Universidad de las Ciencias Informáticas

Facultad *3*

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Eliminación de itemsets no significativos para el usuario en la etapa de pre procesamiento para la extracción de reglas de asociación.

Autor: Bernardo Corrales Armbruster

Tutor: Msc. Julio Cesar Diaz Vera

La Habana, junio del 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Bernardo Corrales Armbruster	Julio Cesar Diaz Vera
_____	_____
Autor	**Tutor**

DATOS DE CONTACTO

Julio Cesar Diaz Vera, graduado de Ingeniería en Telecomunicaciones en la Universidad Martha Abreu ubicada en Villa Clara, Cuba. Máster en Gestión de Proyectos, grado científico alcanzado en la Universidad de la Ciencias Informáticas, La Habana, Cuba. Posee varias publicaciones nacionales e internacionales referentes al tema abordado en esta investigación, alcanzando una experiencia de más de 10 años.

Correo electrónico: jcdiaz@uci.cu

AGRADECIMIENTOS

A mis padres Rosa María Armbruster García y Jorge Luis Silva Reyes, que siempre me han dado su apoyo incondicional y a quienes debo este triunfo profesional, por todo su trabajo y dedicación para darme una formación académica, pero sobre todo humanista y espiritual. De ellos es este triunfo y para ellos mi mayor agradeciemiento. Este trabajo es solo una pequeña muestra de lo mucho que les agradezco por la educación y conducta que juntos me han brindado. Sabiendo que jamás encontraré la forma de agradecer su constante apoyo y confianza, solo espero que comprendan que mis ideales, esfuerzos y logros han sido también suyos e inspirados en ustedes. Gracias por ayudarme a realizar dos de mis más grandes metas en la vida. Primero, la culminación de mi carrera profesional; segundo, hacerlos sentirse orgullosos de esta persona que tanto los ama. Al amor de mi vida, mi mami que es el ser más maravilloso de todo el mundo. Gracias por el apoyo moral, el cariño y la comprensión que desde niño me has regalado, por guiar mi camino y estar junto a mí en todo momento. A mi papi, porque desde pequeño ha sido para mí un hombre intachable al que siempre he admirado. Día a día he tratado ser una copia fiel de su personalidad, espero por lo menos acercarme. Gracias por guiar mi vida con energía, esto ha hecho que sea lo que soy.

A la niña de mis ojos, mi hermanita Rosy, ha sido uno de los motores impulsores en la superación de mi formación como ser humano. Con la suerte de saber que ella estudiaría en la mayoría de los centros por los que pasé, traté de dejar en cada uno de ellos una pauta, una huella, siempre con metas altas, para que pudiera mejorar mis resultados en su afán de ser como su hermano. Gracias por tenerme siempre como un ejemplo a seguir, por quererme y hablar con tanto orgullo de mí, quizas sin saber lo doblemente orgulloso que estoy yo de tí.

A mi abuela Dora, por ser una fuente de amor interminable, un ser humano espectacular y paciente. Por valorar los mejores rasgos de mi persona y llenarme de afecto de un modo tan dulce y frecuente. Gracias por estar presente siempre en mi vida, esperando y motivando lo mejor de mí.

A los “Vandálicos”, Daniel, Andy, Carlos y Roberto, gracias por aguantarme, puede que a veces haya tenido un carácter un poco recio y en ocasiones hayamos tenido nuestros roces; en ese entonces, gracias por gritarme y dejarme gritar cuando hizo falta. Por volverme a hablar como si nada a los 15 minutos. Juntos hemos encontrado el gran valor de una familia, gracias por ser mis amigos, complices y hermanos.

Forman parte de esta aventura y siempre se quedarán en mis recuerdos, por ayudarme a crecer y encontrar mi lugar en el mundo, por darme muchos días felices llenos de risas y "refresco", jajaja, cuándo hay party!!!. Gracias por dejarme entrar en sus vidas y compartir conmigo un poquito de cada uno de ustedes.

A mis "Lokis", las VIP Yelenis, Analay, Janet y Naylin, gracias por estar ahí en diferentes etapas de mi paso por la universidad, con sus personalidades tan diferentes una de otras, lograron hacer la diferencia. No saben lo mucho que significan para mí, por darle sentido a todas las experiencias que he vivido, por abrirme los brazos desde el principio, por mostrarme que el único modo de ver no es con los ojos abiertos. Gracias por darle un toque cómico, mágico y televisivo a mi vida. Me gustaría agradecerles de todo corazón, pero para ustedes mi corazón está sin fondo.

A mis "Perris", Dianela y Maylevis, gracias por compartir conmigo buenos y malos momentos, fiestas, spaguetis, campismos, en fin, años de mucha alegría. Gracias por valorarme, quererme, por pensar y expresar que soy el tipo de amigo que todos quisieran tener; aunque no estoy seguro de ello, trataré de hacer mi mayor esfuerzo para que sea así.

Gracias a todos mis profesores, Alexander, Vilma, Clara Alonso, Agueda, Valido, Arderí, Iosev, Céspedes, Maurice; en especial a mi tutor Julio Cesar Diaz Vera, por entregar parte de su vida para mi desarrollo, por recibirme como un hijo y verme crecer como persona y como profesional. Gracias por brindarme su amistad, buen ejemplo y confianza para desarrollar este trabajo de tesis.

Espero no haber olvidado a alguien y si lo hice mil perdones. Saben que a todos los quiero mucho y les agradezco por haber hecho de estos 5 años de universidad, los mejores. Así termino mi ingeniería, pero "sigo adelante, a la próxima loca aventura debajo de los cielos".

DEDICATORIA

A mis padres Rosa María Armbruster García, Jorge Luis Silva Reyes y a la memoria de mi abuelo Carlos de Jesús Armbruster Quintana.

RESUMEN

Este documento propone un marco integrado para la eliminación de itemsets no significativos para el usuario en la etapa de pre-procesamiento del minado de reglas de asociación, basado en restricciones de eliminación con la ayuda del conocimiento previo expresado en una ontología de dominio específico. Este último, representa una taxonomía enriquecida por las propiedades de los datos que se utiliza para describir el dominio de aplicación. Definir o actualizar estas propiedades es una tarea poco ardua y no implica cambios en la jerarquía de los artículos o en el nivel de aplicación del marco propuesto. El sistema permite la definición de limitaciones o restricciones de dominio específico, mediante el uso de la ontología para filtrar o eliminar los casos utilizados en el proceso de minado de itemsets frecuentes. Esto puede mejorar el costo computacional de los algoritmos de identificación de itemsets frecuentes con la generación de candidatos. Se describe y valida el marco a través de ejemplos demostrativos de eliminación de itemsets sobre base de datos reales.

PALABRAS CLAVE

Minería de datos, reglas de asociación, ontología de dominio, restricciones de dominio específico.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La capacidad de generar y almacenar información ha crecido considerablemente en los últimos tiempos, se estima que la cantidad de datos almacenados en bases de datos de todo el mundo se duplica cada 20 meses. Hoy las organizaciones tienen gran cantidad de datos almacenados y organizados que no pueden analizar eficientemente en su totalidad. De forma simultánea se han desarrollado diversas herramientas para obtener conocimiento de las distintas fuentes de información que poseen dichas empresas.

Actualmente las bases de datos brindan una herramienta útil para estandarizar el almacenaje de los datos, a partir de ellas se pueden desarrollar diversos mecanismos para encontrar información relevante, novedosa y de gran interés para los usuarios. Estos elementos son muy importantes a la hora de diferenciar entre conocimiento y poder desechar todo aquello que no aporta utilidad al usuario.

El proceso para la extracción del conocimiento para usar una base de datos para cualquier consulta que se requiera; incluyendo: procesamiento, muestreo y transformación, aplicación de técnicas de minería de datos para obtener patrones y la evaluación de los resultados de dicha minería para identificar qué patrones se consideran conocimiento (Frawley, 1992); se encarga de desarrollar las herramientas y procedimientos desde la selección, procesamiento y purificación de los datos, hasta los mecanismos de minería de datos para obtener el conocimiento pasando por su interpretación y evaluación.

La minería de datos, entendida como la búsqueda de patrones dentro de grandes bases de datos utilizando para ello métodos estadísticos y de aprendizaje basado en computadora, está empezando a extenderse en nuestro país en el sector aduanero, telecomunicaciones, financiero y de autoservicio por la necesidad de adquirir alguna solución tecnológica en este campo. Concretamente, en el desarrollo de herramientas para la extracción de reglas de asociación, debido a su sencilla interoperabilidad, se obtiene conocimiento interesante para los usuarios en forma de reglas de asociación que reflejan relaciones entre los atributos presentes en los datos.

Los trabajos enfocados en obtener reglas de asociación con mayor calidad, centran sus esfuerzos en la etapa de pos-procesamiento de datos, que tiene como objetivo el incremento de la comprensibilidad e interés del conocimiento extraído. Aquellos que pretenden disminuir la complejidad computacional prestan especial interés a los algoritmos o a la etapa de pre-procesamiento, que incluye entre sus principales tareas (Pyle): integración de datos, cuando los datos provienen de distintas fuentes; limpieza de datos, que detecta y corrige errores y valores perdidos; discretización, que prepara los datos para algoritmos incapaces de trabajar con datos continuos; y selección de atributos, que en ocasiones está integrada en el propio algoritmo de minería de datos. Esta etapa previamente expuesta constituye en última instancia el interés fundamental de esta investigación.

En este escenario, la dicotomía de la misión de la Aduana de un país entre el control del tráfico de mercancías y la viabilidad del comercio, hace deseable para las mismas poder contar con mecanismos que a partir de datos previos, permitan predecir las cargas con mayor riesgo de fraude para que pasen a examen físico.

La utilización de técnicas de minería de datos que analicen la información histórica y permitan crear modelos predictivos pudiese ser relevante para alcanzar resultados positivos, particularmente la minería de reglas de asociación, que posibilita la obtención de reglas relevantes para la clasificación del riesgo de fraude en las declaraciones de mercancías.

La extracción de reglas de asociación es uno de los campos de investigación, en minería de datos, más activos de los últimos años. Uno de los problemas principales que ha sido objeto de investigación en el área, radica en la gran cantidad de reglas que son generadas y en la dificultad de utilizar una gran parte de las mismas dentro del proceso de toma de decisiones, ya sea porque son obvias, demasiado generales, demasiado específicas o porque no tienen interés para el usuario final.

Comúnmente los esfuerzos realizados en la temática siguen tres direcciones diferentes:

- ✓ Definir mecanismos más eficientes para cubrir el espacio de búsqueda.
- ✓ Explotar estructuras de datos más eficientes.
- ✓ Utilizar conocimiento previo del dominio particular.

Las dos primeras clases de optimizaciones se utilizan para reducir el número de pasos del algoritmo, la reorganización de los conjuntos de elementos, la codificación de los elementos y la organización de las operaciones, con el fin de reducir al mínimo el tiempo de complejidad del algoritmo. La tercera clase trata de superar la falta de datos de exploración por el manejo de restricciones de dominio específico.

Por todo lo anteriormente expuesto se plantea como **problema a resolver:** ¿Cómo disminuir la complejidad computacional de la extracción de reglas de asociación en las declaraciones de mercancías mediante la reducción del tamaño de la instancia a partir de la eliminación de los ítems que no resulten de interés para el usuario final?

Centrándose la investigación en el **objeto de estudio:** Minado de reglas de asociación. Enmarcados en el **campo de acción:** Etapa de pre procesamiento para el minado de reglas de asociación. Definiendo como **objetivo general:** Implementar un algoritmo que sea capaz de eliminar un grupo de ítems, de acuerdo a los intereses definidos por el usuario, de manera que se reduzcan los itemsets candidatos y por tanto la complejidad computacional de la extracción de reglas de asociación.

Los **objetivos específicos** y tareas de la investigación se listan a continuación según la propuesta de (Berndtsson, et al., 2008)**:**

- ✓ Establecer el marco conceptual de referencia.

La idea central es obtener un listado de las tecnologías utilizadas en el desarrollo de algún tipo de algoritmo de eliminación de ítems y decidir si es posible utilizar alguna implementación particular o será necesario desarrollar una propia, ya sea por la adaptación de uno de los existentes o definiendo uno propio. En este sentido se definen un grupo de tareas que son listadas a continuación.

1. Recopilar la bibliografía referente al tema.
2. Seleccionar las fuentes relevantes.
3. Analizar la información de relevancia.

- ✓ Establecer un mecanismo de especificación de los intereses del usuario.

En este punto es necesario definir las siguientes tareas.

1. Analizar las variantes de especificación de los intereses del usuario.

2. Formalizar la variante de representación.

- ✓ Proponer el algoritmo de generalización.
- ✓ Definir el modelo de componentes.

El desarrollo de los artefactos anteriormente expuestos posibilitará contar con el algoritmo de eliminación de ítems que constituye el objetivo primario de la investigación. Para ello deben ser ejecutadas las siguientes tareas:

1. Escoger la estrategia de eliminación.

2. Definir un marco arquitectónico. La arquitectura debe ser documentada de forma tal que se pueda validar la implementación realizada por otros desarrolladores que pudiesen estar interesados en el tema y serían capaces de constatar la efectividad del código realizado y la validez de la solución propuesta.

3. Implementar las funciones asociadas al algoritmo. Desarrollo del algoritmo siguiendo la arquitectura referenciada y documentación del mismo con el objetivo de que se pueda comprobar por otros investigadores la validez de los resultados.

- ✓ Probar la validez del resultado.

Como elemento adicional es necesario demostrar la factibilidad de la utilización de un algoritmo de eliminación de ítems con vista a la reducción de la instancia de entrada a los algoritmos de generación de reglas de asociación. Para ello se deben desarrollar las siguientes tareas:

1. Validar el resultado obtenido.

2. Presentar los resultados.

El resto del informe de la investigación está estructurado como se describe a continuación.

En el **Capítulo 1** se desarrolla el marco conceptual de la investigación, se presentan y se asume una posición de los principales conceptos de minería de datos, extracción de conocimiento y ontologías. Se abordan además las temáticas referentes a la complejidad computacional de la extracción de itemsets candidatos, la utilización de conocimiento previo para disminuir la complejidad computacional de la

generación de itemsets candidatos, el uso de ontologías en la minería de datos y la aplicación de ontologías en las reglas de asociación. Además se realiza un estudio sobre los trabajos relacionados con la investigación.

En el **Capítulo 2**, a través de un caso de estudio sobre las declaraciones de mercancías de las aduanas, se describe todo el proceso que conllevará a la propuesta de solución; se propone un mecanismo de especificación de limitaciones o restricciones del dominio con el apoyo de una ontología de dominio específico para utilizar el conocimiento previo expresado en ella, pretendiendo mejorar la complejidad computacional del proceso de identificación de itemsets frecuentes. Se presenta la implementación de la solución, definiendo las entradas y salidas del algoritmo de eliminación de itemsets, así como sus principales procedimientos. Por último se analiza la complejidad computacional del algoritmo, basándose en la estimación del tiempo de ejecución que consume el mismo para realizar su trabajo.

En el **Capítulo 3** se referencian diferentes métodos de evaluación y validación de la solución propuesta en una investigación. Entre ellos se encuentran los experimentos, pruebas matemáticas, el razonamiento lógico, la demostración, entre otros. La solución propuesta se valida utilizando este último método, la **Demostración**, con el objetivo de demostrar que los resultados de la investigación son confiables y válidos. Para cumplir con este objetivo, se emplean tres casos de prueba, se analizan y discuten los resultados obtenidos y se adopta una posición sobre ellos, demostrando la validez de la solución propuesta. Además se describen los recursos de *hardware* y *software* utilizados para realizar la validación.

1. CAPÍTULO 1: MARCO CONCEPTUAL

1.1 Introducción

El desarrollo informático que experimenta el mundo sitúa considerables empresas en el más alto nivel competitivo de las industrias internacionales, que más allá de subsistir, buscan ser líderes en un ámbito dinámico, diversificado e innovador, siendo precisamente la innovación tecnológica el instrumento con que cuentan para lograr este objetivo. Una característica común en los procesos actuales es el constante y rápido crecimiento de la capacidad para almacenar datos. Día a día se dispone de mayores volúmenes históricos de datos empresariales que contienen información acerca de dichos procesos. Si bien la información reduce la incertidumbre y permite por consiguiente tomar mejores decisiones, también se hace necesario el uso de herramientas que permitan extraer conocimiento útil a partir de grandes conjuntos de datos, debido al aumento de la cantidad de datos almacenados y la disminución de la capacidad para asimilarlos.

