

Hola, tesis doctoral en informática !!!

Índice general

1. SRW	9
1.1. Personalización de la Web	10
1.2. Minería de Uso Web	10
1.2.1. DATOS	10
1.2.2. Selección	10
1.2.3. Preproceso	10
1.2.4. Transformación	10
1.2.5. Minería de Datos (DM)	10
1.2.6. Evaluación e Integración	10
1.2.7. CONOCIMIENTO	10
1.3. Minería de Datos	10
1.3.1. Mapas de Navegación Web	10
1.3.2. Reglas de Asociación	10
1.4. Publicaciones	10
1.4.1. Actas de HCII'05	10
1.4.2. Actas de Interacción'05	10
1.4.3. Actas de SICO'05	10
2. ARM	11
2.1. Conceptos básicos	11
2.1.1. Tipo de Datos	11
2.1.2. Primeros algoritmos	11
2.1.3. Formato de D	11

2.1.4.	Fases de ARM	11
2.2.	Minería de Itemsets Frecuentes	11
2.2.1.	Algoritmos y estructuras	12
2.2.2.	Evaluación de diferentes implementaciones	12
2.3.	Generación de Reglas de Asociación	12
2.3.1.	genrules()	12
2.3.2.	Apriori2	12
2.4.	El Ítem Raro	12
2.4.1.	Estudio de ítems raros	12
2.4.2.	Reglas de Oportunidad	12
2.5.	Publicaciones	12
2.5.1.	HCII'07	12
2.5.2.	ESWA, vol. 35(3) 2008	13
2.5.3.	Interacción'10	13
3.	ARM para Clasificación	15
3.1.	Catálogo	16
3.2.	Catálogo comprimido	17
3.2.1.	Lectura de catálogos comprimidos	17
3.3.	Catálogo completo	17
3.4.	Publicaciones	18
3.4.1.	[...]	18
3.4.2.	[...]	19
4.	Conclusiones y Trabajo Futuro	21
A.	Notación	25
A.1.	Sistemas de Recomendación Web	25
A.2.	Minería de Reglas de Asociación	25
A.3.	Catálogos	25
B.	Código	27
B.1.	Sistemas de Recomendación Web	27
B.2.	Minería de Reglas de Asociación	27
B.3.	Catálogos	27
C.	Datos utilizados	31
C.1.	Sistemas de Recomendación Web	31
C.2.	Minería de Reglas de Asociación	31
C.3.	Catálogos	31
	Índice de figuras	33

	5
Índice de cuadros	35
List of Theorems	37
Índice de definiciones	37
Índice de listados	39
D. Sobre la bibliografía	41
Bibliografía	41

Motivación

Este...

SRW

Los SRW surgen de...

Catálogos

Un catálogo es...

1

Sistemas de Recomendación Web

En ...

1.1. Personalización de la Web

1.2. Minería de Uso Web

1.2.1. DATOS

1.2.2. Selección

1.2.3. Preproceso

1.2.4. Transformación

1.2.5. Minería de Datos (DM)

1.2.6. Evaluación e Integración

1.2.7. CONOCIMIENTO

1.3. Minería de Datos

1.3.1. Mapas de Navegación Web

1.3.2. Reglas de Asociación

1.4. Publicaciones

1.4.1. Actas de HCII'05

1.4.2. Actas de Interacción'05

1.4.3. Actas de SICO'05

Minería de Reglas de Asociación (ARM)

Una breve introducción para enlazar con el capítulo anterior y paso a relatar...

2.1. Conceptos básicos

2.1.1. Tipo de Datos

...

2.1.2. Primeros algoritmos

...

2.1.3. Formato de D

...

2.1.4. Fases de ARM

...

2.2. Minería de Itemsets Frecuentes

Frequent Itemset Mining...

2.2.1. Algoritmos y estructuras

...

2.2.2. Evaluación de diferentes implementaciones

...

2.3. Generación de Reglas de Asociación

...

2.3.1. genrules()

...

2.3.2. Apriori2

...

2.4. El Ítem Raro

...

