TRABAJO FINAL
CARRERA DE GRADO DE INGENIERÍA EN SISTEMAS
FACULTAD DE CIENCIAS EXACTAS - U.N.C.P.B.A.
TANDIL, BUENOS AIRES, ARGENTINA

KAgent: Un agente de interfaz para el soporte al usuario de Data Mining

Autor:
Sr. Matías Alberto Nicoletti - *mnicolet@exa.unicen.edu.ar*

Director:
Mg. Héctor Oscar Nigro - *onigro@exa.unicen.edu.ar*

Resumen

Uno de los procesos de KDD (Knowledge Discovery in Databases), denominado Minería de Datos, puede ser conceptualizado como una secuencia de actividades, técnicas o algoritmos que tienen como propósito general la extracción de conocimiento interesante de un conjunto de datos triviales. La posibilidad de combinar aleatoriamente estas técnicas sin un adecuado entrenamiento o como mínimo, un esquema de asistencia, no permite aprovechar el potencial de las herramientas de Software disponibles. En este trabajo, se propone una solución con la utilización de: la tecnología de Agentes Inteligentes, Cadenas de Markov y el método AHP. En base a ello, se diseña un esquema de asistencia inteligente para usuarios que deseen practicar Minería de Datos, que permita recolectar y transferir el conocimiento de usuarios experimentados. Considerando la actual variedad de herramientas para KDD, se optó por el diseño de un framework con el fin de desarrollar una solución que sea aplicable a cierto rango de dichas herramientas. Finalmente, se implementó un esquema concreto para KNIME, el cual permitió llevar a cabo algunas pruebas para medir la utilidad efectiva del asistente.

1 Introducción

La Minería de Datos es uno de los campos de crecimiento más veloz en la industria de la computación. Una de sus principales fortalezas es su amplitud en cuanto a los posibles ámbitos donde puede ser aplicado. Algunos ejemplos de áreas que pueden mencionarse sobre su aplicación efectiva son: industria bancaria, telecomunicaciones, ventas, medicina, deportes y variedad de ingenierías [7]. En los últimos años, han surgido aplicaciones que proveen bibliotecas de técnicas y algoritmos para la ejecución de procesos de Minería de Datos. Como en cualquier otro grupo de herramientas de Software, es posible encontrar ejemplares distribuidos bajo licencia comercial, aunque en Minería de Datos existen herramientas excelentes y ampliamente utilizadas de licencia libre, como Weka[1], Orange[2] y KNIME[3].

A pesar de que dichas herramientas suponen facilitar la práctica de esta disciplina, definir y ejecutar procesos de Minería de Datos no resulta ser una tarea trivial. Por lo tanto, numerosas variables y decisiones a ser tenidas en cuenta se ven involucradas durante el proceso [7]. Las aplicaciones que actualmente soportan este análisis, empaquetan gran cantidad técnicas y algoritmos, expandiendo así las posibilidades de sus usuarios para explotar el descubrimiento de conocimiento. Solo como referencia, KNIME en sus versiones más actuales ya cuenta con alrededor de 500 operadores.

En ciertas ocasiones, la construcción de procesos se convierte en una tarea repetitiva, sobre todo cuando el objetivo de los mismos es similar. El registro y análisis de las combinaciones entre técnicas y de los procesos que se construyen en el tiempo podrían permitir la implementación de un esquema de asistencia basada en experiencias pasadas. Además, definir modelos de los criterios que condicionan la combinación de los diferentes operadores podría complementar dicho esquema. Sin embargo, las actuales herramientas desaprovechan esta oportunidad, que claramente podría beneficiar la generación de nuevos proyectos, facilitando el trabajo de los analistas.

Bajo este contexto, se propone la implementación de un esquema de asistencia basado en agentes inteligentes de interfaz y técnicas de Inteligencia Artificial que proporcione ayuda al usuario de las herramientas de Minería de Datos actualmente utilizadas. El agente podrá organizar la creación de procesos y asistir al usuario, inexperto o no, en diversos contextos de la construcción. Adicionalmente, se registrarán las combinaciones y procesos válidos que se construyan durante el tiempo, para posteriormente ejecutar un esquema de recomendación basado en Cadenas de Markov [4] como modelos predictivos. Llevar un registro de los procesos completos anteriormente creados permite analizar la similitud de éstos con los nuevos procesos que se construyan en la interfaz. Cuando el proceso actual presenta una alta similitud con alguno en la base de conocimiento, probablemente el usuario desee analizarlo y tomar ideas para su trabajo actual. Un algoritmo de comparación de procesos con tales características, es factible de implementarse mediante el uso del método AHP

[1] Weka Software http://www.cs.waikato.ac.nz/ml/weka/
[2] Orange Software. http://www.ailab.si/orange/
[3] Knime Software. http://www.knime.org/

[10]. Dada la compleja naturaleza de las comparaciones entre los procesos, AHP permite la definición de una flexible cantidad y granularidad de criterios para decidir el grado de similitud entre los mismos. Finalmente, la solución se extenderá a cualquier aplicación de Minería de Datos, mediante el diseño de un *framework* orientado a objetos adaptable a las peculiaridades y características específicas de cada herramienta.

El resto del trabajo se organiza de la siguiente forma. En la Sección 2 se explica porqué los usuarios de las herramientas de Minería de Datos deberían ser asistidos en sus tareas. En la Sección 3, se describe las principales características del esquema de asistencia planteado, mostrando el uso de Modelos de Markov, el método AHP, y el diseño de un *framework* independiente de la herramienta. En la Sección 4, se presenta una implementación del esquema para KNIME. En la Sección 5, se presentan algunos resultados obtenidos del proyecto. En la Sección 6, puede hallarse un análisis sobre los trabajos relacionados. Luego, en la Sección 7, se presentan las conclusiones del trabajo. Finalmente se anexan 2 apéndices para extender la información acerca de la herramienta desarrollada y 1 apéndice que muestra el cuestionario que se utilizó para su evaluación.

2 Asistencia en Minería de Datos

En la actualidad, las aplicaciones muestran un gran conjunto de herramientas y opciones para desarrollar las tareas para las que fueron construidas. Cada ítem de menú, cada comando escrito, cada icono, representan el acceso a una herramienta que brinda la aplicación al usuario. El usuario debe entonces, decidir que herramienta usará para cada propósito, y qué secuencia de pasos, utilizando las herramientas individuales, lograrán satisfacer la tarea a realizarse. El problema surge cuando el alcance de la aplicación se incrementa, produciendo un aumento en la cantidad de herramientas disponibles. Y efectivamente, es el problema que se ha venido dando en las aplicaciones de hoy en día; el usuario a veces resulta abrumado por la cantidad de posibilidades que le brinda una interfaz de usuario.

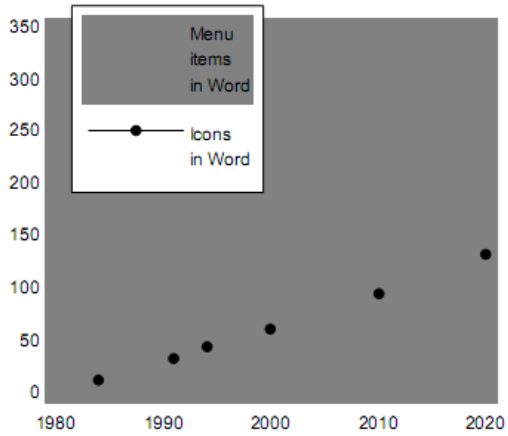

Figura 1. Crecimiento de la cantidad de iconos en Microsoft Word en el tiempo

A modo de ejemplo, se puede apreciar en la Figura 1 como a lo largo del tiempo, se ha dado un crecimiento de herramientas en la interfaz de una conocida aplicación, Microsoft Word. Bajo la suposición de que este crecimiento es lineal, dentro de dos o tres generaciones las personas deberán lidiar con un número absurdo y elevado de posibilidades. A partir de este modelo, se deduce que será necesario determinar formas de asistir al usuario ante interfaces complejas.