La minería de datos es un área de investigación que pretende dar respuesta a esa necesidad de procesar y analizar dichas masas de datos, con el fin de encontrar y aprovechar el conocimiento oculto en ellos. Básicamente, es un proceso conducido por un problema: la respuesta a una pregunta o solución a un problema, se busca analizando los datos disponibles.

1.2 Minería de datos

Habitualmente los investigadores han usado de manera indiferente los conceptos de minería de datos y extracción de conocimiento. En la investigación se dejará claro que aunque ambos conceptos están angostamente relacionados no significan exactamente lo mismo. Muchos estudios han definido el proceso de extracción de conocimiento (KDD, por sus siglas en inglés), sin embargo uno de los más referenciados aparece en (Fayyad, 1996), donde se define como: "*proceso no trivial de identificar patrones en los datos que sean válidos, novedosos, potencialmente útiles y comprensibles",* además se encuentra la siguiente definición: *"proceso para usar una base de datos para cualquier consulta que se requiera; incluyendo: pre-procesamiento, muestreo y trasformaciones, aplicación de técnicas de minería de datos para obtener patrones y la evaluación de los resultados de dicha minería para identificar qué patrones se consideran conocimiento"*.

Como bien se enuncia en la disquisición anterior la minería de datos es una técnica o fase dentro del proceso de extracción de conocimiento, la misma se destaca en la siguiente imagen:

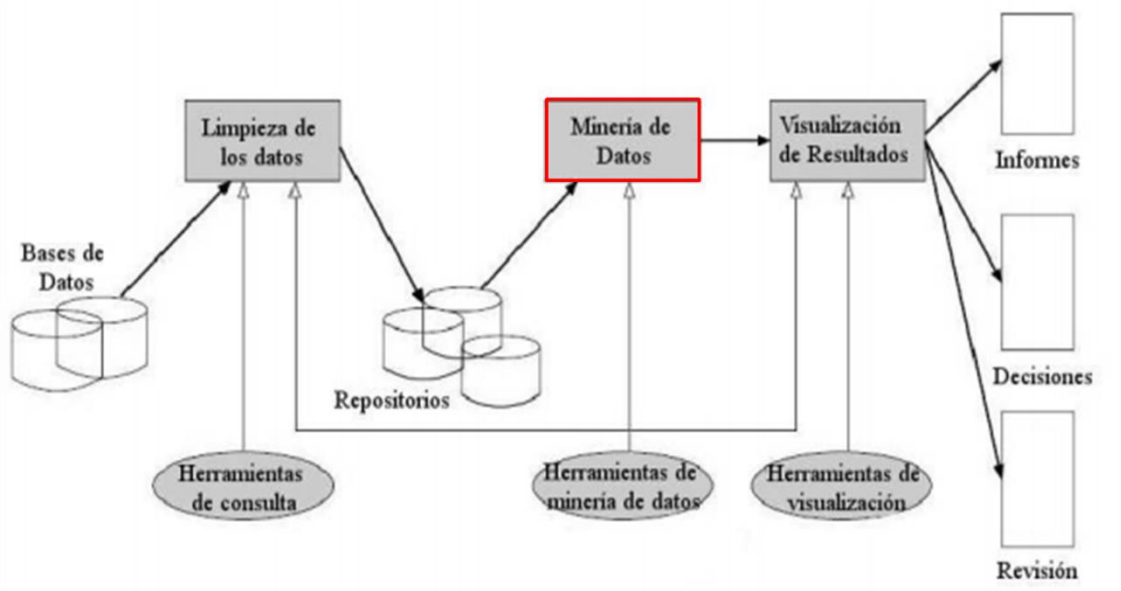

Figura 1. Resumen de los pasos que componen un proceso KDD.

La minería de datos es una de las fases de mayor peso e importancia dentro del proceso de extracción de conocimiento, de ahí que los autores usen ambos conceptos de manera indiscriminada.

Las definiciones que ha arrojado el devenir del tiempo en esta área del conocimiento sobre minería de datos son muy variadas, entre ellas están:

"La minería de datos (DM, Data Mining) consiste en la extracción no trivial de información que reside de manera implícita en los datos. Es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos" (Hernández Orallo, 2004).

"La minería de datos es el proceso de examinar exhaustiva y minuciosamente inmensas cantidades de datos a fin de identificar, extraer y descubrir nuevos conocimientos, de forma automática. Es una

herramienta exploradora y no explicativa. *Es decir explora los datos para sugerir hipótesis"* (Galvis Mario, 2004).

La minería de datos también puede ser entendida como: "*proceso analítico para explorar grandes volúmenes de datos (generalmente de negocio y mercado) con el objetivo de descubrir patrones y modelos de comportamiento o relaciones entre diferentes variables"* (Galvis Mario, 2004).

A pesar de que ninguno de los conceptos anteriores son errados, la mayoría de los estudios más recientes parecen estar de acuerdo con la expresada en (Triantaphyllou, 2010), donde la minería de datos se define como: *"Familia de métodos computacionales que tienen como objetivo recolectar y analizar datos relacionados a un sistema de interés con el propósito de ganar un mejor entendimiento del mismo".*

Según (Fayyad, 1996), la minería de datos tiene dos grandes tareas u objetivos principales: la predicción y la descripción, sin embargo hay autores que dividen estos objetivos en dos disciplinas distintas, considerando la primera como el núcleo principal del aprendizaje automático y la segunda una disciplina del dominio. Estos conceptos no son excluyentes y es necesario distinguirlos:

- ✓ **La predicción** involucra el uso de varias variables en una base de datos para intentar determinar los valores desconocidos de otras variables o para adivinar valores futuros que puedan tener las propias variables.

Para alcanzar este objetivo existen diferentes técnicas o tareas. A continuación se relacionan algunas de las más usadas:

1. **La clasificación** tiene como objetivo pronosticar el valor o la clase que puede tomar un atributo concreto en función de los valores que toman otros atributos. Para ello existen diversos modelos para representar diferentes tipos de clasificación: conjunto de reglas, listas de decisión y árboles de decisión.
2. **La regresión** es una clasificación con consecuente numérico, es decir pretende pronosticar qué valor numérico tiene un atributo en particular.

- ✓ **La descripción** está centrada en encontrar patrones interpretables por los humanos que expliquen de alguna manera la naturaleza de los datos.

Para alcanzar el objetivo previamente descrito existen, entre otras, las siguientes tareas:

1. **El agrupamiento o *clustering*** tiene como objetivo conseguir formar grupos de individuos (tuplas) de acuerdo a sus similitudes.
2. **Las reglas de asociación** tienen como objetivo identificar relaciones que pudieran existir entre atributos categóricos.

Durante el desarrollo de la investigación se presenta especial interés en esta última tarea. La minería de reglas de asociación de bases de datos ha despertado especial interés a los investigadores de esta área del conocimiento, debido a la formidable utilidad de su aplicación en diferentes dominios. La idea de su estudio surgió en el contexto del análisis de las cestas de compras, donde se estudian las compras conjuntas de varios artículos en una determinada tienda. Las correlaciones entre algunos elementos particulares podrían favorecer a la organización de las estrategias de *marketing,* publicidad de artículos y planificación de los almacenes.

1.3 Reglas de asociación

El problema de encontrar reglas de asociación es introducido por (Agrawal, 1993). Dado un conjunto de transacciones, donde cada transacción es un conjunto de literales (llamadas elementos), una regla de asociación es una expresión de la forma $X \rightarrow Y$, donde X e Y son conjuntos de los artículos.

Según su definición original, el problema de minería de reglas de asociación se define como:

Sea $I = \{i_1, i_2, \dots, i_n\}$ un conjunto de n atributos binarios llamados ítems.

Sea $D = \{t_1, t_2, \dots, t_m\}$ un conjunto de transacciones almacenadas en una base de datos.

Donde cada transacción en D tiene un identificador único y contiene un subconjunto de ítems de I. Una regla se define como la implicación de la forma:

$$X \rightarrow Y, donde\ X \subset I, Y \subset I\ y\ X \cap Y = \emptyset$$

El significado intuitivo de esta norma es que las transacciones de la base de datos que contienen X tienden a contener Y. Los conjuntos de ítems X e Y se denominan respectivamente "antecedente" o parte izquierda y "consecuente" o parte derecha de la regla.

De forma general, X e Y serán dos conjuntos disjuntos de ítems: $\{x_1, x_2, \dots, x_n\}$ y $\{y_1, y_2, \dots, y_n\}$ respectivamente, donde $x_i, y_j \in I$ para todo $i \in \{1, \dots, n\}$ y $j \in \{1, \dots, m\}$.

Muy a menudo las bases de datos contienen miles o incluso millones de registros. Una regla de asociación necesita un soporte de varios cientos de registros o transacciones antes de que ésta pueda considerarse significativa desde un punto de vista estadístico. Para seleccionar reglas interesantes del conjunto de todas las reglas posibles que se pueden derivar de un conjunto de datos, se pueden utilizar restricciones sobre diversas medidas de "significancia" e "interés".

1.3.1 Medidas de interés de las reglas de asociación

Uno de los principales problemas en el descubrimiento de reglas de asociación es el desarrollo de una medida que permita evaluar la significación de las reglas descubiertas. Una regla que es interesante para un usuario pudiera no serlo para otro, por lo cual resulta muy difícil definir una medida de interés ideal. No obstante, una medida de interés objetiva que se base en métodos estadísticos o lógicos ayuda a eliminar las reglas innecesarias y reducir el espacio de búsqueda.

Las restricciones más conocidas son los umbrales mínimos de **"soporte"** y **"confianza"**, considerándose interesantes las que superen valores de umbral asociados.

El soporte de un conjunto de ítems o itemsets es el porcentaje de transacciones de D que contienen al itemset:

$$sop\ X = \frac{|\ t \in D \quad X \subseteq t\}|}{|D|}$$

Dicho soporte es una estimación de la probabilidad de que una transacción en D contenga al itemset. Usualmente es usado como una medida absoluta que indica el número de transacciones que satisfacen al itemset.

El soporte de una regla de asociación es el porcentaje de transacciones de D que contiene $X \cup Y$:

$$Sop\ X\ = \frac{|\ t\ \in D\ \ X \cup Y \subseteq t\}|}{|D|}$$

La confianza de la regla es el porcentaje de transacciones de D que contienen $X \cup Y$ de entre todas aquellas que cumplen el antecedente:

$$Conf\ X\ = \frac{|\ t\ \in D\ \ X \cup Y \subseteq t\}|}{|\ t\ \in D\ \ X \subseteq t\}|}$$

La confianza puede interpretarse como un estimador de $P(\ Y\ \ X\ )$, la probabilidad de encontrar la parte derecha o consecuente de una regla condicionada a que se encuentre también la parte izquierda o antecedente (Agrawal, 1993).

Un ejemplo de una regla de asociación es:

"30% de las transacciones que contienen cerveza también contienen pañales; 2% de todas las transacciones contienen estos elementos".

Aquí 30% se denomina confianza de la regla, y el 2% es el soporte de la regla. Tanto el lado izquierdo como el lado derecho de la regla pueden ser conjuntos de elementos. El problema es encontrar todas las reglas de asociación que satisfagan las especificaciones del usuario de soporte mínimo y mínimo de restricciones de confianza en contra (Ramakrishnan Srikant).

El **"factor de certeza"** es otra medida de interés que en el marco de las reglas de asociación fue propuesto por (Berzal F., 2001). Según su definición el factor de certeza (*certainty factor*) de una regla $X \rightarrow Y$ se expresa de la siguiente forma:

$$CF\ X\ = \begin{cases} \dfrac{Conf\ X\ - sop(Y)}{1 - sop(Y)}, si\ Conf\ X\ > sop(Y) \\ \dfrac{Conf\ X\ - sop(Y)}{sop(Y)}, si\ Conf\ X\ < sop(Y) \\ 0, en\ otro\ caso \end{cases}$$

Esta medida expresa (en forma de índice) el porciento del incremento (o decremento) de la probabilidad condicional de Y respecto a X relativo a la magnitud del intervalo definido por la probabilidad a priori de Y. Esta medida se expresa en el intervalo $-1,1$, indicando los valores positivos el incremento relativo de la probabilidad condicional, los negativos el decremento, y el valor 0 la independencia entre el consecuente y el antecedente.

La **"confianza neta"** fue propuesta en (Ahn, 2004), con el propósito de estimar la importancia del consecuente de una regla respecto a su antecedente. Su definición puede enunciarse mediante la expresión:

$$netconf\ X\ = Conf\ X\ - Conf(-X)$$

Tras esta definición se tiene la siguiente consideración, cuando Y es frecuente en transacciones que no contienen a X, la regla $X \rightarrow Y$ suele no ser interesante aún cuando $Conf\ X$ es alta. La confianza neta permite resolver esta situación. El dominio de esta medida es el intervalo $-1,1$, lo cual se puede deducir de su formulación. La confianza neta permite medir la independencia entre consecuente y antecedente. Esto puede verificarse con las siguientes transformaciones:

$$netconf\ X\ = \frac{Sop(X)}{sop(X)} - \frac{sop\ Y\ - Sop(X)}{1 - sop(X)} = \frac{Sop\ X\ - sop\ X\ * sop(Y)}{sop\ X\ * (1 - sop(X))}$$

Según la expresión anterior, cuando $netconf\ X\ = 0$ se verifica la independencia entre esos itemsets, ya que $Sop\ X\ = sop\ X\ * sop(Y)$.

También se puede verificar que la confianza neta permite discriminar la dirección de la dependencia, al comprobarse que usualmente $netconf\ X\ \neq netconf(Y)$. Este indicador permite medir la fuerza de la implicación en ambas direcciones. No obstante, tal afirmación no es evaluada en toda su dimensión.

Otra propiedad que resalta en (Ahn, 2004), es que si $netconf\ X$ es positivo entonces la dependencia es positiva y si es negativa la dependencia también es negativa. Por otra parte, muestran que la confianza neta permite medir la dependencia positiva y negativa en una escala simétrica respecto a

$Sop\ X$. Estas dos propiedades confieren a esta medida características interesantes cuando se precise analizar dependencias negativas, limitaciones presentes en el factor de certeza.

1.4 Complejidad computacional de la extracción de itemsets candidatos

Muchos estudios en el área de la extracción de reglas de asociación con el objetivo de encontrar algoritmos eficientes en la extracción de itemsets candidatos, han prestado especial interés en analizar su complejidad computacional. Estos, al tratar con conjuntos de datos muy grandes y dispersos (tienen un promedio de ítems por transacción muy pequeño, por debajo del 30%, con respecto al total de ítems), por ejemplo, analizando colecciones de documentos, tardan mucho en dar respuesta o simplemente no pueden procesarlos. Cuando se dice conjuntos de datos muy grandes, se habla de conjuntos de datos con millones de ítems diferentes y/o millones de transacciones. La demora de los algoritmos existentes para procesar estos conjuntos se debe, fundamentalmente, a la presencia de miles de ítems diferentes (en ocasiones millones) y/o al uso de un umbral de soporte muy bajo.

Los métodos de generación de los itemsets se diferencian por las estrategias de recorrido en los espacios de estado y las formas de representación de esos conjuntos de ítems. De manera general, existen dos estrategias para recorrer el espacio de búsqueda de los ítems frecuentes: en amplitud y en profundidad. Los algoritmos que siguen una estrategia en amplitud necesitan generar todos los conjuntos de K-ítems antes de generar los conjuntos de $K+1$ -ítems. Si la cantidad de ítems es muy grande o el umbral de soporte es muy bajo, se generan muchos conjuntos candidatos a ser frecuentes, lo cual puede ser imposible de mantener en memoria. Por tanto, si se tienen n ítems diferentes y un soporte cercano a cero, la cantidad de itemsets candidatos tiende a 2^n.

Los algoritmos que siguen una estrategia en profundidad basan sus estructuras en árboles para almacenar todo el conjunto de datos en memoria; si los conjuntos de datos son muy grandes, las estructuras resultantes también lo son y los recorridos sobre estas son muy costosos en tiempo.

Algunos de los algoritmos más destacados en la extracción de itemsets candidatos son:

SETM (*Set-Oriented Mining*). Este algoritmo se propuso para el minado de reglas de asociación mediante el uso de operaciones relacionales en un entorno de base de datos relacionales, motivado

por el deseo de utilizar el lenguaje de consulta estructurado (SQL, por sus siglas en inglés) para calcular conjuntos de elementos frecuentes.

El siguiente estudio (Agrawal, 1994) presentó tres nuevos algoritmos:
Apriori, propuesto bajo el principio de que si un itemset X no es frecuente, tampoco lo es ningún otro que lo contenga y su contra recíproco, es decir, si X es un itemset frecuente también lo es cualquiera de sus subconjuntos. Por ser el algoritmo básico por excelencia en la minería de reglas asociación, ha sido objeto de innumerables ajustes con el propósito de mejorar su complejidad computacional.

