Hola, tesis doctoral en informática !!!

Este es el índice de mi último informe:

Es el contenido del informe que envié a Fede el 21 de julio. Ha llovido mucho desde entonces pero hay mucho material útil, al menos hasta 3.2.

Índice general

Argumentación

Este...

SRW

Los SRW surgen de...

Catálogos

Un catálogo es...

Capítulo 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

Capítulo 2

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 Minning...

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

...

Capítulo 3

ARM sobre Catálogos

Los trabajos expuestos en los capítulos anteriores muestran una dificultad presente en muchas investigaciones en el ámbito de la informática, la imposibilidad de comprobar si los resultados obtenidos son correctos y aplicables con la tecnología actual. Todas las Ciencias recogen una Teoría que la sustenta, la Informática también, pero las demostraciones teóricas basadas en otras Ciencias no siempre son válidas para la Informática. Hay muchos artículos teóricos sobre Minería de Datos, pero algunos de ellos no evolucionan en un artículo posterior que muestre cómo se ha realizado el experimento con la tecnología actual usando datos reales. Es fácil calcular teóricamente el número de reglas de asociación presente en una colección concreta de datos, pero si el algoritmo propuesto no es capaz de almacenar en RAM todas las reglas de asociación del problema no servirá de nada ese algoritmo en esa situación. `mushroom` fue el primer caso que encontré curioso en el ambicioso campo de la Minería de Datos, una colección de tan solo 5 644 registros de 23 valores que sólo contenía 100 valores diferentes no podía ser analizada a fondo por el mejor algoritmo de ARM que conozco, Apriori, cuando el número de transacciones distintas que se puede obtener bajo estas circunstancias es X XXX XXX y con ellas tenemos un máximo de X XXX XXX XXX XXX reglas de asociación. La tecnología actual de un equipo de escritorio no puede gestionar en RAM tanta información, pero me negaba a creer que no se pudiera extraer *toda la información* que contuviera un problema tan pequeño. Al profundizar en `mushroom` y los artículos que lo mencionaban y otras colecciones de datos que encontré publicadas en los mismos portales encontré un modo de indicar al algoritmo de ARM que estoy tratando con un tipo de colecciones de datos especial. No es simplemente una colección de transacciones como las analizadas en 2.1 si no que éstas tienen unas restricciones muy fuertes en su definición.

Este informe refleja el trabajo que he realizado para descubrir lo suficiente como para ser merecedor del título de doctor en Informática. Los anteriores capítulos reflejan una gran labor de documentación y exposición científica de los resultados que podía obtener con colecciones de datos fijas, no disponía de un servidor capaz de poner a prueba nuestras aportaciones y realizar sugerencias en

tiempo real a un gran número de usuarios. Podía comprobar que mis cálculos se podían realizar con la tecnología actual y las colecciones de datos que yo manejaba pero no sabía qué ocurriría si aumentaran mis colecciones de datos o si pudiera realizar sugerencias en tiempo real. Es un buen trabajo teórico apoyado en algunas realizaciones prácticas pero del que no puedo extraer aún la conclusión que es un buen trabajo de investigación en Informática.

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.

Acabo de descubrir LUCS, que discretiza las colecciones de UCI y me ofrece 97 valores distintos en adult, frente a los 27 245 que tiene el de UCI, he de analizarlo con mi código y EXPLICAR MEJOR LAS CONSECUENCIAS DE APLICAR ANTES O DESPUÉS MI MÉTODO O LA AGRUPACIÓN DE VALORES EN ATRIBUTOS NUMÉRICOS ya que se obtendrán reglas y catálogos completos bastante diferentes, esto da para otro artículo y más si tengo en cuenta que tiene datos missing por lo que puedo obtener catálogos completos usando menos atributos con más registros o catálogos completos usando sólo los atributos registrados en cada registro (a no ser que el análisis nos diga que cierto atributo no aporta información....

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 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

3.4.1. Interacción'12

En Interacción'12 expusimos...

3.4.2. [...]

En [...]

3.4.3. [...]

Estamos terminando [...]

Capítulo 4

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 habían 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 3.2. 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 3.4 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

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

de catálogos que pueden ser utilizadas en tiempo real en grandes problemas de clasificación que pueden ser escalados 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".

Apéndice A

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...

Apéndice B

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...

Listing 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 TAtributo
      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
```

Apéndice C

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