La Minería de Datos no es la excepción a esta regla. No solo se incrementan la cantidad de operadores disponibles por cada herramienta constantemente, sino que algunas permiten la incorporación de paquetes de operadores externos, como por ejemplo KNIME, donde pueden adquirirse extensiones con operadores de Weka, así como también operadores para Química, Matemática, entre otros. Pero la complejidad no solo se encuentra en la cantidad de posibilidades, sino que también debe tenerse en cuenta la correcta combinación de los operadores considerando las restricciones asociadas. De esta forma, es posible establecer una analogía entre la Minería de Datos y resolver un rompecabezas [7]. Las piezas individuales no suponen estructuras complejas, aunque al agruparse constituyen sistemas complejos y elaborados. Durante la construcción de estos sistemas, seguramente el analista puede frustrarse, comenzar a forzar las partes para que encajen entre sí, y eventualmente cansarse y desistir. Pero una vez que se comprende como utilizar dichas piezas, puede concluirse que no era un proceso tan complejo como parecía al comienzo. Con el tiempo, los usuarios aprenderán y se familiarizarán con el adecuado uso de las técnicas y algoritmos de Minería de Datos, aunque investigar formas de acelerar la curva de aprendizaje motiva el desarrollo de este proyecto.

La asistencia para Minería de Datos es realmente un campo de investigación poco explorado, sobre todo cuando se aplica específicamente a las herramientas. Uno de los motivos de esta situación podría ser el hecho de que las aplicaciones para soporte de Minería de Datos se han popularizado en los últimos tiempos. Por el contrario, la Minería de Datos ha sido utilizada ampliamente en el diseño de sistemas inteligentes, como en [19], [8], entre otros. Considerando esto, la asistencia a través de agentes inteligentes en el desarrollo de proyectos de Minería de Datos resulta en un enfoque novedoso y de notable valor práctico.

3 Visión general del asistente

Los agentes inteligentes de interfaz son entidades de software que tienen el objetivo de asistir a usuarios en sus tareas basadas en computadoras. A través del aprendizaje de sus hábitos de trabajo y de sus preferencias, estas entidades pueden mejorar la productividad de los usuarios e incluso reducir su carga de trabajo [18, 9]. La inteligencia en las aplicaciones puede identificarse sencillamente, cuando las mismas exhiben comportamiento similar al razonamiento humano. Lógicamente, un agente no puede alcanzar la inteligencia humana, ya que la misma es extremadamente compleja como para ser modelada en un software. Sin embargo, existen aproximaciones donde se consiguen emular algunas características, como por ejemplo el razonamiento, la inferencia o conocimientos específicos para ciertos dominios [21].

Es posible identificar 2 tipos de usuarios de las aplicaciones para Minería de Datos: i) los usuarios novatos, que aun no han comprendido los conceptos necesarios para llevar a cabo proyectos, y ii) los usuarios expertos, que manejan con facilidad los

conceptos pero que pueden no estar familiarizados con la aplicación a utilizarse. En el caso de i), con la ayuda de un asistente, los usuarios inexpertos pueden hacer frente tanto a las particularidades de las nuevas herramientas, como a la gran cantidad de operadores disponibles. En ii), el asistente puede ayudar al usuario experto a adaptarse a la herramienta con mayor velocidad.

El conocimiento que poseen los usuarios experimentados claramente le resulta de utilidad al novato, y comúnmente se ve desperdiciado. Dentro de dicho conocimiento se pueden identificar a) restricciones conceptuales entre los operadores (de carácter estático), b) restricciones contextuales entre los operadores (referidas al tipo de datos), c) combinaciones entre operadores comúnmente utilizados, y d) procesos completos que han demostrado ser útiles y podrían reutilizarse en un futuro.

En los casos a) y b), se diseñó un esquema flexible en cuanto a la definición de nuevas restricciones, soportando actualmente los dos tipos mencionados. Para a), el usuario experto puede definir la restricción entre dos operadores que conceptualmente no pueden combinarse, y que deben restringirse bajo cualquier contexto. En b), los usuarios expertos deben definir los tipos de datos (nominal, discreto, etc.) que un operador acepta, de modo que el asistente interprete en base al contexto si dicho operador puede ser utilizado.

En c), el asistente observa como los expertos componen los operadores y genera un modelo predictivo en base al historial de combinaciones. Posteriormente, sugiere la utilización del conjunto de operadores más comúnmente utilizados bajo el actual contexto de construcción, generando un ranking por relevancia. En este punto se analizaron 2 modelos predictivos: Cadenas de Markov y Redes de Bayes. El modelo de dependencias entre los operadores es sencillo, ya que se asume que el próximo operador solo depende del anterior. Por este motivo, se decidió utilizar las Cadenas Markovianas, dado que las Redes de Bayes suponen dominios con modelos de relaciones complejos y se estaría desaprovechando su potencial.

En d), los usuarios con experiencia desean almacenar los procesos que han sido exitosos, lo cual es una funcionalidad soportada por las herramientas actuales. Sin embargo, la información generada queda en archivos externos a la aplicación y difícilmente se reutilice. El asistente proveerá un mecanismo de importación / exportación propio que permita el análisis de los procesos y pueda generar recomendaciones a usuarios que se encuentren trabajando en proyectos similares. Además tomará en cuenta el *feedback* o retroalimentación de los usuarios para mejorar sus criterios de recomendación y poder ajustarse a los perfiles de los usuarios.

A pesar de que los agentes de interfaz pueden extraer conocimiento a través de la observación, en nuestro caso se requiere que los usuarios expertos brinden parte de su conocimiento de forma explícita en ciertos casos, como en a), b) y d). Esto se debe a que el esquema de asistencia propuesto requiere de cierto tipo de información que no puede inferirse del contexto de construcción. Luego, se debe hacer frente a uno de los mayores retos a la hora de desarrollar este tipo de agentes: determinar de qué forma el agente de interfaz podrá observar las acciones del usuario en la aplicación asistida. No siempre las aplicaciones se diseñan con el objetivo de ser extendidas o de permitir la observación de los eventos que acontecen. Como podremos observar en la Sección 3.3, en este caso se diseñó un esquema flexible para el monitoreo de eventos desde la interfaz, capaz de adaptarse a las distintas herramientas. De esta forma, el agente aprenderá de los usuarios expertos durante la etapa de entrenamiento y posteriormente

guiará a los analistas a través de una vista de información, que puede ser representada con una ventana aparte de la aplicación o con una vista interna dentro de la misma.

Las secciones siguientes explican algunos de los componentes principales del esquema de asistencia propuesto. En la Sección 3.1 se muestra como puede predecirse el siguiente operador a ser utilizado a través del uso de Cadenas Markovianas. En la Sección 3.2 se presenta el algoritmo de detección de similitudes entre procesos basado en AHP. Finalmente, en la Sección 3.3 se destaca la importancia de la definición de un *framework* para independizar la solución de la herramienta utilizada.