AprioriTid que básicamente no reduce el número de candidatos, su principal aporte es que no utiliza la base de datos D para contar el soporte luego del primer pase, lo que agiliza el proceso de manera considerable al no tener que escanear una y otra vez los datos. AprioriTid muestra una mejora potencial en cuanto a la cantidad de escaneos sobre los datos en comparación con el Apriori tradicional, mejorando así el desempeño computacional en la extracción de reglas de asociación.

AprioriHybrid combina los algoritmos Apriori y AprioriTid, aprovechando la mayor eficiencia del primero en los primeros recorridos por el conjunto de datos y la eficacia del segundo, a partir de cierto número K de escaneos, cuando el conjunto de registros puede almacenarse en la memoria, lo que es posible porque en cada nivel ambos algoritmos generan los mismos conjuntos candidatos.

Otro estudio importante en el campo de la minería de reglas de asociación se describe en (Savasere, 1995). Estos autores presentan un algoritmo llamado **Partition** (Partición) que reduce la búsqueda calculando primero los ítems frecuentes en dos pasadas por la base de datos, sin embargo posteriormente en (Toivonen, 1996) se propone una sola pasada completa. Partition divide las transacciones de la base de datos en tantas partes disjuntas (o sea, en particiones no necesariamente de iguales tamaños) como fueran necesarias para poder almacenarlas en la memoria operativa. La idea principal consiste en seleccionar una muestra aleatoria, y utilizarla para determinar reglas de asociación representativas, que es muy probable que ocurran también en la base de datos completa.

En (Triantaphyllou, 2010) se presenta el **RA1** cuya complejidad es de tiempo polinomial debido a que en su núcleo contiene el algoritmo Apriori, ventaja que aprovecha el mismo para dar lugar a variaciones como la **ARA1**, con excelentes resultados en comparación con sus predecesores.

1.4.1 Experimento de comparación computacional

A continuación se muestra una comparación computacional entre los algoritmos Apriori y ARA1 presentada en (Triantaphyllou, 2010) con la ayuda de una computadora central IBM 9672/R53. Este procesador posee un cuadro de motor-10 donde cada motor está clasificado en 26 MIPS (millones de instrucciones por segundo). Se emplearon para la ejecución los siguientes juegos de datos:

Base de datos 1: {1 000 ítems con 100 000 transacciones}, el soporte mínimo se estable en 250.

Base de datos 2: {1 000 ítems con 10 000 transacciones}, el soporte mínimo se estable en 25.

Utilizando la (Base de datos 1) como entrada, el algoritmo Apriori después de 80 horas 22 minutos y 8 segundos estaba todavía en el proceso de generación de los conjuntos de ítems frecuentes de longitud 2; debido a la demora en el procesamiento de los datos el experimento fue abortado.

Sin embargo, el algoritmo de minería ARA1 completó la ejecución sobre la misma base de datos (Base de datos 1), en solo 44 horas, 22 minutos y 1 segundo. El algoritmo ARA1 extrae una sola regla con cada uno de los niveles de apoyo siguientes: 259, 263, 308, 441, 535, 623, 624, 756, 784, 984 y 1093.

Para los experimentos con la (Base de datos 2), se pusieron en práctica algunas de las técnicas de computación paralela para la ejecución del Apriori. Los conjuntos de ítems frecuentes fueron reunidos en grupos más pequeños, lo que permitió construir los mismos en un tiempo medianamente corto. Como resultado, cada grupo de ítems se analizó por separado, y los tiempos de la Unidad Central de Procesamiento (CPU, por sus siglas en inglés) para cada uno de estos trabajos se sumaron al final.

El algoritmo Apriori terminó en 59 horas 15 minutos y 3 segundos.

Por otro lado, el algoritmo ARA1 solo demoró 2 horas 54 minutos y 57 segundos (Triantaphyllou, 2010).

Estos resultados se representan en la (Figura 2 y 3) respectivamente.

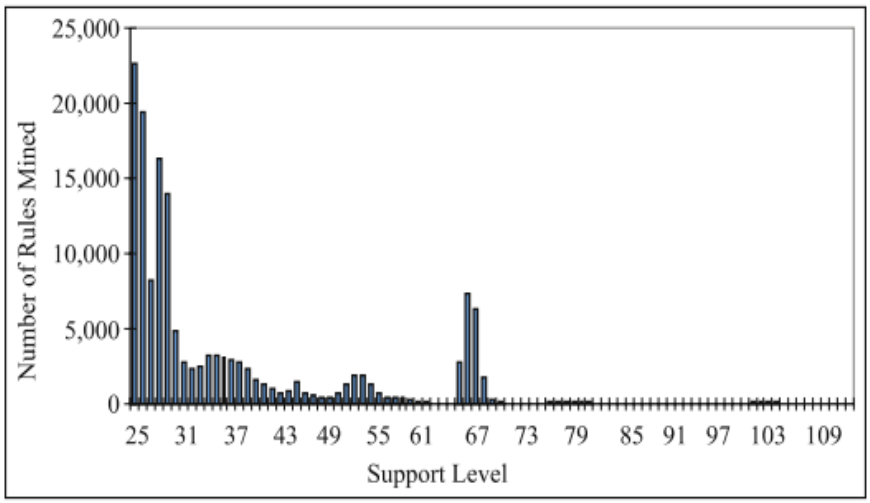

Figura 2. Histograma de los resultados del Apriori en la base de datos 2 (Triantaphyllou, 2010).

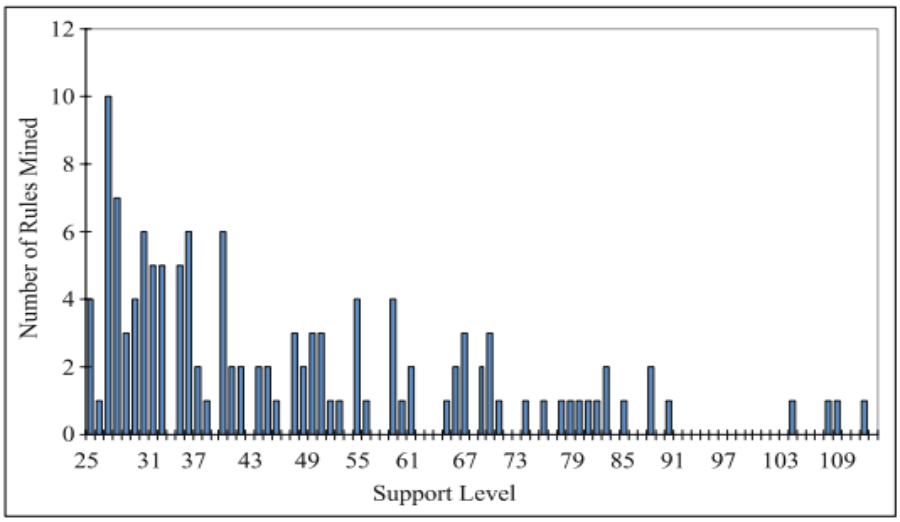

Figura 3. Histograma de los resultados del ARA1 en la base de datos 2 (Triantaphyllou, 2010).

Estos estudios concluyen que cuando se utilizan técnicas estándar de minería de reglas de asociación como Apriori y sus variantes AprioriLike, puede causar el consumo de recursos exponenciales en el peor de los casos, pues en el proceso de generación de itemsets frecuentes, el tamaño de los conjuntos de itemsets frecuentes, y por lo tanto de sus conjuntos candidatos, en las iteraciones iniciales es mucho mayor que los de las iteraciones avanzadas. Esta observación muestra la

conveniencia de disminuir al máximo los conjuntos de candidatos en las fases iniciales, especialmente la de 2 itemsets.

Otro factor que atenta contra el rendimiento del proceso de generación de itemsets frecuentes es la cantidad de datos que es necesario procesar para determinar el soporte de los itemsets candidatos, ya que a medida que el proceso de generación de itemsets frecuentes avanza, no solo se reducen los conjuntos de itemsets frecuentes, sino también la cantidad de transacciones que contienen a los itemsets de cada iteración. Luego, la eliminación tanto de aquellas transacciones que no serán capaces de contener a itemsets mayores, como de los ítems que contienen las transacciones y que no formarán parte de ningún itemsets frecuente de tamaño superior, debe incrementar el rendimiento de un algoritmo de cálculo de itemsets frecuentes.

Actualmente son deseables grandes bases de datos para la obtención de resultados precisos, pero desafortunadamente, la eficiencia de los algoritmos depende en gran medida del tamaño de la instancia de entrada. En consecuencia, es muy conveniente desarrollar un algoritmo que tenga complejidad polinómica y aun así sea capaz de encontrar reglas de buena calidad.

1.5 Ontologías

El término ontología lo introdujo Thomasius y Wolf, en el siglo XVII, para designar, precisamente, en la filosofía primera de Aristóteles, la parte de la Metafísica que estudia el ser en cuanto tal. Sin embargo, el origen epistemológico del término se encuentra mucho antes, en la disciplina filosófica surgida con Parménides (h. 540-h. 450 a.C.), cuyo objetivo era también, el estudio del ser. “La ontología se ocupa de las categorías generales del ser, entendidas de forma abstracta, de las que participa el ser concreto” (Céspedes).

La literatura de Inteligencia Artificial contiene varias definiciones de ontología. Los investigadores en el área acogieron el término Ontología para describir todo lo que puede ser representado computacionalmente. Ramos y Núñez plantean que en este campo “*Una ontología es una especificación formal y explícita de una conceptualización compartida”* (Núñez, 2007).

A principio de los años 90 las ontologías comenzaron a utilizarse en el campo de la informática para representar conocimiento y procesar el lenguaje natural. Hendler plantea que *"Una Ontología define los términos a utilizar para describir y representar un área de conocimiento. Las Ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento, tales como medicina, fabricación de herramientas […], etc.). Las Ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son útiles para los ordenadores"* (Hendler, 2004).

En la ciencia de la computación y la ciencia de la información, una ontología "*es una representación formal de un conjunto de conceptos dentro de un dominio y de relaciones entre dichos conceptos"*. Es usada para razonar acerca de propiedades de un dominio, y quizás usada para definir el dominio. Una ontología brinda un vocabulario compartido, el cual puede ser usado para modelar un dominio, que es, el tipo de los objetos y/o conceptos que existen, sus propiedades y relaciones (Cruz Concepción).

Según los criterios antes analizados se define que una ontología no es más que una forma de modelar términos que se manejan en una empresa o entidad, lo que hace posible una comprensión común de la estructura de la información entre personas o agentes de software, es decir que los términos con los que se trabajan tengan el mismo significado, permitiendo hablar en un lenguaje común.

1.5.2 Tipos de ontologías

Las ontologías se clasifican por su dependencia del contexto en:

- ✓ **Ontologías de Nivel Superior:** Describen conceptos muy generales que son independientes de un problema particular o dominio. Por ejemplo, ontologías sobre el tiempo, el espacio, la materia, el objeto, el acontecimiento, la acción, entre otros.
- ✓ **Ontologías de Dominio:** Proporcionan el vocabulario necesario para describir un dominio dado. Incluyen términos relacionados con los objetos del dominio y sus componentes, un conjunto de verbos o frases que dan nombre a actividades y procesos que tienen lugar en ese dominio, y conceptos primitivos que aparecen en teorías, relaciones y fórmulas que regulan o rigen el dominio.

- ✓ **Ontologías de Tareas:** Proveen un vocabulario sistemático de los términos usados para resolver problemas asociados con tareas particulares, ya sean dependientes o no del dominio.
- ✓ **Ontologías de Aplicación:** Contienen las definiciones necesarias para modelar el conocimiento requerido para una aplicación particular en un dominio dado. Describen conceptos que dependen tanto del dominio particular como de las tareas. Estas ontologías son especializaciones de las ontologías de dominio y de tareas.

En cuanto a la granularidad de la conceptualización (cantidad y tipo de conceptualización) se clasifican en:

- ✓ **Ontologías Terminológicas:** Especifican los términos que son usados para representar conocimiento en el universo de discurso. Suelen ser usadas para unificar vocabulario en un dominio determinado (contenido léxico y no semántico).
- ✓ **Ontologías de Información:** Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información (estructura de los registros de una base de datos).
- ✓ **Ontologías de representación de conocimiento o Meta-ontologías:** Especifican la conceptualización que subyace en un paradigma o formalismo de representación de conocimiento, es decir, proporcionan el vocabulario necesario para modelar otras ontologías utilizando un determinado paradigma de representación de conocimiento.

Se clasifican por su propósito de uso en:

- ✓ **Ontologías para la comunicación entre personas:** Proporcionan los términos necesarios para describir y representar un área de conocimiento. Una ontología informal (no ambigua) puede ser suficiente.
- ✓ **Ontologías para la interoperabilidad entre sistemas:** Permiten realizar traducciones entre diferentes métodos, paradigmas, lenguajes de construcción y herramientas de software. La ontología se usa como un formato de intercambio de conocimiento.

- ✓ **Ontologías para beneficiar la ingeniería de sistemas:** Favorecen la reutilización de componentes, facilitan la adquisición de conocimiento e identificación de requerimientos, y aumentan la fiabilidad de los sistemas al proporcionar consistencia en el conocimiento utilizado.

El nivel de formalidad se refiere al grado de formalismo del lenguaje usado para expresar la conceptualización, en este ámbito se clasifican en:

- ✓ **Ontología altamente informal:** Expresada en lenguaje natural (Glosario de términos).
- ✓ **Ontología informal estructurada:** Expresada en una forma restringida y estructurada de lenguaje natural, que permite incrementar la claridad y reducir la ambigüedad.
- ✓ **Ontología semi-formal:** Usa un lenguaje de definición formal.
- ✓ **Ontología rigurosamente formal:** La definición de términos se lleva a cabo de manera meticulosa usando semántica formal y teoremas (Núñez, 2007).

De acuerdo a los objetivos de la investigación, ésta se interesa por las **ontologías de dominio** previamente descritas en la clasificación por dependencia del contexto.

Las ontologías permiten utilizar la información pre-existente en las bases de datos sin tener que renunciar a los datos originales de partida, de manera que pueden convivir y seguir empleándose simultáneamente, pero se debe tener presente la necesidad de actualizar la ontología con los nuevos datos que vayan añadiéndose a las bases de datos. Para esto no es necesario introducir los cambios manualmente (aunque podría hacerse) en la nueva estructura que hemos desarrollado con la ontología. Se puede automatizar el proceso de volcado de la información mediante el desarrollo informático de procedimientos que transformen el formato de la base de datos inicial en el formato y lugar adecuado en la nueva estructura ontológica. Lógicamente, cuanto más estructurada se halle la información inicial, más sencilla es la transformación correspondiente.

1.6 Utilización de ontologías en la minería de datos

Cuando se realiza un proceso de minería de datos, se necesita tener en cuenta el conocimiento previo; este puede derivar del proceso mismo (elección de variables, técnicas, algoritmos, interpretación de resultados) o del dominio de aplicación. Actualmente, en la mayor parte de proyectos de KDD, el

conocimiento previo está exclusivamente presente de manera implícita en la mente de un analista humano o en la forma de documentación textual. A menudo, los conocimientos previos no son organizados alrededor de un modelo conceptual gramaticalmente correcto. Esta práctica parece no hacer caso del último desarrollo en la Ingeniería de Conocimiento, donde el conocimiento del dominio es típicamente definido por ontologías formales.

En ambientes distribuidos, las ontologías son usadas para construir un servicio semánticamente rico en descripciones. Técnicas para planificación, composición, edición, razonamiento y el análisis sobre estas descripciones, están siendo investigadas y desplegadas para resolver la interoperabilidad semántica entre servicios (Cannataro, 2003).

Por los criterios previamente expuestos, se afirma que uno de los problemas más importantes y desafiantes a ser investigados en minería de datos es la representación del conocimiento previo. Considerando la necesidad de incluir este conocimiento dentro del proceso de descubrimiento por medio de las ontologías, se expresa que la base ontológica es una condición previa para el uso automatizado eficiente de ese conocimiento.

Por estas razones las ontologías fueron introducidas en las técnicas de minería de datos alrededor del año 2000. Dadas sus características que les permiten ser una eficaz herramienta de comunicación automática entre computadoras y/o humanos para el razonamiento, representación y reutilización del conocimiento, se han convertido en una precondición necesaria para una eficiente aplicación de estas técnicas. Las Ontologías de Dominio de Conocimiento o de Conocimiento Previo organizan los dominios de conocimiento, además de ser de gran importancia en varios niveles del proceso de descubrimiento de conocimiento, las Ontologías de Metadatos describen el proceso de construcción de ítems y las de Procesos de Minería de Datos codifican el proceso de minado, contribuyendo a la elección de la tarea más apropiada según el problema a resolver (Marinica).