2.4.1. Estudio de ítems raros

...

2.4.2. Reglas de Oportunidad

...

2.5. Publicaciones

...

2.5.1. HCII'07

...

2.5.2. ESWA, vol. 35(3) 2008

...

2.5.3. Interacción'10

...

ARM para Clasificación

Uno de los problemas de la búsqueda de reglas de asociación mediante equipos informáticos es su necesidad de memoria RAM para guardar todos los *itemsets* frecuentes (\mathcal{L}) en un lugar de rápido acceso para ser más eficientes en la búsqueda de *reglas de asociación*. El *dilema del ítem raro* puede aparecer en cualquier momento, dependiendo de los datos que contenga el almacén \mathcal{D} que estemos analizando, y se siguen proponiendo ideas para aliviarlo como el uso de umbrales automáticos para el *soporte* propuesto por Sadhasivam y Angamuthu (2011). Sería interesante poder averiguar, en base a información básica del fichero \mathcal{D} , qué requisitos de memoria RAM necesitamos, de cuáles disponemos y, con ello, deducir hasta qué *soporte* mínimo podemos trabajar con esa colección específica de datos en este equipo en particular. Hay estudios sobre ello [cita sobre mushroom.dat de gAcademy]. Aunque este proceso puede llevar algo de tiempo (y restar eficiencia global al algoritmo de *FIM*) permite ahorrar el tiempo perdido cuando el programa deja de funcionar por falta de RAM y no es capaz de ofrecernos ningún resultado.

Esta reflexión nos llamó la atención en ficheros tan «pequeños» como mushroom.dat o chess.dat. mushroom.dat es una colección de datos ampliamente utilizada debido a su publicación en UCI - Machine Learning Repository¹, donde podemos encontrar una completa descripción de los datos en sí y de su aparición en el mundo de la investigación. Encontramos información sobre el uso de este almacén \mathcal{D} en artículos de *clasificación*.

Las reglas de *clasificación* son otra línea de investigación en auge (Liu, Hsu y Ma, 1998)

Estos ficheros han sido analizados en muchos artículos desde el punto de vista de la *Minería de Reglas de Asociación* clásica (Suzuki, 2004; Borgelt, 2004; Thabtah, Cowling y Hamoud, 2006; Li y Zhang, 2010; Malik y Raheja, 2013; Ritu y Arora, 2014;

Añadir más citas

¹ <https://archive.ics.uci.edu/ml/datasets/Mushroom>

Sahoo, Das y Goswami, 2014). Se proponen nuevas medidas, como la *utilidad* de los *itemsets* (Wu y col., 2012) para aliviar el *dilema del ítem raro*. En todos los casos se ha de recurrir al *soporte* mínimo para poder llevar a cabo el análisis en un tiempo prudencial (algo más de 2 segundos en el caso de Ritu y Arora, un estudio muy reciente cuyos gráficos coinciden con los de Malik y Raheja). Intentamos aplicar técnicas de *Minería de Datos* a colecciones relativamente pequeñas (mushroom.dat sólo contiene $8\,124 \times 23$ datos) y no podemos profundizar o trabajar en tiempo real si no aplicamos recortes drásticos de información. Hoy en día, almacenes \mathcal{D} de este tamaño deberían poder analizarse en tiempos razonables sin prescindir de ninguno de sus datos.

Antes de obtener información sobre mushroom.dat comenzamos a analizar los resultados obtenidos en nuestros propios experimentos sobre este almacén \mathcal{D} . Al observarlos encontramos una estructura concreta: todas las filas («transacciones») tienen el mismo número de datos y en la primera columna sólo aparece un 1 o un 2, en la segunda sólo un 3, 4 o 5... Se trata de una estructura muy rígida y con exceso de información. Al descubrir que el 1 aparecía en XXXX transacciones dedujimos que el 2 aparecería en las restantes $8\,124 - XXXX$. En el proceso de *Minería de Itemsets Frecuentes* estábamos desperdiciando esta información y nos entreteníamos en contar el número de veces que aparecía el ítem 2. También descubrimos que el ítem 3 aparecía en un total de YYYY transacciones, en ZZZZ ocasiones junto al ítem 1, luego en las restantes YYYY - ZZZZ veces que aparece ha de estar junto al ítem 2. No necesitamos averiguar nada sobre el ítem 2, lo que averigüemos sobre el ítem 1 nos dará toda la información que tiene \mathcal{D} sobre el ítem 2. No necesitamos utilizar memoria RAM para guardar información sobre el ítem 2, tendremos memoria RAM libre para analizar los *ítems raros* más frecuentes de \mathcal{D} .