3.1 Cadenas de Markov como Modelos Predictivos

Los procesos estocásticos pueden definirse como procesos que toman valores dentro de un espacio de posibles estados, en función del tiempo. Se deduce que la mayoría de los procesos que ocurren cotidianamente puede considerarse estocásticos. En particular, las Cadenas de Markov o Procesos Markovianos, son un tipo especial de procesos estocásticos que poseen la Propiedad de Markov, donde el estado actual del proceso condiciona con cierta probabilidad el valor del estado siguiente. Más formalmente, dado un conjunto de posibles estados discreto, $S=\{S_0, S_1, S_2, ...S_n\}$ y un conjunto de instantes de tiempo discretos, $T=\{T_0, T_1, T_2, ...T_n\}$, una Cadena de Markov es un proceso estocástico donde el valor que toma en el tiempo, $P(T_{i+1})$ esta condicionado por $P(T_i)$. Por lo tanto, quedan completamente caracterizadas por las probabilidades condicionales P_{ij} para todo *i*,*j* en S, que representa las probabilidad de transición desde el estado *i* al *j* o la probabilidad de ocurrencia del estado *j* dado el *i*. Pueden representarse mediante grafos o matrices de transición, donde los posibles estados se reflejan en los nodos o filas/columnas y las transiciones en los arcos o intersecciones, respectivamente.

Los procesos Markovianos tienen una amplia gama de aplicaciones, entre las que se encuentran aplicaciones en Internet como el *PageRank* de Google, estadística y simulación como el Método Monte Carlo basado en Cadenas de Markov [5], en la biología aplicado a la dinámica de las poblaciones [1], en los juegos de azar, en la predicción del clima, entre otros. En general, pueden aplicarse a cualquier dominio en el que se pueda identificar: un conjunto de estados del sistema, la definición de transición entre estados, y una ley de probabilidad condicional, que defina la probabilidad del nuevo estado en función de los anteriores.

Bajo este marco, podemos notar que un proceso de Minería de Datos puede ser conceptualizado como una secuencia de operadores, cada uno perteneciente a una de las etapas del proceso de KDD (recolección, pre proceso, extracción, etc.). La construcción de procesos tiende a convertirse en el diseño de *workflows*, donde se generan combinaciones entre dichos operadores. Las combinaciones comúnmente suelen repetirse, generando secuencias válidas y potencialmente útiles. Por lo tanto, en base a esta información que actualmente esta siendo desperdiciada, es posible determinar o inferir que tipo de técnica podría ser factible de utilizar dado el estado del proceso actual. Esta situación es similar a la que se da en [4], en donde bajo la suposición de que las personas muestran regularidades en sus acciones diarias, mediante un método de predicción simple basado en Cadenas de Markov se puede predecir el siguiente comando de consola ejecutado en un entorno UNIX. Lógicamente, para ello el modelo requiere de un tiempo de entrenamiento para

adaptarse a sus posibles usuarios. No existen métodos para determinar el tiempo adecuado, pero en el estudio realizado se demostraron resultados coherentes con un entrenamiento de 168.000 ocurrencias. El modelo resultante se denominó IPAM (*Incremental Probabilistic Action Modeling*), y surge por la necesidad de proveer un mecanismo preciso, eficiente, adaptativo y que no requiera almacenar el historial de comandos ejecutados.

El modelo IPAM se implementa de forma sencilla con una matriz de transiciones de NxN, donde N es la cantidad total de comandos distintos utilizados. La posición M_{ij} representa la probabilidad del comando *j* dado que el actual es *i* o $P_{(j/i)}$. La matriz fue implementada con una estructura dinámica de modo que sus dimensiones fueran aumentando a medida que se ejecutaban nuevos comandos. Para generar recomendaciones, dado que se había ejecutado el comando X, en la fila correspondiente a X en la matriz se realizaba un ordenamiento para poder recuperar el próximo posible comando con mayor probabilidad de ocurrencia. Lógicamente, la predicción puede extenderse a un conjunto de N comandos más probables en vez de solamente el mayor. Finalmente, el algoritmo utiliza un factor conocido como *ALPHA*, que permite determinar el grado de importancia de las ocurrencias más recientes respecto de las antiguas. Este factor esta comprendido en el rango [0,1], y dada la ocurrencia del comando *j* dado que el anterior fue *i*, se multiplican los coeficientes de la fila correspondiente a *i* por *ALPHA* (simulando el olvido de ocurrencias) y la casilla correspondiente a $P_{(j/i)}$ se le adiciona (1- *ALPHA*).

Aplicado en el contexto de predicción de próximas técnicas, la matriz de transición que define el modelo de Markov, puede apreciarse en la Figura 2. Allí los operadores se corresponden con las filas y columnas de la misma, y las probabilidades de las celdas se verán modificarán a medida que nuevas ocurrencias se observen en la interfaz. Adicionalmente, el pseudo-código para las funciones de entrenamiento y de predicción se adjunta a continuación. En cuanto al valor de *ALPHA*, fue establecido en 0,8 considerando los resultados obtenidos el estudio en [4].

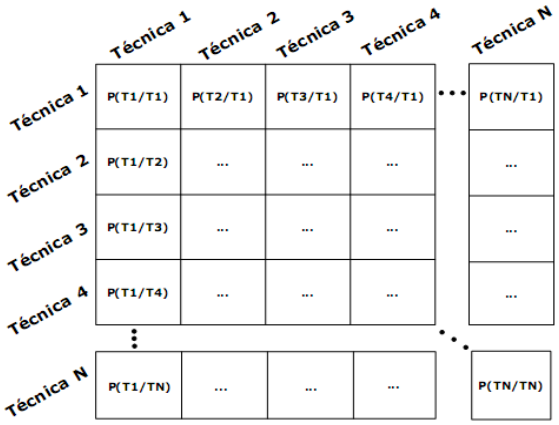

Figura 2. Matriz de transición para el modelo de predicción

```
predecirSiguientes (String fuente, int cantidad) : String[] {
    Obtener copia de la fila Matriz[fuente];
    Ordenar descendentemente la copia de la fila.
    Retornar las primeras (cantidad) técnicas;
}

entrenar (String fuente, String destino) : void {
    Si (no existe fuente)
        Agregar fuente a la matriz;
    Si (no existe destino)
        Agregar destino a la matriz;
    Ocurrencias++;
    Para cada técnica registrada {
        Matriz[fuente,técnica]=Matriz[fuente, técnica]* ALPHA;
    }
    Matriz[fuente,destino]=Matriz[fuente destino] + (1 – ALPHA);
}
```

3.2 Algoritmo de comparación basado en AHP

A medida que la herramienta de Minería de Datos es utilizada, se generan procesos que llevan al éxito del proyecto de análisis llevado a cabo. Estos procesos contienen información valiosa sobre combinaciones válidas entre operadores, y pueden ser aprovechadas en la construcción de nuevos procesos. Una vez concluido el proyecto y comprobada su utilidad, los usuarios tienden a almacenar el proceso mediante el mecanismo de exportación comúnmente provisto en las aplicaciones, aunque en realidad dicha información esta siendo desperdiciada. Existe la oportunidad de que el agente propuesto utilice la base de datos de procesos que se genera implícitamente, para asistir a futuros usuarios de la herramienta. El esquema planteado debería ser capaz de procesar los proyectos almacenados y recomendar aquel/los similares al que actualmente se construye. Esto puede resultarle útil al actual analista, ya que podría analizar por su propia cuenta el proceso similar e identificar posibles ideas y soluciones que beneficien su trabajo. Resulta claro que determinar el proceso más similar de una base de datos o, más concretamente, comparar procesos de Minería de Datos, puede resultar complejo. Más aun, el algoritmo empleado puede requerir de ajustes a medida que se prueba con ejemplos reales. Por lo tanto, a continuación se propone analizar una simple pero potencialmente útil técnica para mejorar el esquema de asistencia planteado.