En la (Figura 4) se muestran los espacios donde son aplicables las ontologías en el proceso de descubrimiento de conocimiento.

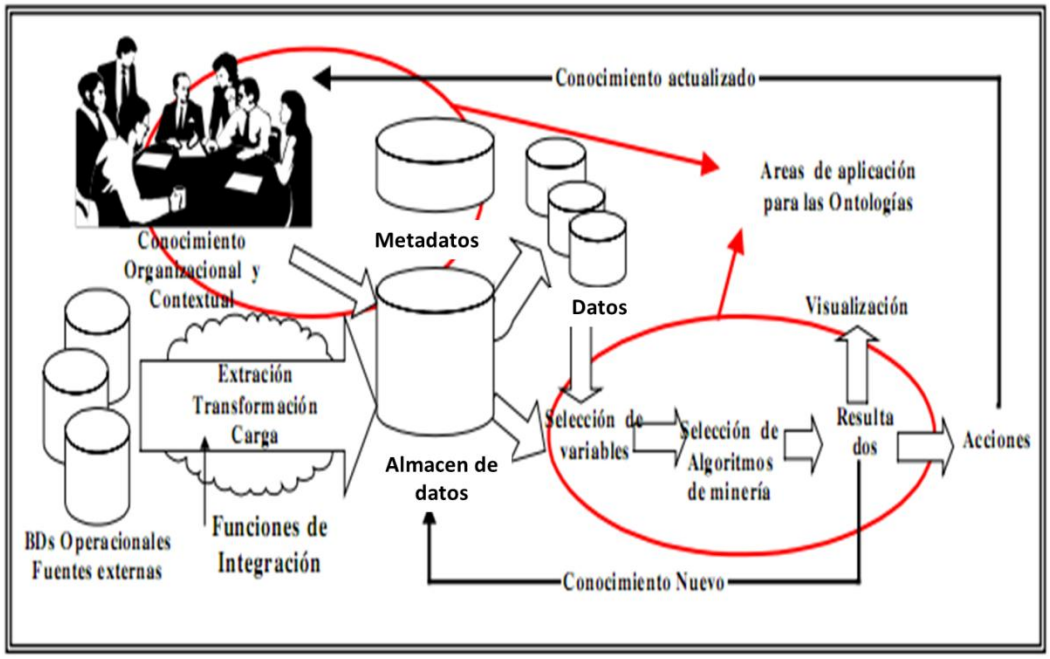

Figura 4. Espacios de aplicación para las ontologías en el proceso de KDD.

De este modo, se establece una relación bidireccional entre ontologías y minería de datos:

- ✓ **De ontologías a minería de datos:** se incorpora al proceso el conocimiento previo a través de ontologías, además de la interpretación y validación del conocimiento minado.
- ✓ **De minería de datos a ontologías:** se incluye el conocimiento como una instancia de entrada de información o como el resultado del proceso, entonces el análisis se hace sobre la ontología (Cannataro M., 2007).

1.7 Utilización de ontologías en las reglas de asociación

Según (Xiong, 2010) el mínimo soporte y confianza de un número de reglas de asociación fuertes extraídas, hace que el análisis y consecuente utilización de estas sea difícil. Xiong propone un índice de calidad de las reglas de asociación basado en la integración de aspectos objetivos y subjetivos con el fin de cuantificar la calidad de las reglas de asociación basados en una Ontología de Dominio. En la construcción de la ontología se tienen en cuenta el dominio de conocimiento, y el conocimiento previo, combinado con el conocimiento experto y la familiarización de los usuarios con el conjunto de datos. El

minado se hace bajo la guía de la ontología y las reglas derivadas cumplen con los parámetros deseados además de expresar el propósito de los usuarios.

En este contexto, la metodología de emparejamiento AROMA (*Association Rule Ontology Matching Approach,* por sus siglas en inglés) (Jerome, 2007) tiene como objetivo encontrar las relaciones de inclusión y equivalencia entre entidades (clases o propiedades) de dos estructuras jerárquicas diferentes proporcionadas por datos textuales. Las ontologías se definen como vocabularios estructurados que describen las relaciones entre conceptos. El lenguaje OWL permite describir conceptos (llamado clases) y organizarlos en taxonomías (de clases y de propiedades) mediante el uso de la relación inclusión.

1.8 Trabajos relacionados

Los estudios relacionados para eliminar itemsets de poca relevancia para los usuarios en la etapa de pre-procesamiento para la extracción de reglas de asociación con el apoyo de una ontología han sido escasos, el más destacado aparece en: ***Pushing constraints in association rule mining: anontology-based approach*** (Bellandi, 2007). Este estudio propone un marco integrado para la extracción de restricciones basadas en reglas de asociación multi-nivel con la ayuda de una ontología. Esta última representa una taxonomía enriquecida que se utiliza para describir el dominio de aplicación gracias a las propiedades de los datos ya que definir o actualizar las mismas, no es una tarea muy complicada y no implica cambios en la jerarquía de los artículos, o en el nivel de aplicación del marco. El mecanismo propuesto permite la definición de restricciones en el dominio específico, mediante el uso de la ontología para filtrar los casos utilizados en el proceso de minería de reglas de asociación. El objetivo de estas limitaciones es excluir algunos artículos de las reglas de asociación extraídas. Mediante su empleo, se puede especificar un conjunto de elementos que pueden ser excluidos de la entrada de operaciones, y en consecuencia, de la extracción de las reglas. Este tipo de limitaciones se refiere ya sea a un solo elemento, o un concepto de ontología, y pueden incluir una condición expresada en un conjunto de propiedades de la ontología.

Otro estudio, que aunque no relaciona el uso de las ontologías, basa sus fundamentos en la eliminación de itemsets y transacciones que no cumplen con ciertas condiciones que se exponen a continuación.

El algoritmo **DHP** (*Direct Hashing and Pruning*, por sus siglas en inglés) emplea una técnica de *hashing* para eliminar itemsets innecesarios en la generación del conjunto de candidatos, considerando a su vez heurísticas para la reducción efectiva del tamaño de las transacciones y la disminución del número de transacciones a procesar (Park J.S., 1997); (Holt J.D., 1999).

Cuando se explora la base de datos para calcular el soporte de cada K-itemset candidato, se generan informaciones sobre cada $(K+1)$ itemset elegible como candidato en adelanto de forma tal que todos los posibles $(K+1)$ itemsets de cada transacción, tras cierto filtraje, son acumulados en una tabla de *hash*.

En cada celda de la tabla de *hash* se cuentan los $(K+1)$ itemsets con iguales valores de *hash*. Es importante que, debido a las colisiones, pueden contarse itemsets diferentes en una misma celda de la tabla de *hash*. Por lo tanto, el valor de cualquier celda será la suma de los soportes de todos los itemsets con iguales valores de *hash*. Lógicamente, dicho valor de celda representa una cota superior de los soportes de los itemsets asociados. De esta forma, cada $K+1$ itemset posible será considerado como candidato en la siguiente pasada si su valor en la tabla de *hash* es mayor que el soporte mínimo.

Para reducir la dimensión de la base de datos, este algoritmo propone una heurística que se basa en la propiedad de la clausura descendente del soporte de los itemsets. Esta propiedad sugiere que una transacción puede contener un candidato $(K+1)$-itemset solo si contiene $K+1$ -itemsets candidatos de tamaño K (K-itemsets) obtenidos en la iteración anterior. Luego, si una transacción es evaluada para contar las ocurrencias de los K-itemsets candidatos, entonces puede determinarse si esa transacción puede ser eliminada (podada) de la base de datos en la pasada siguiente. Por otra parte, si una transacción contuviera un $K+1$ -itemset frecuente, entonces cualquier ítem contenido en ese $K+1$ -itemset debería aparecer en al menos K de los K-itemsets candidatos contenidos en

esa transacción. Por lo tanto, si un ítem de una transacción no satisface tal condición, este puede eliminarse (recortarse) de esa transacción para la siguiente pasada.

La heurística descrita anteriormente se resume en las propiedades siguientes:

Recorte de transacción (*transaction trimming*): Un ítem de una transacción t puede ser eliminado de t si:

- ✓ Este no aparece en al menos K de los K-itemsets candidatos contenidos en esa transacción.
- ✓ Este no aparece en al menos un $K+1$ -itemset elegible de esa transacción, considerando como elegible aquel con todos sus $K+1$ -itemsets de tamaño K contenidos en el conjunto de candidatos de K-itemsets.

Poda de transacción (*transaction pruning*): Si una transacción t puede contener a un $K+1$ -itemset, entonces esta debe contener a $(K+1)$-itemset de tamaño K. En caso contrario, esta puede ser eliminada.

Los autores de este trabajo plantean que el algoritmo DHP se ve afectado positivamente con el incremento de los tamaños de las tablas de *hash*, aunque esto se logra a costo de la memoria (Park J.S., 1997).

En el 2002, se publica un algoritmo denominado **IHP** (*Inverted Hashing and Pruning*, por sus siglas en inglés). El mismo, es semejante a DHP ya que ambos utilizan tablas de *hash* para podar algunos de los itemsets candidatos. En DHP, durante la K ésima pasada sobre la base de datos, todo $K+1$ itemset contenido en cada transacción es acumulado en una tabla de *hash* permitiendo decidir, en caso que el valor de ese $K+1$ itemset sea menor que el soporte mínimo, que dicho itemset no sea considerado como candidato en la siguiente pasada.

En el algoritmo IHP, en lugar de acumular los itemsets en tablas de *hash* lo que se acumulan son los identificadores de las transacciones (TID), denominándose por ello estas tablas como *TID Hash Tables* (THT). Para lograr un efecto similar; o sea, para decidir si se puede alcanzar el soporte mínimo para un itemset se cuenta con una THT por cada ítem, verificándose que en todas las THT de ítem se utilice

la misma función de *hash* de TID. Debido a las colisiones, una celda de THT acumulará varios TID, asociándose las celdas con una partición del conjunto de transacciones. En la primera exploración de la base de datos se acumula el TID de cada transacción en cada THT de los ítems contenidos en ella, eliminándose al final de la primera pasada aquellos ítems que no resultaron frecuentes. Para saber si un ítem es frecuente basta con sumar el valor acumulado en cada celda de la THT de ese ítem y comprobar que no sea menor que el soporte mínimo establecido (Holt J.D., 2002).

1.9 Conclusiones parciales

Se analizaron los principales conceptos asociados a la minería de datos como etapa de mayor volumen e importancia dentro del proceso de extracción de conocimiento, centrando la atención en el estudio de las reglas de asociación como técnica descriptiva dentro del proceso anteriormente mencionado. Se definieron algunas métricas utilizadas para evaluar la relevancia de las reglas en cuestión, concluyendo que las medidas más utilizadas son el soporte y la confianza, aunque los resultados obtenidos con el uso de estas son subjetivos, pues están condicionados por el intelecto humano. Se abordaron además las temáticas referentes a la complejidad computacional de la extracción de itemsets candidatos, concluyendo que la mayoría de los algoritmos dedicados a esta tarea, siguen la idea del Apriori y que su mayor complejidad está en la determinación de los conjuntos de ítems frecuentes. Se determinó que el uso de las ontologías de dominio, es el mejor recurso para la utilización del conocimiento previo, en función de disminuir la complejidad computacional de la generación de itemsets candidatos.

2. CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL ALGORITMO

2.1 Introducción

Las declaraciones de mercancías de las aduanas son una manifestación para indicar el régimen aduanero que se ha de aplicar a las mercancías y los correspondientes datos que la entidad exige para la aplicación de este régimen, sirviendo de base para determinar el importe a pagar como resultado de la aplicación de la tarifa arancelaria correspondiente a las mercancías.

Una muestra como la que se presenta a continuación (Tabla 1) de algunas declaraciones de mercancías almacenadas en bases de datos de las aduanas, permitirá explicar el enfoque de la investigación y el porqué de la solución que en ella se propone.

La muestra almacena datos referentes al “origen” o país de procedencia donde se produce más del 50% de la mercancía; la “mercancía” que tendrá la denominación comercial, suficientemente explícita para permitir su identificación en la nomenclatura del Sistema Armonizado de Clasificación de Productos (SACLAP), aprobada en el Arancel de Aduanas, sin embargo para una mejor comprensión del ejemplo se emplea una descripción específica para cada una de ellas, en la práctica se almacena un código de 7 cifras; “tipo de moneda” con que se efectúa el pago; el “importe” que se encarga de almacenar para cada tipo de impuesto o tasa, el monto o cantidad a pagar como resultado de la aplicación de la tarifa correspondiente a las mercancías descritas en esa sub-partida arancelaria; la “cantidad de bultos” correspondiente a los artículos declarados; el “flete” donde se almacena el importe total en dólar estadounidense; el “peso” bruto total en Kg; el “tipo de declaración de mercancías” (DM Completa, DM Provisional, DM Anticipada, DM Incompleta); la “nacionalidad” del medio de transporte que traslada la carga importada o exportada, el “declarante” el código del Agente de Aduana o Apoderado, debidamente registrado ante la Aduana y por último si fue o no “mal clasificada”.

origen	mercancía	t_mon	importe	c_bultos	flete	peso	tipo_dm	Nac	decl	m_clas
Colombia	carne porcina congelada deshuesada	USD	20000	1	612	2000	2	Brasil	1	true
Canadá	carne pollo congelado en trozos	USD	1000	2	520	500	1	Canadá	5	false
Colombia	carne porcina congelada deshuesada	USD	55000	5	460	8000	2	EEUU	2	true
Brasil	carne porcina fresca en trozos	USD	4500	3	702	1000	3	Brasil	2	false
Argentina	carne pollo fresca sin trocear	USD	8000	6	821	6000	1	Uruguay	4	true
Colombia	carne porcina congelada deshuesada	USD	2000	8	963	25000	2	Venezuela	1	true
Brasil	carne equina fresca en trozos	USD	4500	3	426	1000	3	Chile	2	false
México	carne pollo congelado sin trocear	USD	150000	10	148	55500	1	EEUU	2	true
Argentina	carne pollo fresca en trozos	USD	8000	6	821	6000	1	Uruguay	4	true

Tabla 1. Muestra de datos asociados a las declaraciones de mercancías aduanales.

Los algoritmos que trabajan el descubrimiento de reglas de asociación, basan su funcionamiento en dos pasos (Agrawal R., 1994):

- ✓ La identificación de todos los conjuntos de ítems frecuentes con sus respectivos soportes, dado un soporte mínimo proporcionado por el usuario.
- ✓ La generación de las reglas de asociación que satisfagan las medidas de calidad indicadas como, por ejemplo, la confianza mínima proporcionada por el usuario.

Dado que el costo computacional del primer paso es mucho mayor que el del segundo, en esta investigación se aborda con mayor detenimiento el primero.

Generar todos los itemsets frecuentes es una tarea ardua y de alto costo. Varios han sido los algoritmos propuestos para ejecutar dicha tarea. De todos ellos, el método Apriori ha sido uno de los de mayor impacto, quizás el más referenciado de todos (Agrawal R., 1994).

De forma general, la mayoría de los algoritmos tipo Apriori primero construyen un conjunto de itemsets candidatos basados en alguna heurística y, posteriormente, determinan el subconjunto que realmente contiene los itemsets frecuentes. Este proceso puede realizarse de forma repetitiva conociendo que los itemsets frecuentes obtenidos en una iteración servirán de base para la generación del conjunto candidato en la siguiente iteración.

En particular, en el método Apriori, y variaciones de este como el AprioriTID y el AprioriHybrid, en la K-ésima iteración se generan todos los itemsets frecuentes de tamaño K. En la siguiente iteración, para construir el conjunto de los $(K+1)$-itemsets candidatos, se expanden determinados K-itemsets frecuentes en un $(K+1)$-itemset, considerando ciertas reglas y condiciones conocidas por "heurística Apriori". Este proceso se repite hasta un cierto K, o hasta que no se puedan generar más itemsets frecuentes.

Considerando lo previamente expuesto como "heurística Apriori", sería necesario transformar los artículos de la muestra en la combinación sucesiva de cada uno de sus valores, dando lugar a nuevos artículos, quedando de la siguiente forma:

nac	Brasil	Canadá	EEUU	Uruguay	Venezuela	Chile

Tabla 2. Transformación del atributo nac.