Tras llevar a cabo la investigación que se expone en este capítulo podemos afirmar que, aunque todas las citas usadas en los párrafos anteriores han experimentado con mushroom.dat para aliviar su *dilema del ítem raro*, mushroom.dat y muchas colecciones de datos de similares características realmente no contienen *Ítems Raros*.

3.1. Catálogo

Los catálogos son colecciones de registros preparadas para resolver informáticamente un problema de clasificación. Y muchos investigadores de esta especialidad publican sus datos para que otros investigadores puedan hacer pruebas con las mismas condiciones de partida: una colección de datos con ciertas características. En UCI, KEEL, LUCS... encontraremos muchos catálogos entre los datasets que publican para resolver problemas de clasificación.

Cuando no sabíamos que esos ficheros contenían catálogos intentábamos aplicar bien conocidos algoritmos de ARM pero no podíamos extraer información que contienen los datos porque se desbordaba la RAM del equipo en que se está aplicando el algoritmo y se abortaba el proceso tras horas de cálculos que finalmente no obteníamos. Esto nos

Acabo de descubrir LUCS, que discretiza las colecciones de UCI y me ofrece 97 valores distintos en adult, frente a los 27245 que tiene el

sorprendía porque el primer catálogo que intentamos analizar con Apriori sólo tiene 5 644 registros de 23 datos, no son números excesivos para un problema de Minería de Datos analizado con un ordenador de escritorio con cierta potencia y capacidad de RAM. Eso nos llevó a descubrir cómo se creó el catálogo a través de UCI/mushroom. . .

Los catálogos caracterizan un problema de clasificación concreto. Si queremos plantear otro problema de clasificación, bien etiquetando a los mismos individuos en otras clases o bien utilizando atributos diferentes no podemos utilizar directamente cualquier catálogo que tengamos sobre la misma población. Si los dos problemas usaran los mismos atributos pero diferentes clases y las clases en estudio son independientes no servirá de nada la información que tengamos sobre los catálogos completos del primer problema de clasificación si no sabemos analizar qué información puede ser relevante y cuál no, de hecho la información menos relevante en esta situación es la distribución de las clases en cada uno de los problemas de clasificación por lo que debemos huir de interpretaciones erróneas utilizando estos datos para estimar soportes o confianzas poblacionales.

De un catálogo se puede extraer información válida para otro problema de clasificación que utilice los mismos atributos ya que si en la muestra en que se basa el catálogo no presenta cierta relación entre los valores de los atributos YA SABEMOS QUE NO APARECERÁ ESA RELACIÓN AUNQUE CAMBIEMOS DE CLASES (siempre que el catálogo sea válido, aún tengo que hacer muchas definiciones sobre muestra, población, distribución de clases, problema de clasificación, atributos, clases, catálogos, catálogos completos, validez de un catálogo. . .).

Aunque la ARM busca cualquier relación entre cualquier par (o k -itemset) de valores de D , el objetivo del problema de clasificación es siempre el mismo, etiquetar cada registro con una clase basándose en la información disponible sobre otros registros con valores idénticos en sus atributos.

3.2. Catálogo comprimido

Aprovechando las restricciones implícitas de los catálogos como mushroom. . .

3.2.1. Lectura de catálogos comprimidos

3.3. Catálogo completo







En [...] expusimos. . .