El AHP (*Analytic Hierarchy Process*) [10] es una técnica comúnmente utilizada en la toma de decisiones complejas, que involucran múltiples criterios a tenerse en cuenta. Este método no sólo se limita en ámbitos informáticos como en [11], sino que se extiende a cualquier dominio en donde sea necesario tomar decisiones complejas, como en la medicina [20] o en los negocios [3]. Para aplicar este método en cualquier contexto, deben identificarte los tres elementos principales, los cuales pueden verse relacionados en la Figura 3:

Objetivo: es la meta principal o propósito para el cual se ejecuta el análisis. En el contexto de este problema sería determinar el proceso almacenado con mayor similitud al que actualmente se este construyendo.
Criterios: representan una separación lógica para alcanzar ese objetivo principal. En AHP se acostumbra a definir una estructura de jerárquica de criterios que permiten desglosar un problema considerable en pequeños sub problemas. Aplicando el método a este proyecto, se debería utilizar una jerarquía de criterios para determinar la similitud entre procesos. Como bien se definió, comparar procesos de Minería de Datos no es una tarea simple y una separación en sub criterios beneficia la construcción de tal algoritmo. A cada criterio y sub criterios se los pondera con un peso, modelando la importancia de cada uno y teniendo como restricción que la suma de los pesos de los sub criterios de un criterio padre debe ser igual a 1.
Alternativas: representan todas las posibles respuestas al objetivo. La pregunta principal en el problema sería: ¿Qué proceso almacenado es el más semejante al que se encuentra en construcción? Las alternativas son los procesos en la base de datos y son solamente evaluadas de forma concreta en los criterios inferiores u hojas del árbol. En los criterios intermedios la evaluación es simplemente aplicar el peso de cada criterio a las evaluaciones más concretas provenientes de los sub criterios, por lo que se resume en una multiplicación de probabilidades.

Figura 3. Estructura genérica de criterios para AHP

En el contexto del trabajo, el método AHP es utilizado para el análisis de similitudes entre procesos que se almacenen en la base de conocimiento. Dicho análisis se desencadena luego de un evento que modifique significativamente el proceso, como por ejemplo la creación de una nueva conexión entre operadores. Ante un alto grado de similitud entre el proceso actualmente en construcción con alguno de los procesos almacenado, se le recomienda al usuario la revisión de dicho proceso. En una primera aproximación al uso de este método, se identificaron los criterios que pueden apreciarse en la Figura 4.

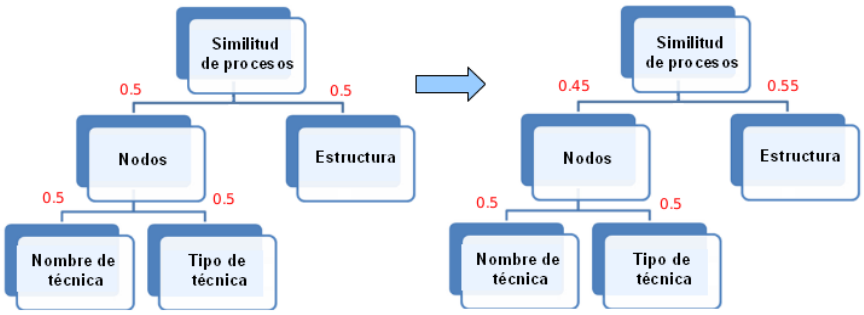

Figura 4. Proceso de ajuste de los criterios para la similitud de procesos

Desde un principio, para definir los pesos de cada criterio se consideran probabilidades equivalentes en cada nivel, obteniendo una configuración estándar. Pero es lógico que este esquema no se adapte a todas las situaciones y a los posibles intereses que posea cada usuario. Por lo tanto, se brinda la posibilidad de evaluar las recomendaciones para que el algoritmo sea ajustado por el agente con el fin de mejorar la búsqueda de similitudes. Cuando el usuario recibe una recomendación, puede aprobarla o rechazarla, de acuerdo a si resultó útil o por lo menos, si los procesos presentaban similitudes interesantes. En base a la evaluación, se propuso que el agente adopte el siguiente comportamiento: 1) en caso de evaluación positiva, como la recomendación fue exitosa, se determina el criterio que obtuvo la mejor evaluación y se incrementa la importancia del mismo en el nivel al que pertenece; 2) en caso de evaluación negativa, se decrementa el peso del criterio mejor evaluado con el fin de corregir la recomendación errónea. Esto permite que el grado de similitud final que retorna AHP sea mayor o menor respectivamente. En la Figura 4 puede observarse como se modifica el algoritmo considerando que el criterio Estructura fue el de mayor peso.

3.3 Framework independiente de la herramienta

Considerando que no existe una única herramienta que satisfaga las necesidades y preferencias de todos los analistas, resulta beneficioso para este trabajo el diseño de un *framework* de objetos Java que permita adaptar el esquema de asistencia a las diferentes aplicaciones, como KNIME, Orange y Weka. Aunque los *frameworks* son estructuras abstractas, poseen un alcance limitado en cuanto a sus dominios de aplicación. En este caso, el propósito general o dominio de dicho *framework* sería la construcción de agentes de interfaz para herramientas de Minería de Datos, por lo que se generarían diferentes implementaciones concretas para cada herramienta. De esta forma, es posible abstraer el comportamiento en común y promover la re usabilidad tanto de diseño como de código fuente Java.

Durante la creación de un *framework*, es importante definir los puntos de modificabilidad o *hot spots* que le provean a sus implementaciones la posibilidad de definir sus comportamientos específicos. Ejemplos de *hot spots* pueden encontrarse en la implementación de una interfaz Java, la definición de un método abstracto, la

composición de objetos, la definición de archivos XML, etc. En el contexto del trabajo, deben definirse los puntos que varían según la herramienta donde se utilice el agente, entre los que encontramos: a) la definición de una vista de información para mostrar alertas y recomendaciones, b) la persistencia de los datos necesarios para los modelos (para la cual se provee una implementación por defecto en archivos XML), c) la definición de nuevos tipos de restricciones o conocimientos relevantes para la asistencia, d) la forma de extraer los tipos de datos manejados en el proceso, e) el monitoreo de los eventos desde la interfaz de la aplicación, entre otros.

Una vez definido el *framework* y sus puntos de extensión, se procedió a extenderlo de modo de alcanzar el comportamiento concreto que caracterice a la herramienta KNIME, generando una asistente funcional denominado *KAgent*. De esta forma, se proporciona una guía de cómo el *framework* puede ser extendido y se obtiene una aplicación concreta sobre la cual evaluar los utilidad del esquema plateado. En la siguiente Sección profundizaremos sobre *KAgent*.

4 KAgent: Un asistente inteligente para usuarios de KNIME

KNIME es una herramienta basada en *workflows* que posee un repositorio con alrededor de 500 operadores y se distribuye como un plug-in para la plataforma Eclipse [13]. De la misma forma, para el desarrollo de *KAgent* se aprovecharon las facilidades que proporciona dicha plataforma, influenciando el diseño de nuestra herramienta. En el Apéndice A puede verse en detalle la arquitectura definida. En la Figura 5 se presenta la interfaz de *KAgent* y los elementos visuales que se agregan a KNIME.

