-mRMR”

iterativos. También se puede apreciar cómo la mayoría de propuestas manejan una cantidad de datos menor a un GB, lo cual crea dudas sobre si determinadas propuestas son capaces de escalar bien cuando tengan que enfrentarse a conjuntos de datos masivos.

Es necesario, por tanto, un mayor esfuerzo en el desarrollo de algoritmos de preprocesamiento en nuevas plataformas naturalmente iterativas, como Spark o Flink, dado el bajo número de propuestas existentes si las comparamos con los algoritmos de preprocesamiento para minería de datos [4].

4.2. Caso de uso: Selección de atributos escalable, algoritmo Fast-mRMR

Una contribución reseñable para el preprocesamiento de datos masivos, y en particular, para selección de atributos, es el algoritmo fast-mRMR [42].

Esta propuesta incluye varias optimizaciones a la eficiencia del algoritmo original mRMR, uno de los más populares en su ámbito. Gracias a estas optimizaciones, fast-mRMR es capaz de trabajar con conjuntos de datos masivos en ambas dimensiones: nº de atributos y nº de instancias. Entre las optimizaciones más relevantes introducidas por fast-mRMR están:

- **Redundancia acumulada**: fast-mRMR desarrolla una aproximación voraz al problema de cálculo de importancia de atributos. El algoritmo almacena la redundancia calculada en cada iteración para evitar cálculos innecesarios. De esta manera, en el algoritmo sólo la información mutua entre los atributos no-seleccionados es calculada en cada iteración.
- **Re-utilización de cálculos previos**: fast-mRMR almacena para su re-utilización algunos datos importantes cuando son calculados por primera vez, como, por ejemplo, la relevancia inicial o algunas probabilidades con respecto a la clase.
- **Procesamiento por columnas**: en fast-mRMR los datos iniciales (tabulados normalmente formando una lista de filas) son transformados a una lista de columnas de manera que los cálculos entre atributos se tornan más sencillos de realizar. Para ello, se realiza una transformación inicial sobre los datos originales y se almacenan los datos en este nuevo formato para las posteriores iteraciones.

Para mostrar la eficiencia de fast-mRMR, se muestra un experimento con un *clúster* de 432 *cores* (24 cores/nodo) y 864 GB de RAM (48 GB/nodo), donde se seleccionaban los 100 atributos más relevantes para cada conjunto de datos. Cada atributo seleccionado implica una iteración del algoritmo.

La **tabla 2** muestra los resultados de tiempo de procesamiento (en segundos) tras aplicar fast-mRMR sobre tres conjuntos de datos masivos. Hemos de destacar que fast-mRMR es capaz de procesar estos conjuntos de datos en rangos de tiempo razonables que nunca superan una hora de procesamiento. Especialmente remarcable es el caso de *kddb* con casi 30 millones de atributos a considerar.

La propuesta, además de ser eficiente, ha mostrado ser escalable ante un aumento del número de núcleos de procesamiento. La **figura 6** muestra los resultados de tiempo de procesamiento de fast-mRMR con diferentes configuraciones del *clúster* con respecto al número de núcleos disponibles para el procesamiento. Los resultados nos permiten afirmar que el tiempo de selección usando fast-mRMR desciende siguiendo una tendencia conforme el número de núcleos disponibles aumenta.

5. Conclusiones

En este trabajo se estudia la creciente importancia del preprocesamiento de datos en *Big Data*. Se presenta una revisión de las tecnologías de *Big Data*, herramientas de analítica de datos y técnicas y algoritmos disponibles para el preprocesamiento de datos masivos.

Como ya se ha mencionado, el número de propuestas es muy bajo en comparación con el número de algoritmos en minería de datos. Se tiene el reto de diseñar nuevos algoritmos de preprocesamiento de datos masivos.