En la muestra en cuestión (Tabla 1) teniendo en cuenta que los 11 atributos tienen 5 o más valores distintos se generarían más de 55 nuevos atributos y contando con la presencia de 9 transacciones, sería necesario realizar más de 500 escaneos sobre los datos analizando cuáles son los itemsets cuyo soporte es mayor que el definido por el usuario y cada vez que cumpla con la condición se crearían nuevos itemsets con una mayor longitud, de 2 ítems, de 3 ítems, de 4 ítems y así sucesivamente hasta que no es posible crear más, generando en cada iteración nuevos escaneos sobre los datos provocando aumentos en la complejidad computacional. Por tanto mientras más grande sea la instancia de entrada a los algoritmos de identificación de conjuntos de ítems frecuentes, mayor será el número de escaneos que se realicen sobre los datos y aumentará exponencialmente el costo computacional de dichos algoritmos.

En la práctica las bases de datos de las declaraciones de mercancías de las aduanas contienen millones de transacciones y cientos de clasificaciones de mercancías, resultando prácticamente imposible para el cerebro humano, identificar los conjuntos de ítems frecuentes de forma manual. Por otro lado, si se aplican algoritmos de minado de itemsets frecuentes, resulta exponencial el número de escaneos sobre los datos para lograr generar todos los itemsets frecuentes y con ellos el conjunto de ítems candidatos, resultando prácticamente insostenible su total ejecución en un tiempo aceptable.

Considerando que un tiempo de ejecución aceptable, en el caso de respuestas de operaciones sobre una base de datos, se encuentra en el orden de los milisegundos o segundos, mientras que para el tipo de procesamiento que aborda la investigación, se tienen órdenes de minutos cercanos a 10. Además, debido a que la mayoría de estos procedimientos presentan una complejidad computacional $O\ m * n^2$, donde n y m representan el número de atributos y transacciones respectivamente, sería potencialmente inmensurable su costo final.

Por tanto, es necesario desarrollar un mecanismo que permita especificar una serie de restricciones sobre los datos con la que los usuarios puedan definir uno o varios conjuntos de atributos y/o transacciones de poca significancia y que no aporten utilidad relevante en el proceso de generación de reglas de asociación, para que puedan ser excluidos del proceso de identificación de itemsets frecuentes. En la muestra del caso de estudio (Tabla 1) pudieran especificarse restricciones de eliminación sobre los atributos **t_mon** y **flete**.

t_mon	flete
USD	612
USD	520
USD	460
USD	702
USD	821
USD	963
USD	426
USD	148
USD	821

Tabla 3. Atributos y valores irrelevantes asociados a las declaraciones de mercancías.

Eliminando dichos atributos se reduce la cantidad de elementos contenidos en la muestra de las declaraciones de mercancías de 11 a 9 campos (ver Tabla 3), disminuyendo las combinaciones de valores distintos de cada atributo de 55 a 45, y por tanto, descienden de 500 a 400 los escaneos sobre la base de datos.

origen	mercancía	importe	c_bultos	peso	tipo_dm	nac	decl	m_clas
Colombia	carne porcina congelada deshuesada	20000	1	2000	2	Brasil	1	true
Canadá	carne pollo congelado en trozos	1000	2	500	1	Canadá	5	false
Colombia	carne porcina congelada deshuesada	55000	5	8000	2	EEUU	2	true
Brasil	carne porcina fresca en trozos	4500	3	1000	3	Brasil	2	false
Argentina	carne pollo fresca sin trocear	8000	6	6000	1	Uruguay	4	true
Colombia	carne porcina congelada deshuesada	2000	8	25000	2	Venezu	1	true
Brasil	carne equina fresca en trozos	4500	3	1000	3	Chile	2	false
México	carne pollo congelado sin trocear	150000	10	55500	1	EEUU	2	true
Argentina	carne pollo fresca en trozos	8000	6	6000	1	Uruguay	4	true

Tabla 4. Muestra de datos asociados a las declaraciones de mercancías sin los atributos t_mon y flete.

Sin embargo, si en vez de trabajar con la muestra se aplicara el procedimiento en condiciones reales como las mencionadas anteriormente y se eliminaran por ejemplo, 50 o más campos y al menos 100 transacciones de los millones existentes en las declaraciones de mercancías de las aduanas, los resultados serían realmente considerables.

2.2 Propuesta de solución

Se ha diseñado un mecanismo que permite definir una serie de limitaciones o restricciones de eliminación relacionadas con los conceptos y propiedades del dominio, permitiendo eliminar un grupo de ítems y/o un conjunto de transacciones de la base de datos, de acuerdo a los intereses definidos por el usuario, de manera que se reduzca el tamaño de la instancia de entrada a los algoritmos de minado de itemsets frecuentes y por consecuente su complejidad computacional.

El mismo ha sido desarrollado a partir del método *Pruning Constraints* (Bellandi, 2007), en cuyas restricciones se excluye a un solo elemento o todos los elementos que pertenecen a un concepto de ontología del dominio, que además pueden incluir una condición expresada en un conjunto de propiedades de dichos conceptos. La utilización de la ontología permitirá el empleo del conocimiento previo expresado en ella para obtener todos los elementos involucrados en las restricciones y finalmente poder excluirlos de la base de transacciones de las declaraciones de mercancías de las aduanas.

2.3 Algoritmo

El algoritmo propuesto tiene como entrada tres parámetros proporcionados por el usuario, los mismos deben cumplir con un formato determinado para su correcta ejecución.

El primer parámetro representa una ontología de dominio (O) que no está pensada como un repositorio de datos de entrada, sino que se utiliza para modelar el dominio de interés representando una taxonomía enriquecida con las propiedades de los datos. Su empleo permite centrar el análisis en un único fragmento de los datos, sin tener en cuenta todos los elementos específicos.

Este parámetro debe ser expresado en un formato $OWL\backslash XML$ por su capacidad de representar clases mediante restricciones sobre propiedades, valores o cardinalidad. La nomenclatura a emplear en la definición de los conceptos y sus propiedades, debe coincidir con la utilizada en la base de datos donde se encuentran almacenados los mismos.

La taxonomía de conceptos de la ontología correspondiente al caso estudio (Tabla 1) con sus propiedades y algunos valores correspondientes, se presentan a continuación:

Estructura jerárquica	Propiedades de mercancia
OWL: Things ⬇ declaracion_mercancia ⬇ mercancia	origen: Colombia importe: 20000 c_bultos: 1 peso: 2000 tipo_dm: 2

Figura 5. Jerarquía y propiedades ontológicas definidas a partir de las declaraciones de mercancías.

El segundo parámetro define una serie de limitaciones o restricciones de eliminación (R) relacionadas con los conceptos y propiedades del dominio de interés expresado en la ontología.

Dichas restricciones deben cumplir con la siguiente formalidad:

Sea $I = \{i_1, i_2, \dots, i_n\}$: Un conjunto de ítems.

Sea $C = \{c_1, c_2, \dots, c_n\}$: Conjunto de conceptos de la ontología.

Sea $Pc = \{p_1, p_2, \dots, p_n\}$: Conjunto de propiedades de los conceptos $c \in C$.

$cond_c$: Combinación de condiciones usando los operadores $<, >, \leq, \geq, !=, =$, y v un valor constante del dominio de Pc.

Una restricción de eliminación tendría dos formas distintas de escribirse, empleando como factor común las palabras reservadas $eliminar$ y $donde$. Además, durante la especificación de las mismas debe seguirse el siguiente orden lógico:

Se escribirán primero todas las restricciones que permitan el particionamiento horizontal de la base de transacciones. Estas comenzarían por la instrucción $eliminar$, seguida por los argumentos entre paréntesis i o c, donde $i \in I$ y $c \in C$, luego de una coma, la palabra reservada $donde$ seguida de una condición de la forma $i\ cond_c\ v$ donde $c \in C \cup I, i \in I$.

Una vez que se han terminado de especificar todas las restricciones horizontales, se escribirán aquellas que permitan particionar la base de transacciones de forma vertical. Las mismas comenzarían a declararse por la instrucción $eliminar$ seguida por los argumentos entre paréntesis i o c, donde $i \in I$ y $c \in C$.

Respetar el orden lógico previamente establecido, evitará que la solución se vea obligada a lanzar excepciones durante su ejecución. Es necesario evitar que se invierta el orden de especificación de las restricciones de eliminación; de no ser así, se realizarían particionamientos verticales que eliminarían un grupo de ítems (columnas) que pudieran estar involucrados en las restricciones de particionamiento horizontal.

Formalmente una restricción es definida en I, C, Pc de la siguiente forma:

$eliminar\ c\ , donde\ i\ cond_c\ v;\ c \in C \cup I, i \in I$

$eliminar\ i\ , donde\ i\ cond_c\ v;\ i \in I$

$eliminar\ \ i\ o\ c\ ;\ i\ \in I, c\ \in C$

Algunas restricciones que pudieran escribirse sobre los datos del caso estudio (Tabla 1) serían:

$eliminar\ \ flete\ , donde\ flete\ >\ 600$

$eliminar\ \ t\_mon$

El objetivo de estas limitaciones es excluir un conjunto de artículos de las transacciones utilizadas como entrada a los algoritmos de identificación de itemsets frecuentes.

El tercer parámetro es un conjunto de transacciones correspondientes al dominio descrito en la ontología, las mismas se encontrarán almacenadas en una base de datos (D) que será procesada y coleccionada en un archivo de texto plano. Los datos recopilados en dicho archivo deben cumplir con el formato $tabla - atributo: valor, tabla - atributo: valor,$ ...; donde $tabla$ responde al nombre que llevan las tablas en la base de datos; $atributo$ representa el nombre de los campos contenidos en las tablas y $valor$ almacena los valores correspondientes a cada campo. Nótese además que no existen espacios entre las nomenclaturas ni los signos de puntuación empleados. A continuación se muestra un ejemplo de transformación de los datos de la muestra (Tabla 1) al formato anteriormente descrito:

$declaracion\_mercancia - origen: Colombia, declaracion\_mercancia - origen: Canada,$ ...

La primera línea del archivo representa el conjunto de todos los posibles valores que pueden tomar los ítems $tabla - atributo: valor$, y las líneas restantes responden a las transacciones correspondientes al dominio, quedando como se muestra a continuación:

```
declaracion_mercancia-codarticulo:016,declaracion_mercancia-nodm:00039
declaracion_mercancia-codarticulo:001,declaracion_mercancia-nodm:00050
declaracion_mercancia-codarticulo:002,declaracion_mercancia-nodm:00050
declaracion_mercancia-codarticulo:003,declaracion_mercancia-nodm:00050
declaracion_mercancia-codarticulo:001,declaracion_mercancia-nodm:00049
declaracion_mercancia-codarticulo:002,declaracion_mercancia-nodm:00049
declaracion_mercancia-codarticulo:001,declaracion_mercancia-nodm:00047
declaracion_mercancia-codarticulo:001,declaracion_mercancia-nodm:00054
```

Figura 6. Base de transacciones de las declaraciones de mercancías.

Una vez que la ontología y las limitaciones han sido definidas, se puede comenzar con la interpretación de las restricciones. Este paso se logra mediante la consulta de la ontología de dominio, con el fin de recuperar todos los elementos implicados en el proceso, verificando que cada elemento cumple con las restricciones. El resultado de este proceso es un conjunto de pares de $X = Y, Z$ donde Y representa un conjunto de ítems y Z los operadores empleados en las restricciones de eliminación R.

En el proceso de transformación se utiliza el conjunto de pares $X = Y, Z$ para analizar los datos almacenados en la base de datos D y transformarla mediante la exclusión de cada elemento según las restricciones de eliminación. De este proceso se deriva una base de transacciones transformada D^*, que lógicamente contiene un número menor de campos y transacciones, permitiendo a los algoritmos de minería de itemsets frecuentes con la generación de candidatos trabajar con una instancia de menor tamaño, lo que contribuye a disminuir su costo computacional.

La Figura 7 resume el proceso anteriormente descrito.

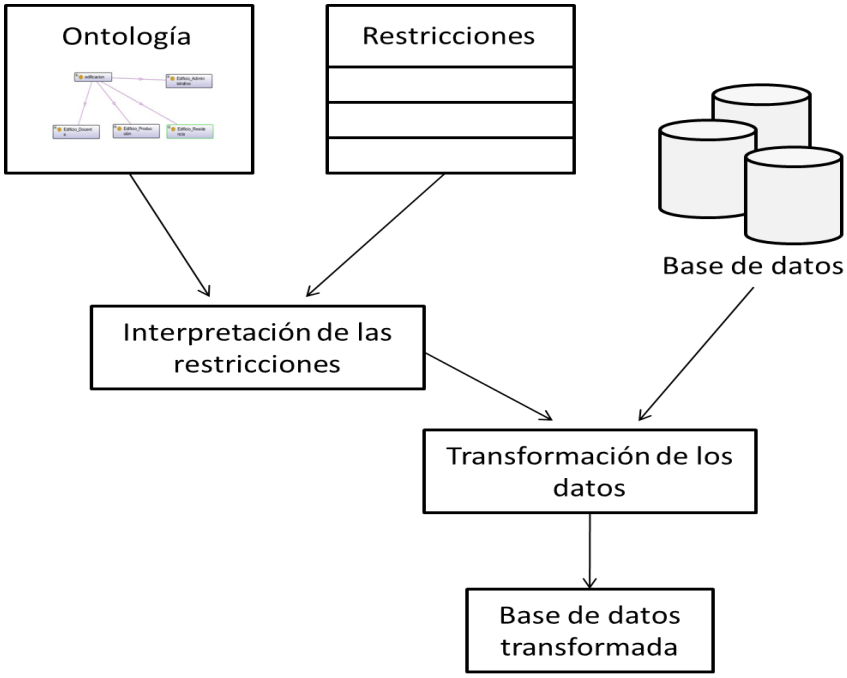

Figura 7. Restricciones de poda o eliminación.

En el algoritmo propuesto se utilizan las siguientes notaciones:

O: Una ontología de dominio específico $O = C \cup Pc$; donde C es un conjunto de conceptos del dominio $C = \{c_1, c_2, \dots, c_n\}$ y Pc es un conjunto de propiedades de los conceptos $Pc = \{p_1, p_2, \dots, p_n\}$

R: Conjunto de restricciones del dominio $R = \{r_1, r_2, \dots, r_n\}$.

D: Conjunto de ítems y transacciones del dominio específico almacenadas en una base de datos $D = \{t_1, t_2, \dots, t_n\}$.

Algoritmo de eliminación

Entradas: O, R, D

Paso 1: Obtener de O todos los itemsets que cumplan con las restricciones expresadas en R.

Paso 2: Eliminar en D todos los itemsets recuperados en **Paso 1**.

Salida: D^*

2.3.1 Complejidad temporal del algoritmo

Una vez que se dispone del algoritmo propuesto, es necesario definir criterios para medir su comportamiento (rendimiento), esto está centrado fundamentalmente en la utilización eficiente de los recursos. El proceso de análisis del mismo se basará en la estimación del tiempo de ejecución.

La complejidad temporal (el tiempo) depende de diversos factores, como pueden ser: los datos de entrada suministrados, el lenguaje utilizado, la calidad del código generado por el compilador o la eficiencia del intérprete, la naturaleza y rapidez de las instrucciones de máquina del procesador que ejecute el programa, y la complejidad intrínseca del algoritmo.

Un análisis detallado que tenga en cuenta todos estos factores es difícil de realizar, costando tiempo y esfuerzo, es por ello que el análisis se simplifica de modo tal que se obtengan resultados igualmente razonables. Una de las aproximaciones fundamentales se basa en estimar el tiempo de ejecución que consume el algoritmo para realizar su trabajo. Para ello se emplea una medida teórica (apriori), que consiste en obtener una función que acote (superior o inferiormente) el tiempo de ejecución del algoritmo para cualquier instancia del problema.

Para calcular la complejidad del algoritmo se utilizaron las siguientes reglas:
Regla 1: El tiempo requerido para el acceso a un valor es un valor constante, así como el tiempo para realizar operaciones aritméticas elementales, como la adición, substracción, multiplicación, división y para las comparaciones es constante. El tiempo para almacenar el resultado en memoria también es constante, con un valor de $O(1)$.

Regla 2: El tiempo requerido para la ejecución de un algoritmo de secuencia lineal es la sumatoria de los tiempos de cada instrucción.