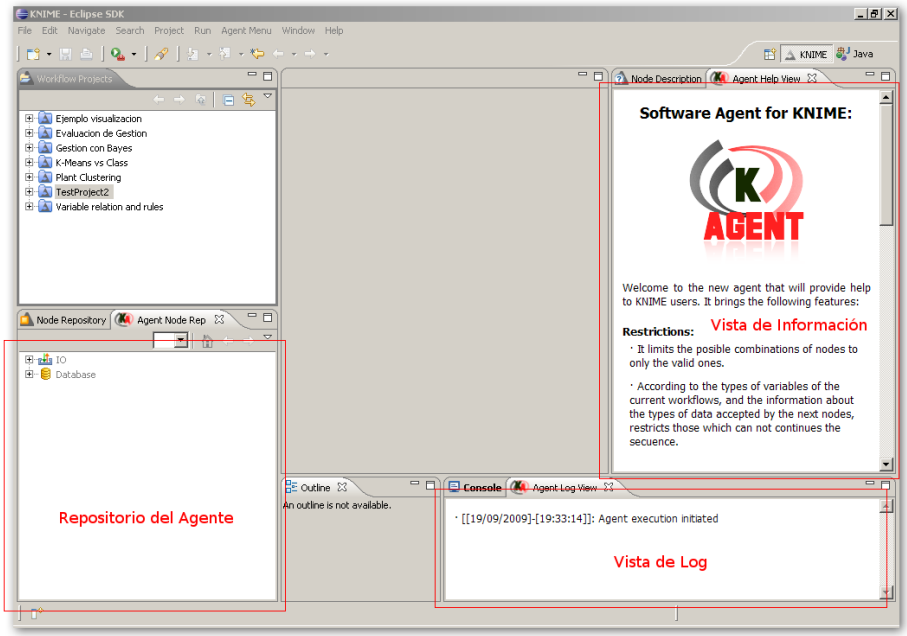

Figura 5. Interfaz de *KAgent*

Vista de información: componente principal del esquema que le permite al usuario visualizar las notificaciones generadas, aunque podrían ser obviadas por el mismo ya que no interfieren directamente con su trabajo. Puede complementarse con la vista de ayuda de KNIME, con información estática sobre los operadores.
Vista de log: provee mayor visibilidad a las acciones de agente, lo cual permite evaluar su comportamiento durante la etapa de desarrollo de la herramienta.
Vista de repositorio del Agente: basada en el repositorio original, genera los filtros adecuados y resalta aquellos operadores que tengan asociados conocimientos, como restricciones o recomendaciones.

Recordando los conocimientos que resultan interesantes en el esquema de asistencia, a continuación se describe como se ven reflejados en *KAgent*:
a) En cuanto a las restricciones conceptuales, el usuario experto selecciona los operadores desde el repositorio y establece una restricción a través del menú del asistente (Figura 6). Adicionalmente agrega una explicación acerca de la incompatibilidad.
b) En las restricciones contextuales, el usuario experto selecciona los operadores del repositorio y define los tipos de datos aceptados como entrada para cada uno. Posteriormente, el agente analiza el tipo de datos resultante de un operador en uso y le indica al usuario que operadores puede utilizar a continuación.
c) En las combinaciones de operadores, cuando se selecciona un operador en el *workflow* que ya forma parte del proceso, el asistente le indica al usuario cuales son los operadores recomendados para continuar el proceso utilizando Markov. Se determinó mostrar los 3 operadores más comúnmente utilizados, ordenados por su relevancia en el pasado. Por otro lado, si el usuario utiliza efectivamente un operador, recomendado o no, el asistente entrena el modelo con una ocurrencia.
d) Cuando un proyecto finaliza con éxito, comúnmente se almacena en KNIME a través del exportador de procesos en archivos ZIP. En este caso, *KAgent* reutiliza el mecanismo de exportación, pero almacena los procesos en un repositorio indexado, generando un descriptor con información relevante para el algoritmo AHP. Luego, cuando un usuario se encuentra en plena construcción y realiza una modificación significativa en el *workflow*, se ejecuta el algoritmo en búsqueda del proceso más similar. Solo en el caso de hallar alguno con una similitud notable con respecto al resto de las alternativas, se genera una recomendación. Luego, el usuario puede indicar si la recomendación fue útil o no, ajustando la importancia de los criterios involucrados.

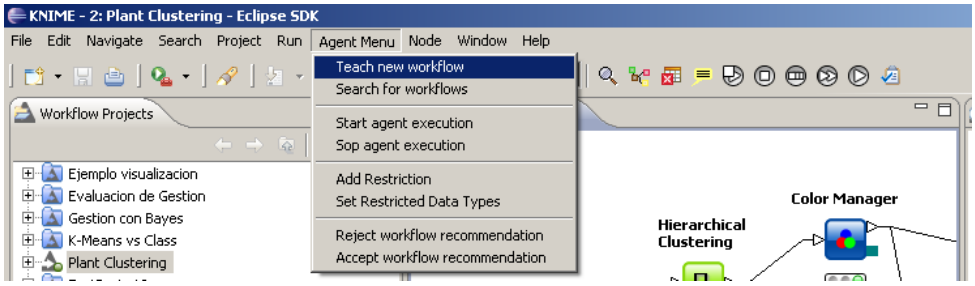

Figura 6. Menú del asistente en KNIME

En el Apéndice B puede observarse un caso práctico de uso del asistente para KNIME.

5 Resultados

La herramienta fue probada por los autores verificando su correcto funcionamiento y evaluando su potencial utilidad. El modelo de Markov se comportó de forma adecuada, recomendando los operadores que comúnmente se utilizaron y priorizando las ocurrencias más recientes. Adicionalmente, se generaron procesos que se asemejaban a los 10 ejemplos que se introdujeron en la base de datos. El algoritmo basado en AHP respondió correctamente en el 73% de los 15 casos, aunque en algunos recomendaba procesos no tan similares. El mecanismo de *feedback* permitió que el asistente corrija sus recomendaciones en el 27% restante de los casos de prueba, evitando así convertirse en una molestia en lugar de una ayuda.

En este momento estamos realizando pruebas sobre estudiantes de Ingeniería en Sistemas que toman el curso optativo *Data Mining*, de forma de poder evaluar las mejoras introducidas por el uso de las herramientas con y sin el asistente, sobre usuarios novatos. Previamente, *KAgent* fue entrenada por los autores del proyecto durante la construcción de 10 procesos válidos que han sido usados en proyectos académicos. A los alumnos se les entrega un trabajo práctico para la aprobación de la materia, en el cual deben aplicar los conocimientos adquiridos durante el curso para resolver un problema de Minería de Datos. Adicionalmente, como requisito los alumnos deben utilizar KNIME para la resolución. Luego de un lapso de 2 semanas, se les distribuye una versión de *KAgent* para continuar y finalizar el práctico. Finalmente se efectúa una encuesta (referirse al Apéndice C) para determinar si *KAgent* ayudó a los alumnos con su labor en relación con la versión original de KNIME. Las primeras encuestas recibidas indican resultados favorables para nuestro enfoque, aunque la evaluación aun se encuentra en proceso.

6 Trabajo relacionado

Los agentes inteligentes han sido ampliamente utilizados en diversas áreas, sobre todo cuando se requiere manejar grandes cantidades de información o ejecutar tareas repetitivas, como filtrado de información, búsqueda y navegación en la Web [6, 14], manejo y consultas en bases de datos [16], manejo de e-mail, turismos [17], planificación de reuniones y comercio electrónico [15]. A pesar de ello, el uso de agentes para asistir el desarrollo de proyectos de Minería de Datos es un campo realmente poco explorado.