Por otra parte, la mayoría de los algoritmos de preprocesamiento para *Big Data* han sido diseñados para abordar el problema de la selección de características. Por tanto, son necesarios nuevos algoritmos que deben centrarse en problemas tales como la selección de instancias o el tratamiento de datos imperfectos (ruido y valores perdidos), entre

CONJUNTO	# ATRIBUTOS	# INSTANCIAS	TIEMPO (S)	DISPERSO
ECBDL14	631	65.003.913	2.420	No
epsilon	2.000	400.000	542	No
kddb	29.890.095	19.264.097	2.789	Sí

Tabla 2. Tiempo de selección de fast-mRMR para varios conjuntos de datos masivos[2].

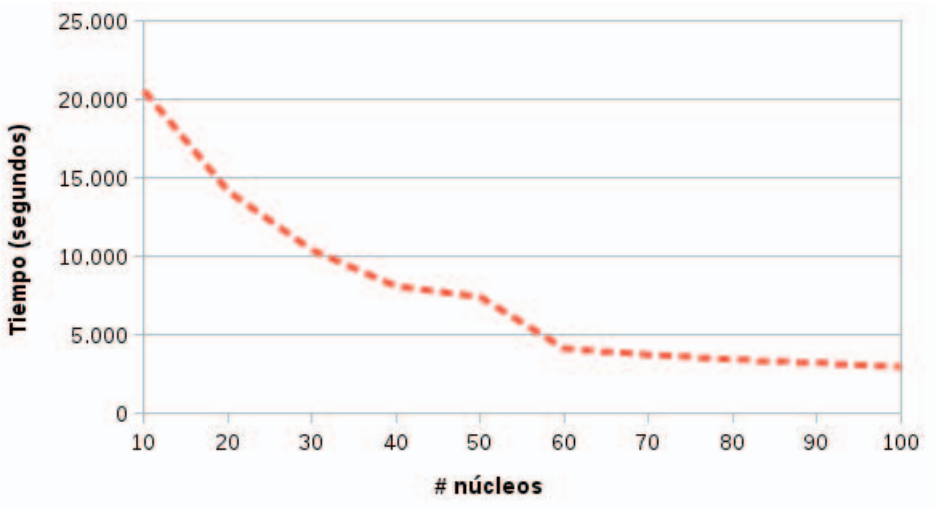

Figura 6. Tiempo de procesamiento en segundos (eje Y) utilizado por fast-mRMR dependiendo del número de núcleos de procesamiento disponibles (eje X).

otros, con altos requerimientos de computación y complejidad de las relaciones entre los datos. Y, por supuesto, prestando atención a la escalabilidad y disponibilidad para las diferentes plataformas distribuidas como Apache Spark o Flink u otras que pueden ir apareciendo en el futuro.

Terminamos destacando la importancia del nuevo paradigma emergente, *Smart Data*, como paso hacia la disponibilidad de datos de calidad. *Big Data* es una gran mina a explotar, datos almacenados que poseen un gran valor en sí mismo pero que necesitan ser tratados, refinados y preprocesados mediante herramientas de analítica de datos.

Smart Data son datos de calidad, prestos para ser utilizados en la extracción de conocimiento y la toma de decisiones inteligente basada en datos. El preprocesamiento de datos es fundamental para convertir los datos almacenados en datos de calidad. ¡Decisiones de calidad deben estar soportadas por datos de calidad!

Agradecimientos

Este trabajo está financiado por el Proyecto de Investigación Nacional TIN2014-57251-P y el Plan Andaluz de Investigación P11-TIC-7765. S. Ramírez-Gallego está financiado por una beca FPU del Ministerio de Educación y Ciencia (FPU13/00047).

Referencias

[1] X. Wu, X Zhu, GQ. Wu, W. Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*. 2014;26(1): pp. 97–107.

[2] V. Mayer-Schnberger. *Big Data: A Revolution that will Transform how We Live, Work and Think*. John Murray Publishers, 2013.

[3] D. Laney. *3D Data Management: Controlling Data Volume, Velocity and Variety*. 2001. <http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>. Último acceso: Julio de 2016.

[4] S. García, J. Luengo, F. Herrera. *Data Preprocessing in Data Mining*. Berlin, Germany: Springer, 2015.