Regla 3: El tiempo requerido para la ejecución de una instrucción condicional $if\ C\ \ then\ I_1\ else\ I_2$, es la suma del tiempo necesario para evaluar la expresión más el tiempo máximo de ejecución de los bloques asociados a las ramas *then* y *else*.

$$T\ =\ T(C)\ +\ Máximo(\ T(I_1), T(I_2)\ )$$

Regla 4: El tiempo requerido para la ejecución de una instrucción repetitiva $while\ (C)\ do\ I$, es la suma del tiempo necesario para la evaluación de la expresión C + el número de iteraciones multiplicado por el tiempo para la instrucción sumado con el tiempo necesario para la evaluación de la expresión por la cantidad de iteraciones.

$$T = T(C) + iteraciones * T(C) + iteraciones * T(I)$$

Agrupando términos semejantes obtenemos:

$$T = T(C) + (iteraciones) * (T(C) + T(I))$$

Regla 5: Bloque de sentencias. Se aplica la regla de la suma, de forma que se calcula el tiempo de ejecución tomando el máximo de los tiempos de ejecución de cada una de las partes.

Regla 6: Llamadas a funciones. Si una determinada función P tiene un tiempo de $O(f(n))$, con n a la medida del tamaño de los argumentos, cualquier función que llame a P tiene en la llamada una cota de $O(f(n))$. Las asignaciones con diversas llamadas a funciones deben de sumar las cotas del tiempo de ejecución de cada llamada.

Aplicando estas reglas se determinó que la complejidad temporal del algoritmo propuesto es: $O(n^5)$

2.4 Implementación

La implementación del algoritmo de eliminación de itemsets está basada en dos artefactos o clases elementales:

- ✓ Una clase (***ProcesarDatos***) encargada de procesar y transformar las transacciones de la base de datos al formato descrito en el **epígrafe 2.3**. Además, permite cargar los archivos de extensión $*.owl$ (que contiene la ontología de dominio específico) y $*.txt$ (que contiene las restricciones de eliminación).

- ✓ Una clase (***RestriccionPoda***) que se encarga de ejecutar el proceso de interpretación de las restricciones de eliminación para posteriormente eliminar los itemsets y/o transacciones de la base de datos.

La primera clase (***ProcesarDatos***) es la encargada de interactuar con un archivo de extensión $*.sql$ resultante de realizar la siguiente consulta a la base de datos de las declaraciones de mercancías:

$$copy\ select\ 'tabla-atributo:', tabla.atributo\ from\ tabla\ to\ ruta\_fichero$$

Como resultado de esta consulta se obtiene una copia de los datos en el archivo anteriormente mencionado, que posee un formato de texto plano como se muestra a continuación:

```
COPY declaracion_mercancia(id_declaracion,origen,mercancia,t_mon,importe,c_bultos,flete,peso,tipo_dm,nac,decl,m_clas
1   Colombia    carne porcina congelada deshuesada  USD 20000   1   612 2000    2   Brasil  1   true
2   Canada  carne pollo congelada en trozos USD 1000    2   520 500 1   Canada  5   false
3   Brasil  carne porcina congelada deshuesada  USD 55000   5   460 8000    2   EEUU    2   true
4   Argentina   carne pollo fresca sin trocear  USD 8000    6   821 6000    1   Uruguay 4   true
5   Mexico  carne pollo congelada sin trocear   USD 150000  10  148 55500   1   EEUU    2   true
\.
```

Figura 8. Copia de las transacciones de las declaraciones de mercancías de la aduana.

ProcesarDatos implementada la función ***TransformarDatos***, donde se realizan operaciones mayormente cíclicas, por la necesidad de recorrer todas las líneas del archivo de extensión $*.sql$ y por consiguiente cada elemento contenido en ellas. Paralelamente a este recorrido, se construye el archivo con extensión $*.txt$ de salida del proceso de transformación, de forma tal que el archivo original es leído línea a línea, las cuales son modificadas de acuerdo al formato definido en el **epígrafe 2.3.** Como resultado de este procedimiento se obtiene un nuevo archivo de extensión $*.txt$ que contiene las líneas modificadas. Es válido aclarar que no es necesario ejecutar esta función más de una vez, solo en el caso de que se hayan realizado cambios en la base de datos.

La Figura 9 representa el proceso preliminarmente descrito.

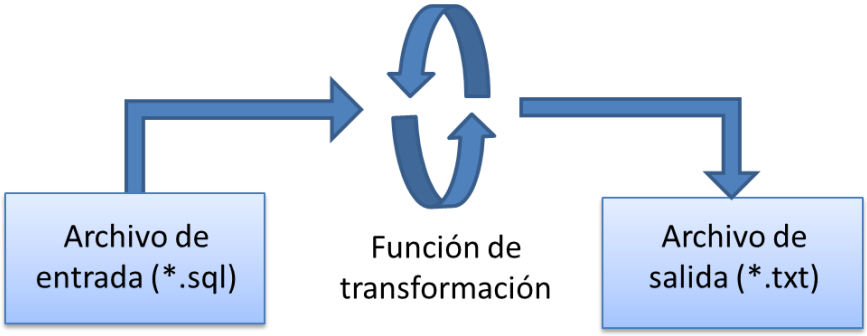

Figura 9. Procedimiento para transformar los datos en el formato definido en epígrafe 2.3.

La segunda clase (***RestriccionPoda***) implementa tres procedimientos esenciales: ***InterpretarRestriccion***, ***EliminarItemsets*** y ***EliminarTransaccion.*** El primero se encarga de interactuar con los archivo de extensión $*.owl$ y $*.txt$ (que contienen la ontología de dominio específico y las restricciones de eliminación respectivamente). Los dos últimos, trabajan directamente con el archivo de extensión $*.txt$ que se obtiene como resultado de ejecutar la función ***TransformarDatos*** de la clase (***ProcesarDatos***).

La función ***InterpretarRestriccion*** realiza operaciones cíclicas, debido a que es necesario recorrer línea a línea los archivos de extensión $*.owl$ y $*.txt$, verificando en cada recorrido que coincida la información brindada en las restricciones de eliminación, con la expresada en la ontología de dominio específico, permitiendo recuperar todos los elementos involucrados en el proceso de eliminación de itemsets y/o transacciones. Estos últimos servirán de entrada a las funciones ***EliminarItemsets*** y ***EliminarTransaccion***, encargadas de ejecutar las particiones verticales y horizontales respectivamente sobre la base de transacciones. ***InterpretarRestriccion*** es el centro del algoritmo, razón por la cual, se ocupa de establecer la coreografía de ejecución del mismo, de acuerdo a lo especificado en el **epígrafe 2.3**.

En la Figura 10 se resume el procedimiento de interpretación.

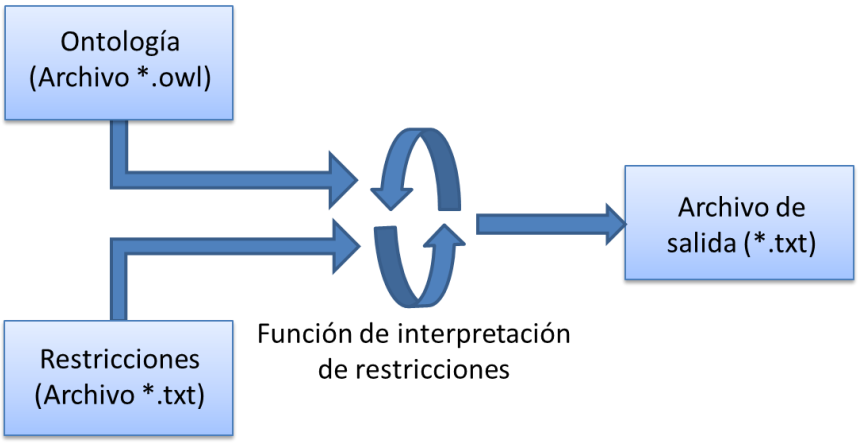

Figura 10. Procedimiento para interpretar las restricciones.

La función ***EliminarItemsets*** realiza operaciones cíclicas por la necesidad de recorrer línea a línea el archivo con extensión $*.txt$ que contiene las transacciones, con el objetivo de eliminar en cada caso la ocurrencia de los itemsets provenientes de ejecutar la función ***InterpretarRestriccion***. Como resultado de este procedimiento se actualiza el archivo anteriormente mencionado con las líneas modificadas, conteniendo una menor cantidad de itemsets a procesar por los algoritmos de identificación de itemsets frecuentes.

La siguiente figura resume el proceso enunciado.

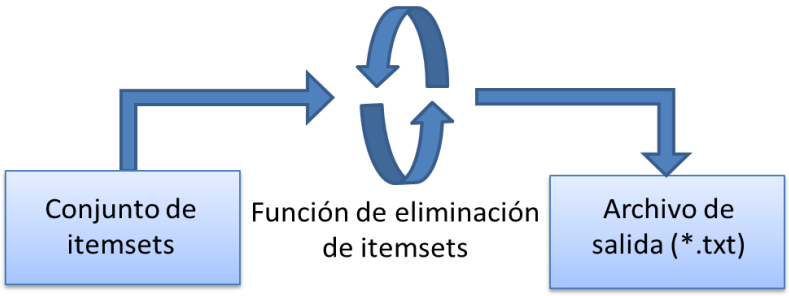

Figura 11. Procedimiento para eliminar itemsets.

La función ***EliminarTransaccion*** también realiza operaciones mayormente cíclicas debido a que es necesario recorrer línea a línea el archivo de extensión $*.txt$ que contiene las transacciones de la base de datos, con el propósito de verificar si existen elementos en las transacciones que cumplan

con las condiciones que han sido expresadas sobre los itemsets. Estas condiciones han sido recuperadas al ejecutar la función ***InterpretarRestriccion***. Una vez que termina el proceso de verificación, se excluyen de la base de datos aquellas transacciones que cumplieron con las condiciones**.** Como resultado de este procedimiento es modificado el archivo en cuestión, que lógicamente presenta una menor cantidad de transacciones a procesar por los algoritmos de identificación de itemsets frecuentes con la generación de candidatos.

La Figura 12 representan el procedimiento previamente descrito.

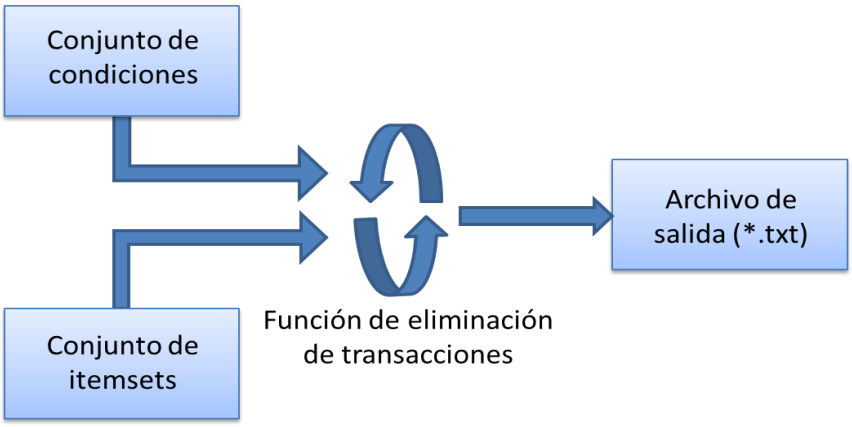

Figura 12. Procedimiento para eliminar transacciones.

Para el correcto funcionamiento de los procedimientos previamente descritos, fue necesario implementar varias funcionalidades auxiliares. La descripción y codificación de cada una de ellas se muestran a continuación.

La función **ObtenerValor** (Figura 13) realiza un recorrido cíclico por todas las transacciones contenidas en el archivo de extensión $*.txt$ donde se encuentra almacenada la base de datos transaccional, recuperando en cada iteración el valor correspondiente al ítems involucrado en el proceso de eliminación.

```
public String ObtenerValor(String item, String transaction) {
    String aux = "";
    if (transaction.contains(item)) {
        int pos = transaction.indexOf(item);
        for (int i = pos; i < transaction.length(); i++) {
            if (transaction.charAt(i) != ',') {
                aux += transaction.charAt(i);
            } else {
                break;
            }
        }
    }
    return aux.split(":")[1];
}
```

Figura 13. Función para obtener el valor del ítem involucrado en las restricciones.

La función **LeerRestriccion** (Figura 14) permite recuperar todos los elementos, tanto conceptos y propiedades, como las condiciones contenidas en el archivo con extensión $*.txt$ donde se encuentran las restricciones de eliminación que se establecieron sobre el dominio.

```
public String[] LeerRestriccion(String linea) {
    String elementos[] = null;
    if (linea.contains(",donde")) {
        elementos = linea.split(",donde");
        elementos[0] = elementos[0].substring(elementos[0].indexOf("(") + 1,
                                              elementos[0].indexOf(")"));
    } else {
        elementos = new String[1];
        elementos[0] = linea.substring(linea.indexOf("(") + 1, linea.indexOf(")"));
    }
    return elementos;
}
```

Figura 14. Función para obtener los elementos de las restricciones.

La función **ExistePropiedad** (Figura 15) se encarga de verificar cíclicamente en cada iteración si existen en el archivo de extensión $*.owl$ que contiene la ontología de dominio específico, las propiedades recuperadas en el proceso de interpretación de las restricciones de eliminación.

```
public boolean ExistePropiedad(String propiedad) {
    NodeList nodos_propiedades = archivo.CargarOntologia().
                                    getElementsByTagName("DataProperty");
    for (int i = 0; i < nodos_propiedades.getLength(); i++) {
        Node nodo = (Node) nodos_propiedades.item(i);
        if (nodo.getAttributes().getNamedItem("IRI").
            getNodeValue().equals("#" + propiedad)) {
            return true;
        }
    }
    return false;
}
```

Figura 15. Función que permite verificar la existencia de propiedades en la ontología.

La función **ExisteConcepto** (Figura 16) trabaja cíclicamente verificando en cada iteración si existen en el archivo de extensión $*.owl$ que contiene la ontología de dominio específico, los conceptos recuperados en el proceso de interpretación de las restricciones de eliminación.

```
public boolean ExisteConcepto(String concepto) {
    NodeList nodos_propiedades = archivo.CargarOntologia().
                                    getElementsByTagName("Class");
    for (int i = 0; i < nodos_propiedades.getLength(); i++) {
        Node nodo = (Node) nodos_propiedades.item(i);
        if (nodo.getAttributes().getNamedItem("IRI").getNodeValue().
            equals("#" + concepto)) {
            return true;
        }
    }
    return false;
}
```

Figura 16. Función que permite verificar la existencia de conceptos en la ontología.

La función **BuscarPropiedades** (Figura 17) realiza un recorrido cíclico recuperando en cada iteración sobre el archivo de extensión $*.owl$ que contiene la ontología de dominio específico, las propiedades pertenecientes a un concepto resultante del proceso de interpretación de las restricciones de eliminación.

```
public ArrayList<String> BuscarPropiedades(String concepto) {
    ArrayList<String> propiedades = new ArrayList<>();
    NodeList nodos_propiedades = archivo.CargarOntologia().
                                 getElementsByTagName("DataPropertyDomain");
    for (int i = 0; i < nodos_propiedades.getLength(); i++) {
        Node nodo = (Node) nodos_propiedades.item(i);
        if (nodo.getChildNodes().item(3).getAttributes().
            getNamedItem("IRI").getNodeValue().equals("#" + concepto)) {
            propiedades.add(nodo.getChildNodes().item(1).
            getAttributes().getNamedItem("IRI").getNodeValue());
        }
    }
    return propiedades;
}
```

Figura 17. Función que permite obtener propiedades de un concepto en la ontología.

La función **BuscarSubConcepto** (Figura 18) recupera cíclicamente en cada iteración sobre el archivo de extensión $*.owl$ que contiene la ontología de dominio específico, los subconceptos o subclases pertenecientes a un concepto o clase padre resultante del proceso de interpretación de las restricciones de eliminación.

```
public ArrayList<String> BuscarSubConcepto(String concepto) {
    ArrayList<String> sub_conceptos = new ArrayList<>();
    NodeList nodos_sub_conceptos = archivo.CargarOntologia().
                                   getElementsByTagName("SubClassOf");
    for (int i = 0; i < nodos_sub_conceptos.getLength(); i++) {
        Node nodo = (Node) nodos_sub_conceptos.item(i);
        if (nodo.getChildNodes().item(3).getAttributes().
            getNamedItem("IRI").getNodeValue().equals("#" + concepto)) {
            sub_conceptos.add(nodo.getChildNodes().item(1).
            getAttributes().getNamedItem("IRI").getNodeValue());
        }
    }
    return sub_conceptos;
}
```

Figura 18. Función que permite obtener subconceptos de un concepto en la ontología.