Actualmente se ha estado investigando en el uso de ontologías para la generación de procesos en Minería de Datos [12]. Uno de los proyectos que podemos encontrar es e-LICO[1], en donde existe el sub proyecto DMOP (Ontologías en Minería de Datos para la Optimización de Flujos de Trabajo). Entre las bases de este proyecto, se encuentra la investigación de [2], en donde se introduce el concepto de Asistentes Inteligentes de Descubrimiento (IDA) que permiten la generación de procesos válidos

[1] e-Laboratory for Interdisciplinary Collaborative Data Mining http://www.e-lico.eu/

y su posterior priorización, para ayudar a los usuarios de Minería de Datos en la búsqueda de conocimiento. A través de la correcta definición de una ontología, incluyendo los operadores y las condiciones bajo las cuales pueden emplearse, puede explorarse el espacio de posibles procesos válidos para su posterior implementación. De la misma forma, un IDA se convierte en una herramienta para la transferencia de conocimiento entre los analistas, aunque para ello es necesario que se hagan contribuciones a la estructura de la ontología. Esto implica que los expertos en Minería de Datos deben comprender el concepto de ontología y analizar la forma en la que se debe ingresar nuevo conocimiento en el modelo.

Tanto en el enfoque explicado en el párrafo anterior como en el propuesto en este trabajo es posible modelar el conocimiento de expertos en el área, por ejemplo las restricciones a tenerse en cuenta para la generación de procesos válidos. Adicionalmente, en nuestra propuesta, es posible extraer mayor cantidad de conocimiento desde los expertos, como lo son las combinaciones más comúnmente usadas entre operadores de Minería de Datos. De esta manera, el entrenamiento de los novatos se realiza en un esquema paso a paso, de forma que en cada situación o punto en la construcción de procesos pueda aprenderse qué operadores es posible utilizar y cuáles no, justificando las decisiones del asistente en cada caso. Recapitulando, en ambas propuestas el propósito general difiere y la transferencia de conocimiento se efectúa en formas distintas.

7 Conclusiones

En este trabajo, hemos presentado un esquema de asistencia inteligente para usuarios de herramientas de Minería de Datos, basado en la captura y modelado del conocimiento de usuarios expertos, que comúnmente se desperdicia. El esquema se compone de un agente inteligente de interfaz que utiliza Cadenas de Markov y un algoritmo basado en AHP para generar notificaciones y avisos a los usuarios, guiándolos en la elaboración de procesos de análisis. A través del diseño de un *framework*, el asistente puede adaptarse a las distintas herramientas disponibles. En base a los resultados preliminares realizados sobre *KAgent* para KNIME, se supone que el asistente consiguiría simplificar la curva de aprendizaje de los usuarios novatos y que los modelos se comportan de la forma esperada, aunque más experimentación confirmaría esta situación.

Como trabajo futuro, podrían analizarse qué otras formas de conocimiento resultan útiles para los usuarios novatos y con qué modelos podrían representarse en el esquema de asistencia. Por otro lado, podrían analizarse nuevas dependencias entre los operadores y demás variables, como por ejemplo si el tipo de la tarea de minería (clasificación, agrupamiento, etc.) condiciona el próximo operador. Además, actualmente el conocimiento es almacenado de forma local y resulta difícil de compartir entre comunidades de usuarios, por lo que diseñar un esquema distribuido de almacenamiento de los modelos facilitaría la transferencia de conocimiento entre los distintos grupos de analistas y enriquecería la calidad de dicho conocimiento.

Referencias

1. Balzter H. Markov chain models for vegetation dynamics. Ecological Modelling. Volume 126, Issues 2-3, 28 February 2000, Pages 139-154.
2. Bernstein A., Provost F., Hill S. Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification. IEEE Transactions on Knowledge and Data Engineering. Volume 17, Issue 4 (April 2005).
3. Chena M., Wang S. The critical factors of success for information service industry in developing international market: Using analytic hierarchy process (AHP) approach. Expert Systems with Applications. Volume 37, Issue 1, January 2010, Pages 694-704.
4. Davison B., HirshH.. Predicting Sequences of User Actions. AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time Series Analysis, Madison, WI, July 27, 1998.
5. Diaconis P. The Markov chain Monte Carlo revolution. Bull. Amer. Math. Soc. Volume 46, pp.179-205. 2009.
6. Godoy D., Schiaffino S., Amandi A.. Interface agents personalizing Web-based tasks. Special Issue on Intelligent Agents and Data Mining for Cognitive Systems, Cognitive Systems Research Journal, Vol. 5, Issue 3. pp. 207-222. 2004.
7. Kantardzic M.. Data Mining: Concepts, Models, Methods, and Algorithms. John Wiley & Sons. ISBN: 0471228524. Chapter 1. 2003.
8. Kim Y., Street W. An intelligent system for customer targeting: a data mining approach. Decision Support Systems, Volume 37, Issue 2, May 2004, Pages 215-228.
9. Lieberman H., Selker T. Agents for the User Interface. Handbook of Agent Technology, Jeffrey Bradshaw, ed., MIT Press, 2003.
10. Mochol M., Jentzsch A., Euzenat J. Applying an Analytic Method for Matching Approach Selection. Freie Universitat Berlin, Institut fur Informatik. 2006.
11. Ngai T., Chan C. Evaluation of knowledge management tools using AHP. Expert Systems with Applications, Volume 29, Issue 4, November 2005, Pages 889-899.
12. Nigro H.; González Císaro S., Xodo D. Data Mining with Ontologies: Implementations, Findings and Frameworks. Idea Group Reference ISBN-13: 978-1599046181. 2008.
13. Object Technology International Inc. Eclipse Platform Technical Overview. Technical report. 2003. Available at http://www.eclipse.org/whitepapers/eclipse-overview.pdf .
14. Pazzani M., Muramatsu J., Billsus D. Syskill. Syskill & Webert: Identifying interesting web sites. Technical report Department of Information and Computer Science, University of California, Irvine. 1999.
15. Schafer J., Konstan J., Riedl J. Electronic commerce recommender applications. Journal of Data Mining and Knowledge Discovery, pp.115-152. 2000.
16. Schiaffino S., Amandi A. An Interface Agent Approach to Personalize Users' Interaction with Databases. Journal of Intelligent Information Systems. Springer, 25(3), pp. 251-273. 2005.
17. Schiaffino S., Amandi A. Building an expert travel agent as a software agent. Expert Systems with Applications, Volume 36, Issue 2, Part 1, March 2009, Pages 1291-1299. Elsevier, 2009.

18. Schiaffino S., Amandi A. User - interface agent interaction: personalization issues. I. J. of Human-Computer Studies. Volume 60 , Issue 1 (January 2004). Pages: 129-148.
19. Symeonidis A., Chatzidimitriou K., Athanasiadis I., Mitkas P. Data mining for agent reasoning: A synergy for training intelligent agents. Engineering Applications of Artificial Intelligence, Volume 20, Issue 8, December 2007, Pages 1097-1111.
20. Tsuen-Ho Hsua, Frank F.C. Pan. Aplication of Monte Carlo AHP in ranking dental quality attributes. Expert Systems with Applications. Volume 36, Issue 2, Part 1, March 2009, Pages 2310-2316.
21. Wooldridge, Jennings N. Intelligent agents: Theory and practice. Knowledge Engineering Review. Volume 10, pp. 115-152. 1995.