[5] A. Fernández, S. del Río, V. López, A. Bawakid, MJ. del Jesús, JM. Benítez, et al. Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2014, 4(5)*: pp. 380–409.

[6] J. Han, M. Kamber, J. Pei. *Data Mining: Concepts and Techniques*. 3rd ed. Burlington, MA, USA: Morgan Kaufmann Publishers Inc., 2011.

[7] MJ. Zaki, W. Meira. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. New York, NY, USA: Cambridge University Press, 2014.

[8] J. Dean, S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *OSDI 2004*; 2004: pp. 137–150.

[9] T. White. *Hadoop, The Definitive Guide*. O'Reilly Media, Inc., 2012.

[10] J. Lin. MapReduce is Good Enough? If All You Have is a Hammer, Throw Away Everything That's Not a Nail! *Big Data. 2012,1(1)*: pp. 28–37.

[11] H. Karau, A. Konwinski, P. Wendell, M. Zaharia. *Learning Spark: Lightning-Fast Big Data Analytics*. O'Reilly Media; 2015.

[12] Flink. *Apache Flink*, 2016. <https://flink.apache.org/>. Último acceso: Julio de 2016.

[13] Mahout. *Apache Mahout*, 2016. <https://mahout.apache.org/>. Último acceso: Julio de 2016.

[14] X. Meng, JK. Bradley, B. Yavuz, ER. Sparks, S. Venkataraman, D. Liu et al. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research, 2016; 17(34)*: pp. 1–7.

[15] FlinkML. *Flink ML – Machine Learning for Flink*, 2016. <https://ci.apache.org/projects/flink/flink-docs-master/apis/batch/libs/ml/index.html>. Último acceso: Julio de 2016.

[16] H2O. *H2O library*. 2016. <http://www.h2o.ai/>. Último acceso: Julio de 2016.

[17] C. Aggarwal, C. Zhai. *Mining Text Data*. Springer Publishing Company, Incorporated, 2012.

[18] J. Stoer, R. Bulirsch. *Introduction to Numerical Analysis*, Springer, 2002.

[19] D. Peralta, S. Río, S. Ramírez, I. Triguero, JM. Benítez, F. Herrera. Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach. *Mathematical Problems in Engineering, 2015*, Article ID 246139.

[20] S. Ramírez-Gallego, S. García, H. Mouriño Talín, D. Martínez-Rego, V. Bolón-Canedo, A. Alonso-Betanzos, J.M. Benítez, F. Herrera. Data discretization: taxonomy and big data challenge. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2016,6(1)*: pp. 5–21.

[21] Z. Zhao, R. Zhang, J. Cox, D. Duling, W. Sarle. Massively parallel feature selection: an approach based on variance preservation. *Machine Learning. 2013, 92(1)*: pp. 195–220.

[22] M. Tan, IW. Tsang, L. Wang. Towards Ultrahigh Dimensional Feature Selection for Big Data. *Journal of Machine Learning Research, 2014,15*: pp. 1371–1429.

[23] S. Singh, J. Kubica, SE. Larsen, D. Sorokina. Parallel Large Scale Feature Selection for Logistic Regression. *SIAM International Conference on Data Mining (SDM), 2009*, pp. 1172–1183.

[24] B. Ordozgoiti, S. Gómez-Canaval, A. Mozo. Massively Parallel Unsupervised Feature Selection on Spark. *New Trends in Databases and Information Systems, vol. 539 of Communications in Computer and Information Science, 2015*, pp. 186–196.

[25] M. Kumar, SK. Rath. Classification of microarray using MapReduce based proximal support vector machine classifier. *Knowledge-Based Systems, 2015, 89*: pp. 584-602.

[26] Z. Sun, Z. Li. Data intensive parallel feature selection method study. *International Joint Conference on Neural Networks (IJCNN), 2014*, pp. 2256–2262.

[27] K. Chen, Wen-qiang Wan, Y. Li. Differentially private feature selection under MapReduce framework. *The Journal of China Universities of Posts and Telecommunications. 2013, 20(5)*: pp. 85-103.