2.5 Conclusiones parciales

En la investigación se ha desarrollado un marco integrado para la eliminación de itemsets no significativos para el usuario en la etapa de pre-procesamiento del minado de reglas de asociación basado en restricciones de eliminación con la ayuda del conocimiento previo expresado en una ontología de dominio específico. El marco permite la definición de limitaciones o restricciones de eliminación sobre los conceptos y/o propiedades de la ontología permitiendo pre-procesar las transacciones de la base de datos utilizadas como entrada a los algoritmos de identificación de itemsets frecuentes con la generación de candidatos. El mismo, cuenta con una arquitectura en la que se desacopla el motor de minería de datos con el dominio de especificación. Las principales ventajas de la técnica propuesta se puede resumir en términos de extensibilidad y flexibilidad: uno puede cambiar los detalles algorítmicos o extender la ontología, sin cambiar el nivel de aplicación ni las transacciones de la base de datos.

3. CAPÍTULO 3: VALIDACIÓN DEL ALGORITMO

3.1 Introducción

Las pruebas experimentales o de validación son una actividad en la cual un sistema, un componente o un algoritmo (como el desarrollado en la investigación), son ejecutados bajo condiciones o requerimientos especificados, con el objetivo de observar y registrar los resultados. A través de ellas se realiza un análisis teniendo en cuenta los datos de entrada, resultados esperados y condiciones que deben cumplirse mientras se ejecutan los casos de prueba, identificando los errores que pueda tener el algoritmo, corregirlos y obtener un óptimo funcionamiento.

3.1.1 Evaluación y validación

Una vez realizada la investigación, es necesario evaluar y validar que las afirmaciones cometidas sobre la solución propuesta sean aceptables para la comunidad investigadora. Los siguientes métodos (Vaishnavi, et al., 2008) ofrecen medios para la evaluación y validación de la solución desarrollada:

Demostración

El objetivo fundamental de este método es demostrar que la solución a un problema es realizable y válida en situaciones predefinidas. Se emplea cuando se ha desarrollado una situación de un problema, cuya solución no es posible demostrar matemáticamente con exactitud. Sin embargo, se desea demostrar que la solución es realizable y trabaja para un conjunto de situaciones predefinidas.

El método es especialmente relevante cuando la demostración de una solución en sí misma se considera una contribución. Para desarrollarlo es necesario tener en cuenta los siguientes pasos:

1. Construir la solución. Esto puede significar la construcción de un prototipo de la solución. La construcción de la solución mostrará que la misma es realizable.
2. Demostrar que la solución construida es razonable para un conjunto de situaciones predefinidas. Estas situaciones deben ser predefinidas y no creadas para adaptarse a la solución. Deberán estar construidas para ejercer las variaciones del problema.

La demostración puede mostrar las deficiencias de la solución. Por otro lado, se puede mostrar que la solución es viable y aceptable. Las pruebas exhaustivas aumentará la confianza en la solución. Si las situaciones de prueba están diseñadas apropiadamente, entonces la construcción de la solución y sus pruebas para estas situaciones, puede demostrar su validez. A pesar de que teóricamente la Demostración constituye el método de validación menos formal, ha sido utilizado en innumerables artículos científicos de la especialidad hasta el punto de ser el tipo de validación que más se utiliza desde el año 1999 (Shaw, 2002).

Experimentación

Su objetivo es validar o rechazar un conjunto de hipótesis relacionadas con las afirmaciones acerca de la solución. Es empleado cuando se han desarrollado un conjunto de hipótesis relacionadas con las afirmaciones acerca de la solución (por lo general de un sistema) que no pueden ser probadas matemática o lógicamente. Es necesario para generar los datos del sistema y luego utilizar esta información para validar o rechazar las hipótesis.

La naturaleza del experimento y la validación de hipótesis dependen del tipo de experimento. Estos tipos, a su vez, dependen del método utilizado en el desarrollo de la solución. En general, un experimento debe cumplir los siguientes criterios que inciden en la confianza o la generalidad de los resultados establecidos por el experimento:

1. La validez de la construcción. Los sustitutos de las construcciones que no pueden ser fácilmente observados en el experimento, deben ser sustitutos válidos.
2. La validez interna. El experimento no debe involucrar a las construcciones que influyen en el comportamiento observado, con los que forman parte de las hipótesis.
3. La validez externa. Si se supone que los resultados del experimento no son generales, sino que se prueban en un entorno limitado, simulado, se debe argumentar que los resultados son generalizables.
4. Fiabilidad. El experimento debe ser reproducible.

Este método ayudará a establecer resultados asociados con la solución del problema de investigación en situaciones donde la recogida y análisis de datos es el único método factible de validación.

Simulación

Se encarga de evaluar y validar una solución al problema de la investigación. Es aplicable cuando el problema de investigación es complejo, tal que una solución, no puede ser demostrada matemáticamente como válida. Además cuando la evaluación y validación de la solución en el ámbito de la vida real, no es viable o muy costosa.

En la utilización de este método se tienen en cuenta los siguientes pasos:

1. Desarrollar el modelo conceptual del problema y su solución que será simulado en una computadora. Esto implica decidir qué entidades e interacciones deben ser capturadas en la simulación, cuyo propósito es evaluar el desempeño de la solución al problema y probar su validez.
2. Desarrollar un conjunto inicial de datos de prueba que pueden ejercer el modelo. Esto debe tener en cuenta los objetivos de la solución (artefacto) y el entorno exterior en el que la misma debe operar. Esto implicará la modelización del entorno exterior.
3. Seleccionar un paquete de simulación que está diseñado específicamente para el dominio del problema. Esto implicará la menor cantidad de programación. Si un paquete no está disponible, a continuación, se elige un lenguaje de programación general, como C + + o Java y el modelo del problema, solución, y las construcciones externas del medio ambiente.
4. Ejecutar el programa de simulación para el conjunto de pruebas desarrollado previamente. Recoger datos de rendimiento y analizarlo para evaluar la solución. Si el desempeño no cumple con las expectativas, entonces se puede volver a analizar y revisar la solución. En caso contrario, comprobar la solución sobre una amplia gama de condiciones. Probar la solución en condiciones extremas para ver la gama de condiciones ambientales exteriores sobre el cual la solución es válida.
5. Argumentar que el análisis de los datos apoya la validez de las hipótesis acerca de la solución.

La Simulación ofrece una forma razonable de coste efectivo de evaluación y validación de una solución. La alternativa de poner a prueba la solución en la configuración de la vida real puede ser a la vez costosa y consume mucho tiempo, o tal vez ni siquiera sea factible.

Uso de Métrica

Tiene como propósito usar medidas establecidas para ayudar a la validación de la propia solución al problema de investigación. Se emplean indicadores establecidos que existen en la literatura para evaluar el desempeño de la solución; para probar o argumentar la corrección de las hipótesis que se han hecho en relación con el rendimiento de la solución. En el caso de que las métricas no estén disponibles para medir el rendimiento de la solución, se pueden usar para un problema similar.

Este método sigue las siguientes formalidades:

1. Determinar si existen o no las métricas establecidas que son apropiadas para medir el rendimiento de la solución y compararlo con el rendimiento de soluciones anteriores (si es que existen). Si tales parámetros no existen, determinar si existen o no las métricas para medir el desempeño de problemas similares a nuestro propio problema. En tal caso, se debe argumentar que el uso de las métricas elegidas es una forma razonable de evaluación y validación.
2. Analizar y medir la solución usando la o las métricas elegidas. Esto puede implicar las pruebas matemáticas, las mediciones experimentales, o simulación.
3. Demostrar que la solución tiene la hipótesis de rendimiento de acuerdo a las métricas seleccionadas.

Evaluación comparativa

Emplea un punto de referencia disponible para demostrar que una solución tiene un rendimiento razonable o que es mejor que otra solución disponible. Se utiliza cuando no hay métricas establecidas disponibles que se puedan utilizar para medir el rendimiento de una solución, pero es necesario verificar que el rendimiento es razonable o mejor que otra solución disponible. La comunidad científica, sin embargo, desarrolló un punto de referencia para evaluar las soluciones a la misma clase de problemas. Si no hay ningún punto de referencia disponible, se puede crear un escenario de prueba o una clase de estos escenarios que se puedan utilizar para evaluar la solución, así como cualquier otra solución disponible.

En la aplicación de este método se siguen los siguientes pasos:

1. Identificar el punto de referencia que se puede utilizar para evaluar y validar la solución. Si no hay ningún punto de referencia disponible, se puede crear uno propio. En este caso, sin embargo, se necesita establecer que el punto de referencia tiene cierta validez independiente y no está parcializado hacia una solución.
2. Utilizar el punto de referencia para mostrar el mérito de la solución. Si no existe ninguna solución para el problema de investigación, entonces se debe demostrar que la solución cumple con los criterios especificados en el punto de referencia. Si existen soluciones para el problema, entonces se debe demostrar (con el punto de referencia) que la solución propuesta es una mejor solución al problema en comparación con las soluciones existentes.

La evaluación comparativa proporciona un vehículo para la evaluación objetiva de una solución o comparaciones de soluciones diferentes. Esto hace que sea fácil afirmar que realmente se ha proporcionado una solución a un problema o que se ha demostrado que una solución propia, es mejor que otras soluciones existentes.

Razonamiento lógico

Su objetivo es sostener la validez de la solución. Es aplicable cuando no es posible utilizar una prueba matemática formal para establecer la validez de la solución, ya sea porque el problema es demasiado complejo, no sea posible formularlo y los criterios de solución no están en un marco formal. Sin embargo, las construcciones y supuestos del problema son tan suficientemente precisas, que se puede construir un argumento lógico acerca de la hipótesis de la solución. Este método podría servir como un suplemento o como alternativa a la evaluación experimental y validación de la solución.

Esto suele ser una forma más débil de la validación que una prueba matemática o el uso de la validación experimental. Los pasos para esta forma de validación son:

1. Identificar las suposiciones (axiomas) que están relacionadas con el problema de investigación.
2. Identificar las reglas (reglas de deducción) que están relacionadas con el problema o la solución.
3. Construir un camino lógico de las hipótesis (axiomas) acerca de la solución (hipótesis), utilizando las reglas de deducción que se han revelado.

Pruebas Matemáticas

Este método se encarga de demostrar matemáticamente las afirmaciones realizadas sobre la solución que se ha desarrollado para el problema de investigación. Se utiliza cuando las afirmaciones hipotéticas para la solución pueden expresarse cuantitativamente, y los aspectos esenciales del problema y la solución se pueden expresar formalmente, en un sistema lógico cerrado.

Esta forma de validación adopta los siguientes pasos:

1. Expresar las afirmaciones sobre la hipótesis de la solución de forma cuantitativa y precisa.
2. Expresar la pretensión de ser probado como un teorema en un lugar bien definido, sistema cerrado de lógica formal.
3. Demostrar los resultados auxiliares (lemas) que pueden ayudar a demostrar el teorema acerca de las afirmaciones sobre la hipótesis de la solución.
4. Demostrar el teorema de declaraciones, que pueden utilizar los lemas ya probados.

Este modelo ofrece la forma más fuerte de validación de las reclamaciones realizadas sobre la solución. Esta validación es incluso más fuerte que la validación experimental.

Todos los métodos de validación previamente expuestos, varían en función de su competitividad y la fuerza con la que pueden establecer la validez de una solución. El método de Pruebas Matemáticas proporciona la forma más fuerte de validación. La potencia del método de Razonamiento Lógico depende de la fuerza y la precisión de sus argumentos y suposiciones. En general, es una alternativa o complemento al uso de los métodos de Experimentación y Simulación. Estos últimos son útiles cuando el problema es complejo y no susceptible de una demostración matemática. El empleo del patrón de Uso de Métricas es valioso en la Experimentación, Simulación, y los métodos de Pruebas Matemáticas. Además ayuda en la cuantificación de las afirmaciones acerca de la solución. El modelo de Evaluación Comparativa es una forma más débil del método de Uso de Métricas, pero es muy útil junto con los métodos de Experimentación y Simulación, se utiliza cuando los sistemas de medición adecuados no están disponibles (Vaishnavi, et al., 2008).

El uso de uno o más métodos pueden ayudar a convencerse a sí mismo y la comunidad investigadora, de la validez y valor de la solución. Esto, a su vez, es muy importante en la publicación de los resultados. En este escenario, para evaluar y probar la validez de la solución propuesta en esta investigación, se adoptará el método Demostración, por ser apropiado para un procedimiento novedoso y que resuelve un problema para el cual no existe solución previa. Por último, la demostración constituye el método de validación más utilizado en las publicaciones científicas para el área de conocimiento de las ciencias de la computación, de acuerdo a los trabajos de (Shaw, 2002), (Cañete, 2002).

3.2 Recursos utilizados en la validación

Las pruebas de validación, aplicando el método de la Demostración, se realizarán con la ayuda de una computadora ACPI Multiprocessor PC con una motherboard Intel Rogers City DG965RY, incorporando un procesador DualCore Intel Core 2 Duo E4500 a 2.20GHz y una memoria DDR2 SDRAM Kingston 9905320-007.A00LF con 1GB de capacidad. El sistema operativo sobre el cual se ejecutarán las pruebas, será Microsoft Windows XP Professional publicada en el año 2002, versión a la que se le incorporó el paquete de corrección de errores Service Pack 3.

3.3 Descripción de los casos de prueba

Se utilizarán tres fracciones de bases de datos reales pertenecientes a algunas declaraciones de mercancías de la aduana, transformadas a transacciones con el formato de entrada (ver **epígrafe 2.3**) requerido por el algoritmo de eliminación de itemsets y una ontología desarrollada a partir del “Capítulo 61” del Sistema Armonizado de Clasificación de Productos (SACLAP) de la Aduana General de la República. En cada uno de los casos se evaluará el tiempo de ejecución del algoritmo, se comprobará y demostrará la reducción de la cantidad de ítems y su consecuencia en el tamaño de la instancia de entrada a los algoritmos de identificación de itemsets frecuentes. A continuación se describen los tres casos de prueba:

- ✓ El primer caso de prueba (**Datos 1**), poseerá una base de datos de 1000 transacciones (filas) y 33 artículos o elementos (columnas). Teniendo en cuenta la heurística del Apriori y el formato

(ver **epígrafe 2.3**) requerido por el algoritmo de eliminación de itemsets, es necesario transformar cado uno de estos artículos en la combinación sucesiva de sus valores distintos, obteniendo como resultado 5138 nuevos elementos (columnas) o ítems. Como resultado de este proceso, se tendría una entrada de base de datos transaccional al algoritmo de 5138 ítems y 1000 transacciones.

- ✓ El segundo caso de prueba (**Datos 2**) contará con una base de datos de 3000 transacciones (filas) y 33 artículos o elementos (columnas). Se tendría una entrada de base de datos transaccional al algoritmo de 12593 ítems y 3000 transacciones.
- ✓ El tercer caso de prueba para la validación (**Datos 3**), tendrá una base de datos de 6000 transacciones (filas) y 33 artículos o elementos (columnas). Traduciéndose esta entrada en 13374 ítems y 6000 transacciones.

La Tabla 5 muestra las características anteriormente explicadas de cada caso:

Datos	Ítems	Transacciones	Columnas	Tamaño instancia
Datos 1	5138	1000	33	26399044×10^3
Datos 2	12593	3000	33	475750947×10^3
Datos 3	13374	6000	33	1073183256×10^3

Tabla 5. Conjunto de datos utilizados en los casos de prueba.

Además se tendrá en cuenta para los tres casos de prueba previamente expuestos, la interpretación de un grupo de restricciones de eliminación, divididas en restricciones de particionamiento vertical y horizontal respectivamente (ver **epígrafe 2.3**):

- ✓ El **Caso 1** estará compuesto por una restricción de particionamiento horizontal y una restricción de particionamiento vertical, permitiendo mediante estas últimas, aplicar las limitaciones a dos columnas de la base de transacciones.
- ✓ Para el **Caso 2** se tendrán en cuenta dos restricciones de particionamiento horizontal y dos restricciones de particionamiento vertical. Estas últimas permiten involucrar en el proceso a tres columnas de la base de datos transaccional.

- ✓ En el **Caso 3** estarán presente cuatro restricciones de particionamiento horizontal y cuatro restricciones de particionamiento vertical, las mismas permitirán establecer las limitantes a ocho columnas de la base de transacciones.

La siguiente tabla resume lo previamente descrito.

Casos	Restricciones Verticales	Columnas Afectadas	Restricciones Horizontales	Total
1	1	2	1	2
2	2	3	2	4
3	4	8	4	8

Tabla 6. Cantidad de restricciones establecidas para los casos de prueba.

