LO QUE HEMOS HECHO:

-          Hemos visto en un libro que usan como baseline el dataset original, sin interpretar las características relacionales.  Quizá podría ser interesante que nosotros pusiéramos también las medidas que se obtienen con el primer dataset, antes de empezar a interpretar las características relacionales.  Así podríamos ver cuánto se mejora desde el primer paso hasta el final.  De momento apunto tan sólo esta idea para tenerla en cuenta en el futuro.

-          Parece que las técnicas relacionadas en el campo de Machine Learning se agrupan en tres campos:

o   XML Mining:

  Las hemos descartado pues todas están orientadas a encontrar reglas de asociación entre los nodos de los árboles XML.

o   Multi-relational data mining

  Al parecer es lo mismo que relational data mining.  El “Multi”  es un prefijo que los autores quitan o ponen en función a sus preferencias.

  Los trabajos que hemos visto en este campo intentan adaptar técnicas ya conocidas en el campo de la inducción de árboles de decisión o reglas para que sean capaces de aprender a partir de datasets expresados como “relaciones”, es decir, como predicados lógicos de primer orden con símbolos de constante tan sólo.

  Hemos analizado SCART, que es una variante de SCAR para aprender árboles de decisión:

         SCAR es para aprendizaje proposicional.

o   Toma el conjunto de entrenamiento y calcula todos los selectores de la forma A \theta v, en donde A es un atributo, \theta es un comparador (=, <>, >, <, >=, <=) y v es un valor posible de ese atributo.

o   Calcula un score para cada uno de esos selectores.

o   Se queda con el selector que tiene mejor score.

o   Al aplicar ese selector sobre el training set original se generan dos conjuntos: uno con los datos que cumplen el selector y otro con los  que no lo cumplen

o   Se vuelve a aplicar el mismo algoritmo a cada uno de los conjuntos, hasta que en una de las subdivisiones se obtiene un conjunto en el que todos los datos tienen la misma clase.

         SCART es para aprendizaje “relacional”… pero en el sentido de primer orden

o   Básicamente es lo mismo, lo que ocurre es que en lugar de trabajar con selectores trabaja con los predicados de soporte.

  También hemos analizado TILDE que es una variante de C4.5 para aprendizaje relacional.

         Es la misma idea de C4.5, pero en lugar de usar selectores, que es lo normal en aprendizaje proposicional, utiliza predicados de soporte

o   Graph data mining

  No hemos visto aún estos trabajos.

LAS CONCLUSIONES:

-          Lo que llaman relacional es realmente aprendizaje de primer orden.

-          Las propuestas que hemos estudiado toman inspiración de un algoritmo clásico de aprendizaje proposicional y extenderlo cambiando el language bias; es decir, en lugar de usar selectores de la forma A \theta v, utilizan predicados de soporte.

-          A falta de estudiar las propuestas de graph mining parece que nuestros puntos fuertes son los siguientes:

o   El resto de propuestas son ad-hoc:

  Toman como base una técnica existente, pero tienen que adaptarla para cambiar el language bias.

  Nosotros tomamos cualquier técnica existente sin hacer ninguna modificación en la misma y la extrapolamos para que haga aprendizaje relacional.

o   El resto de técnicas parten de una representación en lógica de primer orden con símbolos de función tan sólo.  No hacen una distinción explícita entre las características atributivas y relacionales.  ´

  Esto hace que el procedimiento de exploración tenga que trabajar con cada “relación” (predicado de soporte) de forma separada.  Es decir, es un procedimiento de exploración estilo FOIL.  En el caso de TILDE se usa look-ahead para mitigar un poco los efectos de la miopía, pero dicen que esto incrementa el coste computacional.

  Nosotros analizamos todas las características atributivas de golpe y de forma tan eficiente como el algoritmo de aprendizaje base sea capaz de hacerlo.  Si el clasificador que obtenemos no es suficientemente bueno, entonces usamos una cantidad de contexto no acotada para mejorarlo (el contexto son los datos a los que se llega a través de las características relacionales).

-          Si después de este análisis llegamos a la conclusión de que la técnica de ROLLER es original, tendríamos que pensar en venderla en un contexto más general:

o   La primera idea es en el contexto del Machine Learning

  El problema aquí es que para hacerlo tenemos que conseguir “vectorizar” los problemas de ILP clásicos.

  De momento tengo algunas ideas, pero no logro amarrarlas suficientemente bien.

o   Se me ocurre una idea mejor en el campo de la Web Semántica.

  Aquí los problemas se representan en Description Logic, que es una forma restringida de First-Order Logic en la que todos los predicados son unarios para indicar que algo es de algún tipo o binario, para representar propiedades de datos (attributive features) y de objeto (relational features)

  Es decir, nos lo han puesto sencillito, sencillito dado que la representación es justamente la que nosotros necesitamos.

  Y en el campo de la web semántica esto de aprender a partir de los datos es ciencia ficción, así que creo que podemos tener posibilidades de colarlo en una buena revista sobre el tema

TO DO

PATRICIA: como comentamos anoche, tú me envías el primer draft con la primera descripción de ROLLER en forma de enumerado de ideas, para repasarlo antes de atacar la escritura del texto.  Por favor, como tú vas a llevar la batuta en este artículo, mete ya una sección con estas conclusiones y las de mi mensaje anterior (incluida la repuesta que tú mandaste), de forma que no se nos olvide nada de lo que hemos estado discutiendo.

§  Usar MDL como medida de calidad de las reglas.   Podemos meter una columna final en la que decimos si las reglas producidas cumplen con el principio MDL o no.  Evidentemente una técnica que es muy eficiente y efectiva no es realmente tan buena si resulta que no cumple con el principio MDL.