3.4. Publicaciones

En Interacción'12 comenzamos a publicar nuestros resultados sobre las colecciones de *transacciones* estructuradas mediante *catálogos*. Los mayores avances los hemos ob-

tenido este último año y estamos en un proceso actual de publicación de resultados en diversas revistas y un nuevo congreso internacional.

Efficient analysis of transactions to improve web recommendations, 2012

Lazcorreta Puigmartí, E., Botella Beviá, F. y Fernández-Caballero, A. Efficient analysis of transactions to improve web recommendations. *Actas del XIII Congreso Internacional de Interacción Persona-Ordenador*, ACM International Conference Proceeding Series, 2012.   Scopus    

Resumen

When we deal with big repositories to extract relevant information in a short period of time, pattern extraction using data mining can be employed. One of the most used patterns employed are Association Rules, which can measure item co-occurrence inside large set of transactions. We have discovered a certain type of transactions that can be employed more efficiently that have been used until today. In this work we have applied a new methodology to this type of transactions, and thus we have obtained execution times much faster and more information than that obtained with classical algorithms of Association Rule Mining. In this way we are trying to improve the response time of a recommendation web system in order to offer better responses to our users in less time. Copyright 2012 ACM.

3.4.1. [...]

En la actualidad estamos completando la redacción de un artículo sobre esta fase de nuestra investigación, que será enviada en breve para su revisión y posible publicación en revista. En él se exponen la teoría y experimentos que forman la sección 3.3.

3.4.2. [...]

Como no somos especialistas en *Clasificación* presentaremos nuestra aportación a esta rama de la ciencia en el Congreso Internacional Interacción'16, que se celebrará en septiembre de 2016 en Salamanca.

²https://www.researchgate.net/publication/266652926_Efficient_analysis_of_transactions_to_improve_Web_Recommendations Revisado en diciembre de 2015.

³<https://www.deepdyve.com/lp/association-for-computing-machinery/efficient-analysis-of-transactions-to-improve-web-recommendations-mBubBNiv7c> Revisado en diciembre de 2015.

⁴https://github.com/EnriqueLazcorreta/tesis-0/blob/master/bib/nuestros/LazcorretaBotellaFCaballero_EfficientAnalysisOfTransactionsWebRecommendations_2012.pdf Revisado en diciembre de 2015.

⁵https://github.com/EnriqueLazcorreta/tesis-0/blob/master/bib/nuestros/LazcorretaBotellaFCaballero_AnalisisEficienteDeTransacciones_12_Presentacion.pdf Revisado en diciembre de 2015.

Conclusiones y Trabajo Futuro

Las mejores ideas expuestas en esta tesis son muy simples. Desde el primer algoritmo que entra en juego, Apriori, hasta la elaboración de catálogos completos ínfimos son ideas muy simples que implementadas de forma eficiente pueden hacer lo que se le pide a la Minería de Datos: buscar una aguja en un pajar.

Los catálogos completos tienen un potencial fácil de descubrir mediante sencillas técnicas informáticas de Minería de Datos. Este trabajo presenta una teoría en torno a un tipo de datos muy utilizado que posibilita la obtención extrema de la información que contienen grandes colecciones de datos utilizando la tecnología actual en tiempo real.

Los datos bien recogidos reflejan el estado actual del mundo que nos rodea, por eso es importante poder analizarlos rápidamente utilizando en algunos casos información histórica sobre el mismo problema o bien partiendo de un nuevo problema y analizando rápidamente las características de los datos que proporciona su estudio. Si sabemos qué puede descubrir la Minería de Datos a partir de la observación de los datos que hemos recogido podremos crear algoritmos que descubran lo que estamos buscando en tiempo real y con un uso aceptable de recursos de un servidor dedicado.

Este trabajo presenta unos antecedentes que encaminan al investigador a descubrir, quizá por casualidad, las características especiales de un modelo matemático de almacenamiento de información y el uso que se está dando a estas colecciones de datos por parte de especialistas en el problema de clasificación. La aparición del problema del Minado de Reglas de Clasificación Asociativas en [...] era previsible, todas las reglas de asociación tienen un aspecto muy simple que sugiere a cualquier investigador que puede ser utilizado en el problema de clasificación. El hecho de que yo, especializado en el problema de asociación, observara los mismos datos que los especialistas en clasificación tendría que llevarnos al mismo resultado si ellos habrían alcanzado el óptimo o a un mejor resultado si yo era capaz de aportar ideas sobre cómo utilizar los elementos de ARM.