[28] P. Chao, W. Bin, D. Chao. Design and Implementation of Parallel Term Contribution Algorithm Based on Mapreduce Model. *7th Open Cirrus Summit, 2012*. pp. 43–47.

[29] Q. He, X. Cheng, F. Zhuang, Z. Shi. Parallel feature selection using positive approximation based on MapReduce. *11th International Conference on Fuzzy Systems and Knowledge Discovery FSKD*, 2014, pp. 397-402.

[30] S. Tanupabrungsun, T. Achalakul. Feature Reduction for Anomaly Detection in Manufacturing with MapReduce GA/kNN. *19th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 2013*, pp. 639–644.

[31] M. Dalavi, S. Cheke. Hadoop MapReduce implementation of a novel scheme for term weighting in text categorization. *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014*. pp. 994–999.

[32] MJ. Meena, KR. Chandran, A. Karthik, AV. Samuel. An enhanced ACO algorithm to select features for text categorization and its parallelization. *Expert Systems with Applications. 2012, 39(5)*: pp. 5861-5871.

[33] VJ. Hodge, S. O'Keefe, J. Austin. Hadoop neural network for parallel and distributed feature selection. *Neural Networks. 2015*, pendiente de publicar. DOI: <10.1016/j.neunet.2015.08.011>.

[34] I. Triguero, S. del Río, V. López, J. Bacardit, JM. Benítez, F. Herrera. ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems, 2015, 87*: pp. 69–79.

[35] S. del Río, V. López, JM. Benítez, F. Herrera. On the use of MapReduce for imbalanced big data using Random Forest. *Information Sciences. 2014, 285*: pp. 112-137.

[36] D. Galpert, S. Del Río, F. Herrera, E. Ancede-Gallardo, A. Antunes, G. Aguero-Chapin. An Effective Big Data Supervised Imbalanced Classification Approach for Ortholog Detection in Related Yeast Species. *BioMed Research International, 2015*, article 748681.

[37] SH. Park, YG. Ha. Large Imbalance Data Classification Based on MapReduce for Traffic Accident Prediction. *8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014*, pp. 45–49.

[38] J. Wang, P. Zhao, SCH. Hoi, R. Jin. Online Feature Selection and Its Applications. *IEEE Transactions on Knowledge and Data Engineering, 2014, 26(3)*: pp. 698–710.

[39] RC. Bhagat, SS. Patil. Enhanced SMOTE algorithm for classification of imbalanced big-data using Random Forest. *IEEE International Advance Computing Conference (IACC), 2015*, pp. 403–408.

[40] J. Zhang, JS. Wong, Y. Pan, T. Li. A Parallel Matrix-Based Method for Computing Approximations in Incomplete Information Systems. *IEEE Transactions on Knowledge and Data Engineering. 2015, 27(2)*: pp. 326–339.

[41] F. Chen, L. Jiang. A Parallel Algorithm for Datacleansing in Incomplete Information Systems Using MapReduce. *10th International Conference on Computational Intelligence and Security (CIS), 2014*, pp. 273–277.

[42] S. Ramírez-Gallego, I. Lastra, D. Martínez-Rego, V. Bolón-Canedo, JM. Benítez, F. Herrera, A. Alonso-Betanzos. Fast-mRMR: Fast Minimum Redundancy Maximum Relevance Algorithm for High-Dimensional Big Data. *International Journal of Intelligent Systems. 2016*, pendiente de publicar. <DOI: 10.1002/int.21833>.

[43] Y. Zhang, J. Yu, J. Wang. Parallel Implementation of Chi2 Algorithm in MapReduce Framework. *Human Centered Computing - First International Conference, HCC, 2014*, pp. 890–899.

[44] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera. MRPR: A MapReduce solution for prototype reduction in big data classification. *Neurocomputing. 2015, 150, Part A*: pp. 331–345.

Notas

[1] Los métodos han sido agrupados por familia (categoría), y ordenados por el tamaño máximo de datos manejado.

[2] Se indican número de atributos, número de instancias, tiempo de selección y si el conjunto de datos está en formato disperso.