Apéndice A: Vista general de la arquitectura de KAgent

Analizando la arquitectura de la solución implementada para el caso particular de la herramienta KNIME, debemos considerar que se distribuye como un *plug-in* para Eclipse. La arquitectura de Eclipse [13] esta basada en el concepto de *plug-ins*, que son módulos que agregan funcionalidad al núcleo del IDE, de forma similar a como funciona un micro-kernel de un sistema operativo moderno. Por lo tanto, puede considerarse a Eclipse como un extenso módulo, rico en funcionalidad, que provee servicios para la captura de eventos en su interfaz y la comunicación con el usuario a través de vistas y editores. KNIME utiliza esos servicios e interfaces para proveer al usuario las facilidades de desarrollo de proyectos de Minería de Datos. Aunque no se encuentra dentro del alcance de este proyecto analizar la arquitectura de KNIME, es importante destacar un componente interno encargado de la importación/exportación de procesos, que será reutilizado por la solución. Recordando el funcionamiento de KNIME, los procesos se materializan en *workflows* de técnicas/algoritmos, por lo que el componente de importación/exportación permite transformar los proyectos en un formato portable, como por ejemplo archivos ZIP.

Con el objetivo de proveer una herramienta de asistencia que presente un diseño modular, facilitando su modificabilidad, separando la interfaz de la funcionalidad, la arquitectura se basa en un estilo Modelo-Vista-Controlador. En la Figura 7 se presenta un diagrama de componentes y conectores, donde pueden apreciarse los siguientes módulos:

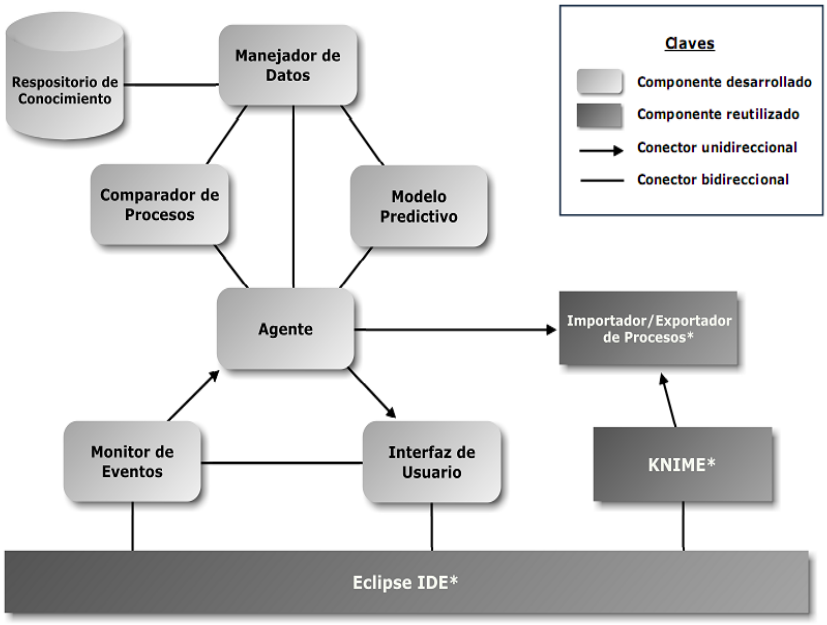

Figura 7. Arquitectura de la solución para KNIME

Interfaz de Usuario: se encuentra representado por un conjunto de vistas y editores de Eclipse, que permitirán informar al usuario el estado interno, así como también todo tipo de notificaciones necesarias para comunicar las decisiones del Agente. Adicionalmente se incluirán vistas para adicionales para el manejo de procesos almacenados.

Monitor de Eventos: debe interpretar los eventos producidos por el usuario mientras trabaja con la interfaz de KNIME (o la interfaz de Eclipse de forma más general), para determinar que acciones debe ejecutar el agente como respuesta.

Agente: representa el núcleo del sistema, que será el componente encargado de mantener la lógica que simule la inteligencia artificial. Siendo informado de los eventos del usuario, el Agente ejecutará acciones en respuesta a cada evento. Estas acciones modifican el estado interno del sistema, por lo cual eventualmente deba notificarse al usuario ante situaciones importantes a través de la Interfaz de Usuario.

Modelo Predictivo: encargado la construcción del modelo de inferencia en base a las combinaciones de técnicas / algoritmos históricas y de la generación de recomendaciones en base a ello.

Comparador de Procesos: define la lógica necesaria para comparar procesos almacenados en el Repositorio con el proceso que el usuario construye en busca del mayor grado de similitud.

Manejador de Datos: encargado de administrar el Repositorio, de forma de independizar al resto del sistema de la representación y método de persistencia de los datos. Entre los datos a ser administrados se encuentran las restricciones, el modelo predictivo, el registro de procesos exitosos, la estructura y configuración del algoritmo de comparación, entre otros posibles datos de configuración de la herramienta.

Repositorio de Conocimiento: debe proveer persistencia a los datos mencionados, pudiendo ser solo accedido a través del Manejador de Datos. Por motivos de simplicidad y dado que la cantidad de información a registrarse no es extensa, se optó por un almacenamiento en archivos XML.

Apéndice B: Ejemplo práctico en KAgent

Se muestra a continuación un caso práctico de uso del asistente, donde se pueden observar las funciones principales de la herramienta. En KNIME la mayoría de los operadores suponen una etapa de configuración previo a que puedan se ejecutados. En dicha etapa, pueden establecerse ciertos parámetros de los algoritmos como coeficientes o fuentes de datos para trabajar. Los operadores cumplen el rol de procesadores de información, donde cada uno posee puertos de entrada (menos aquellos que permiten el ingreso de datos en el *workflow*) y puertos de salida, donde puede obtenerse la información resultante como datos o modelos creados (menos aquellos de visualización y salida de datos). *KAgent* estructura la construcción resultando en un proceso más organizado. Para ello, cuando no se seleccionan operadores utilizados en el *workflow*, solo se permite el uso de operadores de entrada de datos que comiencen el flujo de información en el proceso. Para poder obtener la información resultante de la salida, el operador debe configurarse y ejecutarse, por lo que se le sugiere al usuario a que realice dichas acciones para continuar con la asistencia del agente (Figura 8).

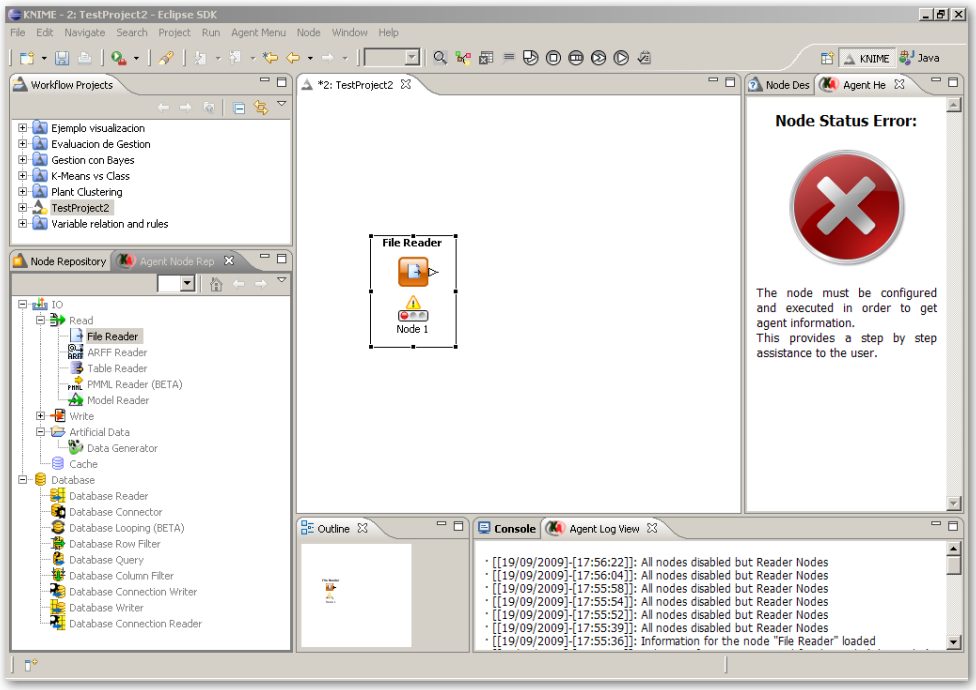

Figura 8. Filtro inicial con operadores de entrada y error por nodo no configurado

Continuando con el ejemplo, se utilizó un operador de entrada de datos desde un archivo. A continuación se lo configura para que lea datos de una conocida base en Inteligencia Artificial llamada *IrisDataSet*, la cual contiene información sobre un tipo de planta. Al estar ejecutado, el agente puede obtener información sobre la salida y

efectuar su análisis. En primera medida, se informa a través de la vista correspondiente que existe información acerca del operador en cuestión: se enumeran restricciones, recomendaciones para continuar el proceso y la información sobre las variables de salida (Figura 9).