El primer descubrimiento simple y útil de esta tesis son los *catálogos comprimidos* expuestos en la sección ?? . Con ellos descubrí que el modo de aplicar técnicas de ARM en los artículos que consultaba no era del todo correcto [. . .]. No soy especialista todavía en el Problema de Clasificación por lo que algunas conclusiones de esos artículos y, sobre todo, las pruebas de eficiencia de los algoritmos que proponían, estaban fuera de mi alcance. Se me ocurrió incorporar las restricciones iniciales del problema de clasificación a un problema general de asociación. Los problemas de asociación se resuelven mediante la fuerza bruta leyendo todos los datos que tenemos y mirándolos desde distintas perspectivas, si quiero resolver un problema distinto, un problema de clasificación, usando técnicas de minería de reglas de asociación debería aprovechar, al menos, la rígida estructura de los datasets usados para clasificación (en asociación sólo hay una norma: en un registro no se cuentan los datos repetidos, lo que hace que el número de reglas de asociación que se puede buscar sea tan grande que provoque desbordamiento de memoria en los programas que intentan analizar grandes colecciones de datos). Se me ocurrió que si todos los registros han de tener un valor para cada uno de los atributos en estudio podía reducir el número de datos a procesar y las dimensiones del dataset eliminando únicamente un valor de cada atributo en todo el dataset. Al hacerlo y comprobar que la nueva colección de datos, compresión sin pérdidas de la colección original, sí se podía analizar utilizando el clásico Apriori y obtener todas las reglas de asociación que contenía empecé a asimilar mejor las características de un catálogo.

Primero descubrí características matemáticas, restricciones teóricas que me permitían reducir las dimensiones del problema original y, usando muchos recursos, obtener toda la información que contienen esas pequeñas colecciones de datos en cuanto a reglas de asociación se refiere. Pero tenía que haber algo más, las características matemáticas que utilicé en ?? me exigían usar muchos recursos y no me ofrecían información demasiado relevante, además seguía necesitando mucha RAM para trabajar con colecciones pequeñas de datos, a pesar de que ya sabía que contenían muchísima información. Quería encontrar mejor información en menos tiempo y usando menos RAM por lo que introduje la STL a mi desarrollo y comprobé en la primera aplicación que la teoría de conjuntos tenía mucho que aportar al análisis de catálogos.

Tantos años de trabajo han dado lugar a muchas ideas teóricas sobre la aplicación de técnicas de DM por lo que quedan abiertas muchas líneas de investigación que podrían ser continuación de este trabajo. Como *trabajar a nivel de bits* buscando la máxima eficiencia en el uso informático de grandes colecciones de datos, o *profundizar en el desarrollo de Clasificadores*, de *lógica difusa para agrupación de valores en atributos numéricos o de amplios rangos* y de tantas otras cosas que han ido apareciendo en el estado del arte de esta tesis y que no he podido abarcar para centrarme en obtener algo tangible mediante el método científico.

La investigación mostrada en el último capítulo de esta tesis está avalada por su implementación en el campo de la Minería de Datos utilizando la tecnología actual. El preproceso de cualquier catálogo permite crear colecciones de catálogos que pueden ser utilizadas en tiempo real en grandes problemas de clasificación que pueden ser escalados

Es evidente
que tengo
que reescribir
este párrafo.
La idea es
interesante
pero...