Estos conjuntos de restricciones serán aplicados en un primer momento de la siguiente forma: **Caso 1** de restricción con **Datos 1** de base de datos, y así sucesivamente con los restantes conjuntos de datos. En un segundo momento se aplicarán los tres casos de restricciones a cada fracción de las bases de datos transaccionales seleccionadas, con el objetivo de establecer una comparación entre las mismas, teniendo en cuenta el esfuerzo o tiempo de respuesta del algoritmo, la cantidad de ítems y/o transacciones eliminadas en cada caso respectivamente, así como el grado de reducción que estas representan.

La fase de interpretación de las restricciones de eliminación, contará con la ayuda de una ontología de dominio específico, permitiendo aprovechar el conocimiento previo expresado en ella mediante una taxonomía de conceptos, enriquecida por las propiedades de los datos (ver Figura 19).

```
<SubClassOf>
    <Class IRI="#pais"/>
    <Class IRI="#mercancia"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#plazo"/>
    <Class IRI="#mercancia"/>
</SubClassOf>
<DataPropertyDomain>
    <DataProperty IRI="#aduanadesp"/>
    <Class IRI="#declaracion"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#annoregistro"/>
    <Class IRI="#declaracion"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#cantart"/>
    <Class IRI="#articulo"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#cantbultos"/>
    <Class IRI="#carga"/>
</DataPropertyDomain>
```

Figura 19. Fragmento de la ontología de dominio específico.

3.4 Discusión de los resultados

Al aplicar el algoritmo de eliminación de itemsets utilizando los datos descritos anteriormente, se obtuvo los siguientes resultados:

- ✓ **Caso 1:** Mediante la aplicación de este caso de prueba al algoritmo de eliminación de itemsets, se interpretó una restricción de particionamiento vertical y una de particionamiento horizontal, para un total de dos restricciones. La primera permitió involucrar en el proceso dos columnas de la base de transacciones, posibilitando al algoritmo la eliminación de 1025 ítems, obteniéndose como resultado un total de 4113 ítems de los 5138 que se tenían inicialmente. La segunda posibilitó disminuir el número de transacciones de 1000 a 795, o sea, se eliminaron 205 transacciones.

- ✓ **Caso 2:** Durante la ejecución de este caso de prueba se interpretaron cuatro restricciones en total, dos de particionamiento vertical y dos de particionamiento horizontal. Las primeras permitieron involucrar en el proceso tres columnas de la base de datos, posibilitando al algoritmo la eliminación de 4364 ítems, obteniéndose como resultado un total de 8229 ítems de los 12593 que se tenían al inicio. Las segundas permitieron disminuir el número de transacciones de 3000 a 2602, excluyendo así 398 transacciones.
- ✓ **Caso 3:** Unas vez ejecutado el algoritmo sobre este caso de prueba, se interpretaron ocho restricciones en total, cuatro de particionamiento vertical y cuatro de particionamiento horizontal. Las restricciones verticales afectaron durante el proceso ocho columnas de la base de datos, posibilitando la eliminación de 9948 ítems, obteniéndose como resultado un total de 3426 ítems de los 13374 que tenía el caso de prueba al inicio del procedimiento. Con las restricciones horizontales se eliminaron 3991 transacciones, por lo que hubo una disminución de 6000 a 2009.

La siguiente tabla relaciona los resultados expuestos.

Entrada		Salida			Tiempo ejecución (min)	Reducción tamaño instancia	GR (%)
Casos	**Tamaño instancia**	**Ítems**	**Transacciones**	**Tamaño instancia**			
1	26399044×10^3	4113	795	13448831355	0.48	-12950212645	49
2	475750947×10^3	8229	2602	139631301342	8.39	-336119645658	70
3	1073183256×10^3	3426	2009	23580589284	22.00	-1049602666716	97

Tabla 7. Resultados del primer momento de ejecución de los casos de prueba.

En la tabla presentada (**Tabla 7**) se puede observar que hubo una reducción del tamaño de la instancia (**Reducción tamaño instancia**) de -12950212645; -336119645658 y -1049602666716

respectivamente para cada caso, lo que representa un grado de reducción (**GR**) del 49; 70 y 97%. Este indicador permite conocer que tan efectivo se comportó el algoritmo para cada combinación de restricciones de eliminación definidas sobre los conceptos y/o propiedades del dominio. El **GR** de la ejecución del algoritmo se calcula atendiendo al porciento que representa la reducción de la longitud de la instancia con respecto al tamaño total de la instancia de entrada. Estos porcentajes se calcularon aplicando el tercer caso de tanto porciento, conocido comúnmente como "regla de tres". Para realizar el referido cálculo se aplicó la siguiente formulación.

Tamaño instancia total: **TIT**
Reducción tamaño instancia: **RTI**
Grado de reducción: **GR**

$$GR = \frac{RTI * 100}{TIT}$$

En el segundo momento de evaluación y validación del algoritmo, se aplicaron todos los conjuntos de restricciones sobre los tres fragmentos de base de datos seleccionados de las declaraciones de mercancías de las aduanas. Durante el proceso se interpretaron nueve restricciones en total, cuatro de particionamiento vertical y cinco de particionamiento horizontal, donde las primeras permitieron afectar ocho columnas de la base de datos transaccional. A continuación se describen los resultados obtenidos:

- ✓ **Caso 1:** Unas vez ejecutado el algoritmo sobre este caso de prueba se eliminaron 2115 ítems de los 5138 que se tenían al inicio del procedimiento, obteniéndose como resultado un total de 3023 ítems. Con las restricciones horizontales se eliminaron 681 transacciones, por lo que hubo una disminución de 1000 a 319.
- ✓ **Caso 2:** Finalizado el proceso de interpretación de este caso, las restricciones verticales posibilitaron al algoritmo la eliminación de 5584 ítems, obteniéndose como resultado un total de 7009 ítems de los 12593 que se tenían al inicio. Las restricciones horizontales permitieron disminuir el número de transacciones de 3000 a 950, excluyendo así 2050 transacciones.

- ✓ **Caso 3:** En este caso el algoritmo logró excluir 9746 ítems, contando la instancia con solo 3628 ítems de los 13374 que tenía el caso de prueba inicialmente. A través de las restricciones horizontales se logró disminuir el número de transacciones de 6000 a 1957, es decir, se eliminaron 4043 transacciones.

En la siguiente tabla se resumen los resultados previamente expuestos del segundo momento de ejecución de los casos de prueba de evaluación y validación del algoritmo.

Entrada		Salida			Tiempo ejecución (min)	Reducción tamaño instancia θ	GR (%)
Casos	**Tamaño instancia**	**Ítems**	**Transacciones**	**Tamaño instancia θ**			
1	26399044×10^3	3023	319	2915190751	0.55	−23483853249	88
2	475750947×10^3	7009	950	46669776950	7.19	−429081170050	90
3	1073183256×10^3	3628	1957	25758785488	21.49	−1047424470512	97

Tabla 8. Resultados del segundo momento de ejecución de los casos de prueba.

A partir de las pruebas desarrolladas en este segundo momento de evaluación y validación del algoritmo propuesto, se pudo determinar que la cantidad de restricciones de eliminación, tanto de particionamiento vertical como horizontal, influyen de manera directa en el grado de reducción del tamaño de la instancia de entrada. Las reducciones de estas pruebas representaron el 88; 90 y 97% respectivamente para cada caso. En este escenario, si la cantidad de restricciones de particionamiento horizontal es mayor que el número de restricciones verticales, y viceversa, la cantidad de itemsets eliminados por el algoritmo, disminuye con respecto a la cantidad de transacciones eliminadas, y viceversa.

Además, analizando los tiempos de ejecución de cada caso de prueba, se puede determinar que el esfuerzo o tiempo de respuesta del algoritmo, depende en gran medida de la cantidad de restricciones

de eliminación que se definan sobre el dominio y del volumen de datos que contenga el *data set* seleccionado. Mientras mayor sea el número de limitaciones que se especifiquen, mayor será el esfuerzo del algoritmo en eliminar todos los elementos contenidos en las restricciones. Igualmente, mientras mayor sea el volumen de datos del *data set*, mayor será la cantidad de recorridos sobre los datos en el proceso de eliminación.

Este análisis nos permite concluir en la conveniencia de particionar el número de transacciones de la base de datos de entrada al algoritmo, en fragmentos de 1000 transacciones, ganando rapidez en la ejecución de la solución propuesta para este tipo de procesamiento.

A partir de la ejecución de estos casos de prueba de evaluación y validación, se obtuvieron resultados que demuestran que el algoritmo de eliminación de itemsets no significativos para el usuario, permite reducir la instancia de entrada a los procedimientos de identificación de conjuntos de ítems frecuentes.

3.5 Conclusiones parciales

Se realizó un estudio de los principales métodos de evaluación y validación, adoptando una posición a favor del método Demostración, siendo uno de los más utilizados y referenciados en el mundo de las soluciones informáticas (Shaw, 2002). Se definieron los elementos fundamentales de la demostración propuesta y se describieron las características de los recursos de *Software* y *Hardware* empleados para la ejecución de las pruebas. Para la evaluación y validación del algoritmo propuesto, se emplearon tres casos de prueba, compuestos cada uno respectivamente, por tres fracciones de base de datos de las declaraciones de mercancías de las aduanas; por tres conjuntos de restricciones de eliminación y la ayuda del conocimiento previo expresado en una ontología de dominio específico. A través de estos, se demostró que la solución propuesta es capaz de eliminar un conjunto de ítems y/o transacciones, reduciendo la instancia de entrada a los algoritmos de identificación de itemsets frecuentes. Se determinó que el grado de reducción de la instancia de entrada y el esfuerzo o tiempo de respuesta del algoritmo propuesto, dependen en gran medida del número de restricciones que se definan sobre el dominio y del volumen de datos que contenga el *data set* seleccionado. Por último, los resultados obtenidos en la ejecución de las pruebas, permitieron demostrar que la solución es confiable y de gran validez en el tratamiento del problema de esta investigación.

CONCLUSIONES

En este trabajo se implementó un algoritmo de eliminación de itemsets para reducir el tamaño de la instancia en el procesamiento de reglas de asociación, usando conocimiento previo del negocio expresado en una ontología de dominio. Los resultados alcanzados en la demostración de la solución permiten concluir lo siguiente:

- ✓ Se desarrolló un mecanismo que permite definir un conjunto de limitaciones o restricciones de eliminación sobre el dominio.
- ✓ La aplicación del algoritmo desarrollado disminuye la cantidad de itemsets y por tanto el tamaño de la instancia de entrada a los algoritmos de minado de reglas de asociación.
- ✓ Existe una relación directa entre el tamaño de la entrada y el grado de reducción que alcanza el algoritmo.
- ✓ A medida que se eliminan conjuntos de ítems y/o transacciones el grado de reducción crece, pero es posible que se pierdan reglas relevantes.

RECOMENDACIONES

Se recomienda integrar la presente investigación con la solución desarrollada por el estudiante Carlos Alberto Hernández Miranda, “Generalización de Itemsets en la etapa de pre-procesamiento para la extracción de reglas de asociación” y con el algoritmo Apriori-Like, desarrollado por el estudiante Andy Fernández Garabote para el minado de reglas de asociación. La combinación de estas soluciones permitiría contar con un marco integrado para la minería de reglas de asociación. Con el objetivo de optimizar la solución, se exhorta la implementación del algoritmo de eliminación de itemsets en un lenguaje de bajo nivel, esto permitiría incluir la solución en una librería de PHP, posibilitando su integración con el sistema Gestión Integral de Aduanas (GINA). Se recomienda particionar la base de datos transaccional de entrada al algoritmo en fracciones de 1000 transacciones, mejorando así el tiempo de ejecución del mismo. También sería recomendable incorporar otros mecanismos de reducción de la instancia de entrada a los algoritmos de identificación de itemsets frecuentes en la etapa de pre-procesamiento.

BIBLIOGRAFÍA

Agrawal, R., Imielinski, T., Swami, A. 1993.***Mining Association Rules Between Sets of Items in Large Databases.*** **SIGMOD Conference : s.n., 1993.**

Agrawal, R., Srikant, R. 1994.***Fast Algorithms for Mining Association Rules.*** **Santiago, Chile. : s.n., 1994.**

Ahn, KI, Jae-Yearn Kim. 2004.***Efficient Mining of Frequent Itemsets and a Measure of Interest for Association Rule Mining. Journal of Information & Knowledge Management.*** **2004.**

Bellandi, Andrea, Grossi, Valerio, Romei, Andrea. 2007.***Pushing constraints in association rule mining: an ontology-based approach.*** **Italy : s.n., 2007.**

Berndtsson, Mikael , et al. 2008. ***Thesis Projects: A Guide for Students in Computer Science. 2008.***

Berzal F., Blanco I., Sanchez D., Vila M.A. 2001.***A new framework to assess association rules. Proceedings of the 4th International Conference on Intelligent Data Analysis.*** **2001.**

Cañete. 2002. **¿Qué se entiende, en España, por Investigación en Ingeniería del Software? s.l. : MIFISIS, 2002.**

Cannataro M., Guzzi P. H., Mazza T., Tradigo, G., Veltri, P. 2007.***Using ontologies for preprocessing and mining spectra data on the Grid.*** **2007.**

Cannataro, M., Comito, C. 2003.***A Data Mining Ontology for Grid Programming.*** **Budapest : s.n., 2003.**

Céspedes, Ramírez, Zulia.***Las ontologías como herramienta en la Gestión del Conocimiento.*** **Ciudad de La Habana,Cuba : s.n.**

Cruz Concepción, Raimil, Cruz, liz Mary, Reyes Hernández, Liannet Lucia.***Utilización de la Representación Formal de Conocimiento en la Toma de decisiones.***

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. 1996.***Advances in Knowledge Discovery and Data Mining, chapter From Data Mining to Knowledge Discovery.*** **1996.**

Frawley, W., Piatetsky-Shapiro, G., Matheus, C. 1992.***Knowledage discovery in databases: An overview.*** **1992.**

Galvis Mario, Martínez Francisco. 2004.***Confrontación de dos técnicas de minería de datos aplicadas a un dominio específico.*** **Bogotá D.C. : s.n., 2004.**

Hendler, Jim. 2004.***Preguntas frecuentes sobre el Lenguaje de Ontologías Web (OWL) del W3C.*** **2004.**

Hernández Orallo, J., Ramírez Quintana, J., Ramírez Ferri, C. 2004.***Introducción a la Minería de datos. Universidad Politécnica de Valencia.*** **Madrid : s.n., 2004.**

Holt J.D., Chung S.M. 1999.***Efficient Mining of Association Rules in Text Databases.*** **1999.**

Holt J.D., Chung S.M. 2002.***Mining association rules using inverted hashing and pruning.*** **2002.**

Marinica, Claudia, Guillet, Fabrice, Briand, Henri.***Post-Processing of Discovered Association Rules Using Ontologies.*** **Ecole polytechnique de l'Université de Nantes, France : s.n.**

Nuñez, Ramos, Esmeralda, Haydemar. 2007.***Ontologías:Componentes, metodologías, lenguajes,herramientas y aplicaciones.*** **Caracas : s.n., 2007.**

Park J.S., Chen M.S., Yu P.S. 1997.***Using a Hash-Based Method with Transanction Trimming and Database Scan Reduction for Mining Association Rules. .*** **1997.**

Pyle, D.***Data Preparation for Data Mining.***

Ramakrishnan Srikant, Quoc Vu, Rakesh Agrawal.***Mining Association Rules with Item Constraints.*** **IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, U.S.A : s.n.**

Savasere, A., Omiecinski, E., Navathe, S. 1995.***An Efficient Algorithm for Mining Association Rules in Large Databases, Data Mining Group, Tandem Computers, Inc.*** **Austin, TX, U.S.A. : s.n., 1995.**

Shaw, M. 2002. ***What makes good research in software engineering.*** **s.l. : FOR TECHNOLOGY TRANSFER (STTT). SPRINGER BERLIN / HEIDELBERG, 2002.**

Toivonen, H. 1996.***Sampling Large Databases for Association Rules.*** **Bombay, India. : s.n., 1996.**

Triantaphyllou, Evangelos. 2010.***Data Mining And Knowledge Discovery Via Logic-Basedmethods.*** **Louisiana State University, Baton Rouge, Louisiana, USA : s.n., 2010.**

Vaishnavi, Vijay K. and Jr, William Kuechler. 2008. ***Design Science Research Methods and Patterns. 2008.***

Xiong, Xia Shi, Fan, Li, Lei, Zhang. 2010.***Ontology-based Association Rule Quality Evaluation Using Information Theory.*** **School of Computer Science and Technology, China University of Mining and Technology Xuzhou Jiangsu China : s.n., 2010.**

1