§  Implementar MDL para cortar reglas que empiezan a ser muy complejas.  Esto ya está implementado en ROLLER 1, pero el problema es que nunca se activó este corte.  Además, si se hubiera activado, lo que ROLLER 1 habría hecho habría sido decir que no podía encontrar regla… pero realmente tendría que haber recuperado un savepoint y seguir por él.  El tema de los savepoints no llegué a implementarlo.

Está claro que MDL se puede usar 1) para medir la complejidad de las reglas, 2) para usarlo como criterio de poda. Si lo usamos como criterio de poda necesitamos implementar backtracking en la técnica.

§  Usar k lookahead.  Experimentalmente hemos llegado a la conclusión de que con explorar 3-4 nodos es más que suficiente para obtener una buena regla.  La cuestión es que podríamos realizar una exploración con k lookaheads; es decir, en lugar de explorar una expansión con una única característica relacional, explorar una expansión con k características relacionales a la vez.  Es decir, en lugar de explorar el conjunto de características relacionales R, exploramos R^k en cada paso del algoritmo.  No estudiamos por ejemplo el binding (node2, parent, node) y después (node3, next, node2), sino que exploramos todas las combinaciones de 2, 3, 4, … k predicados a la vez.

Ya te comenté que a mi esta idea me parece muy interesante y creo que puede vender porque el lookahead es una idea de la que se habla mucho en Machine Learning.

§  PATRICIA, TOÑI: En el pasado comentamos sobre usar funciones alternativas de scoring.  Incluso implementé Calcata para compararla con Information Content e Information Gain, pero los resultados eran sólo ligeramente mejores en tiempo.  El problema es que con ROLLER se exploran realmente muy pocos nodos.  En la experimentación que tenemos, se han explorado entre 2-3-4 nodos y 45-50 nodos… muy poquitos para que realmente haya una diferencia significativa de tiempo cuando se cambia la heurística de búsqueda (=funcion de scoring y de ganancia).  Por eso creo que de momento no merece la pena explorar esto mucho más.

De acuerdo. Yo el interés que le veo a las funciones de scoring es a que funcionen mejor de cara a la validación. Es decir, que a lo mejor las reglas que produce una función de scoring son más generales y a la hora de validar se comportan mejor. El "problema" aquí es, que nuestros resultados ya son bastante buenos.

§  PATRICIA: Comentamos buscar reglas de asociación con el objetivo de eliminar características redundantes.  Las reglas de asociación son de la forma p(a1, a2, …, an) -> q(am) y significan que si los atributos a1, a2, …, an cumplen el predicado p, entonces el atributo am cumple el predicado q.  Esto significa que am es un atributo redundante puesto que se puede calcular sobre la base de los anteriores. Pero esto no lo podemos realmente aplicar por el mismo problema que te comenté en relación a usar cualquier otra técnica de eliminación de atributos.  Suponte que en la primera expansión resulta que a1 = 2 & a2 > 3 => a3 < 7; así que quitamos el atributo a3.  ¿Pero por qué motivo?  La regla sólo dice que cuando a1 y a2 cumplen una determinada propiedad, a3 cumple otra diferente.  Podríamos pensar en eliminar tan sólo cuando las reglas de asociación sean en igualdad, por ejemplo a1 = 3 & a2 = 3 => a3 = 7, pero esto tiene dos problemas: a) este tipo de reglas no son muy habituales; b) ¿qué pasa si en la siguiente expansión a3 y parent\_a9 forman una buena pareja para clasificar o incluso dan lugar a otra regla de asociación?  Pues que ya no tenemos a3.  En cualquier caso, hay que experimentar antes de validar o refutar la hipótesis de que esta técnica puede funcionar, pero a priori le veo estos problemas, por lo que la dejo para el final.

Bueno, yo no me refería exactamente a eso. Porque según lo que comentas, si a1 = 2 & a2 > 3 --> a3 < 7, aquí a3 no sería redundante. Lo que yo digo es que para todos los pares <a1,a2> siempre se da un valor a3, de modo que a3 siempre puede expresarse en términos de a1, a2. Es decir, se debería de obtener un conjunto de reglas de asociación para todos los valores <a1,a2> no sólo para esos que mencionas. Si se encuentra una regla de asociación para cada par, entonces a3 es redundante. Y entonces eliminaría las características a1 y a 2, no a3. Por ejemplo:

a1                    a2                      a3

0                       0                       0

1                       3                       0

2                       3                       0

3                       3                       0

4                       4                       1

5                       4                       1

a1 < 4 & a2 < 4 --> a3 = 0

a1 >= 4 & a2 = 4 --> a3 = 1

Si encontramos reglas de asociación para todo el rango de valores, entonces la característica es redundante, en caso contrario pues no lo es. El problema de esto es que posiblemente no encontremos características redundantes.

§  RAFAEL: ¿Y qué decís si sólo usamos las características que tenían frecuencia mayor a 1 en FOIL del estudio experimental anterior? Es decir, quitar aquellas que no se usaron nunca o aquellas que no se usaron nunca en ROLLER para la versión del SAC. 🡪 creo que esta es buena para la próxima versión de ROLLER incluso para esta del ECML.

§  Usar un beam search para llevar k caminos de exploración en paralelo.

Ya estuvimos comentando esta idea y me transmitiste que no era tan fácil implementarlo. Yo creo que esta la deberíamos dejar para más adelante, si acaso. Pero no creo que vaya a mejorar enormemente los resultados en comparación con el esfuerzo.