sin tener que renunciar en cada nuevo estudio a todo el conocimiento adquirido en estudios sobre las mismas clases. En este trabajo se ha demostrado que cualquier subconjunto de un catálogo completo puede ser tratado como catálogo completo considerando siempre la incertidumbre que puede contener, si quisiéramos utilizar los datos de un problema de clasificación en otro problema de clasificación con otras clases podríamos comenzar con los registros-tipo del primer catálogo, todos los que puedan ser clasificados en el segundo problema se incorporan al catálogo del segundo problema pero así no voy bien, lo que quería decir es que si empezamos con el menor de todos los catálogos ínfimos y vamos catalogando en la segunda clase todos sus registros podemos llegar a no tener incertidumbre (caso ideal y poco probable si la segunda clase es independiente de la primera, dato interesante) pero al menos si tenemos incertidumbre es posible que sea poca, si hacemos lo mismo con otros catálogos ínfimos podríamos descubrir qué atributos aportan más determinación al segundo problema y plantear un catálogo inicial para el segundo problema.

También queda para el futuro la agrupación de valores en los atributos numéricos. Hay ya muchas investigaciones en torno a este campo y creo que con los primeros análisis hechos a un dataset se puede obtener información que pueda ayudar al investigador a hacer las agrupaciones de modo que se pueda seguir trabajando con catálogos completos ya que el agrupamiento puede generar incertidumbre. Este aspecto es muy importante pero es mucho lo que hay que investigar para llegar a conclusiones y resultados útiles, como en "Using Conjunction of Attribute Values for Classification".



Notación

La notación usada en este informe se ha intentado ajustar a la más utilizada en la bibliografía revisada a lo largo de estos años de investigación. Por este motivo no es uniforme en los tres capítulos de investigación en que se divide esta tesis.

A.1. Sistemas de Recomendación Web

En este capítulo...

A.2. Minería de Reglas de Asociación

En este capítulo...

A.3. Catálogos

En este capítulo...



Código

Se ha desarrollado mucho código para poder comprobar todo lo que se afirma en esta tesis. Se ha trabajado del modo más estándar posible para conseguir un código eficiente y que pueda ser incorporado a otras investigaciones. Se publicará como código abierto bajo la licencia... en...

B.1. Sistemas de Recomendación Web

En este capítulo...

B.2. Minería de Reglas de Asociación

En este capítulo...

B.3. Catálogos

En este capítulo...

Listado B.1: Cabecera para lectura de ficheros KEEL

```
#ifndef TFICHEROKEEL_H
#define TFICHEROKEEL_H

#include "defs.h" //(* #include...
#include <fstream>
#include <stdio.h>
#include <iostream>
using std::cout;
using std::endl;
```

```

// #include <forward_list>
// using std::forward_list;
#include <list>
using std::list;
#include <vector>
using std::vector;
#include <string>
using std::string;
#include <map>
using std::map;
/**)

/** class TFicheroKEEL
 *
 * Esta clase es una interfaz para utilizar los ficheros que pone a nuestra
 * disposición el proyecto @link http://sci2s.ugr.es/keel/index.php KEEL @endlink
 *
 * Extrae la información de los metadatos del fichero, lee la colección de datos y
 * crea un fichero con el formato que necesita mi aplicación para gestionarlo con
 * eficiencia:
 *
 * - Se codifican los distintos valores de la clase con los códigos 0, 1...
 * - Se codifican el resto de valores mediante números enteros consecutivos sin dejar
 *   ninguno reduciendo las necesidades de RAM de los algoritmos utilizados.
 *
 * También crea el fichero D comprimido optimizando los códigos usados. Se guardan
 * también las codificaciones hechas.
 *
 * Guarda también todos los datos descriptivos del fichero, que ayudan a la toma de
 * decisiones del analista y a la elaboración de informes para las pruebas que se
 * hagan sobre estos ficheros.
 *
 * Se leen líneas de un máximo de 4096 caracteres, si el fichero tuviera líneas más
 * largas no será correcta la lectura y se podrán obtener resultados inesperados.
 *
 * @todo Mayor control sobre capacidad_linea_ y capacidad_separador_ para no usar
 *       linea_ y posicion_separador_ fuera de su alcance.
 */
class TInfoFicheroKEEL;
class TFicheroKEEL
{
public:
    static bool CompruebaSiEsKEEL(const string &nombre_fichero_datos)
    {
        FILE *fichero = fopen(nombre_fichero_datos.c_str(), "rt");
        if (!fichero)
        {
            cout << "No se ha podido abrir el fichero " << nombre_fichero_datos
                << " (Abortada la lectura de fichero KEEL)";
            fclose(fichero);
            return false;
        }

        // Busco @data, leyendo sólo las 500 primeras líneas
        int caracter = fgetc(fichero),
            num_linea = 0;
        while (caracter != EOF && num_linea < 500)
        {
            num_linea++;
            while (caracter != EOF && caracter != '\n' && caracter != '@')
                caracter = fgetc(fichero);
            if (caracter == '@')
            {
                caracter = fgetc(fichero);
                if (caracter == 'd') caracter = fgetc(fichero); else continue;
                if (caracter == 'a') caracter = fgetc(fichero); else continue;
                if (caracter == 't') caracter = fgetc(fichero); else continue;
                if (caracter == 'a') caracter = fgetc(fichero); else continue;
                // Si lee @data termina el bucle y la búsqueda
                break;
            }
            caracter = fgetc(fichero);
        }
        fclose(fichero);
        return (caracter != EOF && num_linea < 500);
    }

private:
    TFicheroKEEL(const string &ruta_ficheros_OUT, const string &nombre_fichero_KEEL);
    virtual ~TFicheroKEEL();

    bool LeeMetadatos(); //(* Métodos de lectura del fichero KEEL

```

```

bool LeeEtiqueta();
bool LeeNombre();
bool LeeTipoYDominio();
bool LeeMetadato();
bool LeeAtributo();
bool LeeInputOutput();

bool LeeDatos();
bool LeeRegistro(); /**)

// unsigned long GetNumRegistros() { return num_registros_; }
// unsigned long GetNumVariables() { return nombre_variables_.size(); }
// const int GetNumClases() const { return num_clases_; }
// unsigned long GetNumValores() { return num_valores_; }
// vector<string> *GetNombreVariables() { return &nombre_variables_; }

unsigned long GuardaD();
unsigned long GuardaDComprimido(const string &nombre_fichero_D);
// unsigned long GuardaC1();

const bool Codificado() const { return codificado_; }

void MuestraElRestoDeLinea();
int SaltaEspaciosYComas();

void ReorganizaVariables(); /**< Coloca las clases en primer lugar

unsigned long Codifica();
int BuscaDatos();
int LeeYGuardaRegistro(std::ofstream &fichero_OUT);

private:
/**) Miembros privados
string carpeta_proyecto_; /**< Donde guardar ficheros auxiliares
string nombre_fichero_KEEL_; /**< Nombre y ubicación del fichero
FILE *fichero_; /**< Fichero KEEL

int num_variables_,
    num_clases_;
unsigned long num_registros_; /**< Número de registros del fichero
unsigned long num_valores_; /**< Número de valores distintos en el fichero

/** @todo Aclarar si uso list o vector en TODOS los miembros.
/** @todo Sustituir por TAttributo
vector<string> nombre_variables_; /**< Nombres de las variables
vector< vector<string> > dominio_variables_; /**< Dominio teórico de variables
vector<char> tipo_variables_; /**< Real, entero o categórico

map<string, unsigned long> **valores_; /**< Valores leídos en el fichero

vector<string> input_, /**< Nombre de los atributos
    output_; /**< Nombres de las clases

string nombre_coleccion_;

char tipo_metadato_;

string *codigo_2_valor_;
map<string, int> **valor_2_codigo_;
bool codificado_; /**)

friend class TInfoFicheroKEEL;
};

#endif // TFICHEROKEEL_H

```




Datos utilizados

Para llevar a cabo las pruebas de rendimiento y aplicabilidad de nuestras propuestas se han usado datos propios y datos procedentes de diferentes repositorios públicos como UCI, KEEL, LUCS-KDD...

C.1. Sistemas de Recomendación Web

En este capítulo usamos datos de un servidor propio con la intención de poder utilizar las recomendaciones sugeridas por nuestra metodología en el servidor del que se obtuvieron. Son los ficheros TAL y CUAL que no publicaremos por carecer de interés su contenido, ya que el servidor del que se obtuvieron ya no está disponible y no se podría dar ninguna interpretación a los resultados obtenidos...

También usamos TAL...

C.2. Minería de Reglas de Asociación

En este capítulo seguimos trabajando con los mismos datos que en el anterior e incorporamos...

C.3. Catálogos

En este capítulo es en el que más opciones hemos tenido a la hora de seleccionar datos y probar la eficiencia y posibilidades de nuestros desarrollos. Existen muchos repositorios públicos bien documentados sobre el diseño y contenido de estos datasets y es una información que enriquece mucho la investigación...

Índice de figuras

Índice de cuadros

List of Theorems

Índice de listados

B.1. Cabecera para lectura de ficheros KEEL	27
---	----



Sobre la bibliografía

Bibliografía

- Borgelt, Christian (2004). «Efficient implementations of Apriori and Eclat». En: *Proc. of the Workshop on Frequent Itemset Mining Implementations*.  ¹. FIMI'04.
- Li, Haifeng y Ning Zhang (2010). «Mining maximal frequent itemsets on graphics processors». En: *FSKD*. ², págs. 1461-1464.
- Liu, Bing, Wynne Hsu y Yiming Ma (1998). «Integrating Classification and Association Rule Mining». En: *Knowledge Discovery and Data Mining*.  ³, págs. 80-86.
- Malik, Kuldeep Singh y Neeraj Raheja (2013). «Improving performance of Frequent Itemset algorithm». En: *International Journal of Research in Engineering & Applied Sciences* 3.3.  ⁴, págs. 168-177.
- Ritu y Jitender Arora (2014). «Intensification of Execution of Frequent Item-Set Algorithms». En: *International Journal of Recent Development in Engineering and Technology (IJRDET)* 2 (6).  ⁵.
- Sadhasivam, Kanimozhi Selvi Chenniangirivalu y Tamilarasi Angamuthu (2011). «Mining Rare Itemset with Automated Support Thresholds». En: *Journal of Computer Science* 7.3.  ⁶, págs. 394-399.
- Sahoo, Jayakrushna, Ashok Kumar Das y A. Goswami (2014). «An Algorithm for Mining High Utility Closed Itemsets and Generators». En: *ArXiv e-prints*.  ⁷.

¹ www.borgelt.net/papers/fimi_03.ps.gz

² <http://dx.doi.org/10.1109/FSKD.2010.5569206>





³ <http://www.cs.uiuc.edu/class/fa05/cs591han/papers/bliu98.pdf>

⁴ <http://www.euroasiapub.org/IJREAS/mar2013/18.pdf>

⁵ http://www.ijrdet.com/files/Volume2Issue6/IJRDET_0614_08.pdf

⁶ <http://thescpub.com/PDF/jcssp.2011.394.399.pdf>

⁷ <http://arxiv.org/pdf/1410.2988v1.pdf>

- Suzuki, Einoshin (2004). «Discovering interesting exception rules with rule pair». En: *Proceedings of the ECML/PKDD Workshop on Advances in Inductive Rule Learning*. Ed. por J. Fuernkranz.  ⁸, págs. 163-178.
- Thabtah, Fadi, Peter Cowling y Suhel Hamoud (2006). «Improving rule sorting, predictive accuracy and training time in associative classification». En: *Expert Systems with Applications* 31.2. ⁹, págs. 414-426.
- Wu, Cheng Wei, Bai-En Shie, Vincent S. Tseng y Philip S. Yu (2012). «Mining top-K High Utility Itemsets». En: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12.  ¹⁰. Beijing, China: ACM, págs. 78-86.

⁸<http://www.ke.tu-darmstadt.de/events/ECML-PKDD-04-WS/Proceedings/suzuki.pdf>

⁹<http://www.sciencedirect.com/science/article/pii/S0957417405002290>

¹⁰<http://wan.poly.edu/KDD2012/docs/p78.pdf>