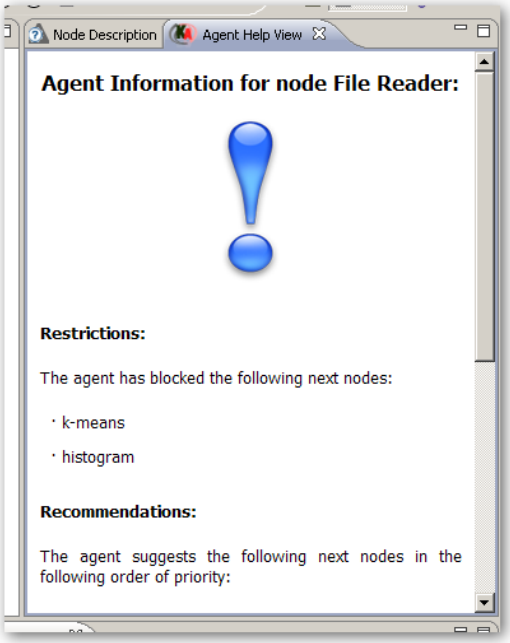

Figura 9. Vista de información sobre el nodo ejecutado

Automáticamente, en el repositorio del Agente se pueden apreciar las técnicas recomendadas y restringidas resaltadas con verde y rojo respectivamente (Figura 10).

En el caso de seleccionar una técnica recomendada, se observa en la vista de información el nivel con el que se recomienda la misma, dependiendo de la probabilidad dada por el modelo de Markov. Al utilizar cualquiera de las técnicas no prohibidas, se entrena el modelo de predicción con una nueva ocurrencia (Figura 11).

En el caso de seleccionar una técnica restringida, se observa en la vista de información una explicación del motivo que lleva a la restricción. Adicionalmente, el nodo no puede ser arrastrado desde el repositorio del agente, impidiendo una combinación inválida (Figura 12).

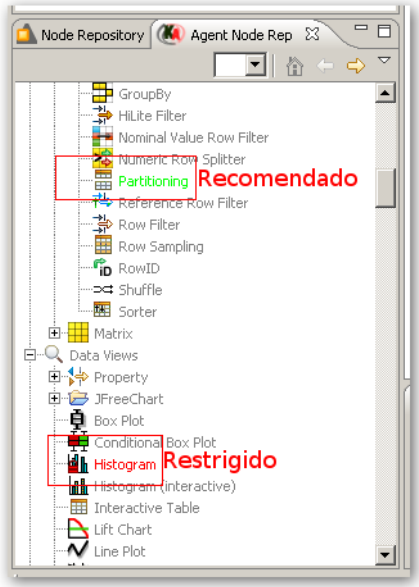

Figura 10. Repositorio reflejando la información del agente sobre el nodo

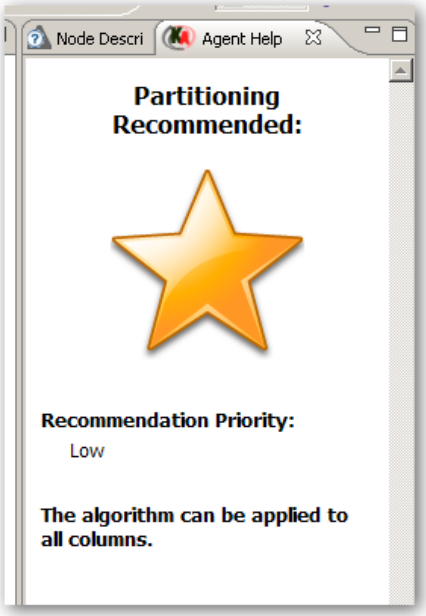

Figura 11. Vista de información recomendando el operador Partitioning

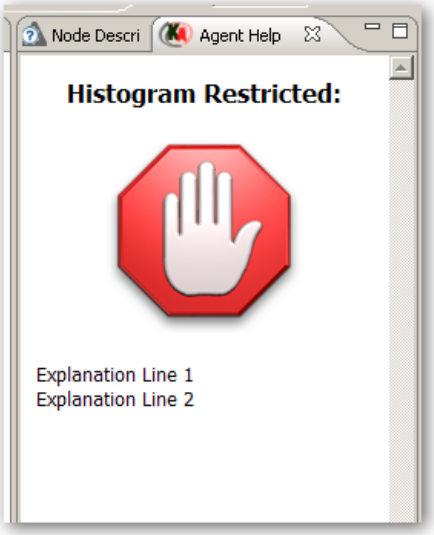

Figura 12. Vista de información restringiendo un nodo sin importar el contexto

Una técnica que no se encuentra afectada por conocimientos, igualmente posee asociada lógica del agente. En este caso si se desea aplicar Clustering al conjunto de variables de salida, el usuario recibe información sobre las variables que pueden utilizarse en base a los tipos de datos aceptados por dicho operador. En el ejemplo, se tienen 4 variables numéricas y una nominal que representa la clase de la planta. Clustering puede ser efectuado sobre valores numéricos, por lo que se informa sobre que variables puede efectuarse la técnica de Minería (es decir, las 4 variables numéricas) (Figura 13).

A modo de ejemplo, se efectuó un filtrado de columnas y como variables resultantes queda solamente la variable *clase* que es de tipo nominal. Si se pretende aplicar Clustering luego del filtro de columnas, se genera una restricción contextual informando que dicho operador solo acepta variables nominales. Adicionalmente, se le recomienda al usuario usar operadores que pueden convertir los tipos de datos, como la Numeración y la Discretización (Figura 14).

A medida que se realizan modificaciones en el *workflow* o proceso actual, se analiza mediante el algoritmo basado en AHP la posibilidad de una alta similitud con un proceso almacenado en la base. Cuando se detecta este fenómeno, se informa al usuario sobre la posibilidad de visualizar dicho proceso que potencialmente le sea útil considerando esta similitud. El usuario debe dirigirse al menú del Agente en la barra de herramientas para recuperar dicho proceso (Figura 15).

Además de poder recuperar un proceso almacenado, en el menú del agente pueden encontrarse otras opciones como: el registro de nuevos procesos, la habilitación/cancelación de la ejecución del agente (que permite continuar usando KNIME sin asistencia), la posibilidad de agregar nuevas restricciones permanentes y de definir tipos de datos rechazados como entrada para cualquier algoritmo del repositorio. De esta forma, se provee una interfaz completa para que pueda rellenarse la base de conocimiento por usuarios expertos. Por otro lado, el usuario podría

aprobar y rechazar la recomendación de *workflow* similar brindada anteriormente, para mejorar gradualmente esta característica de la herramienta.

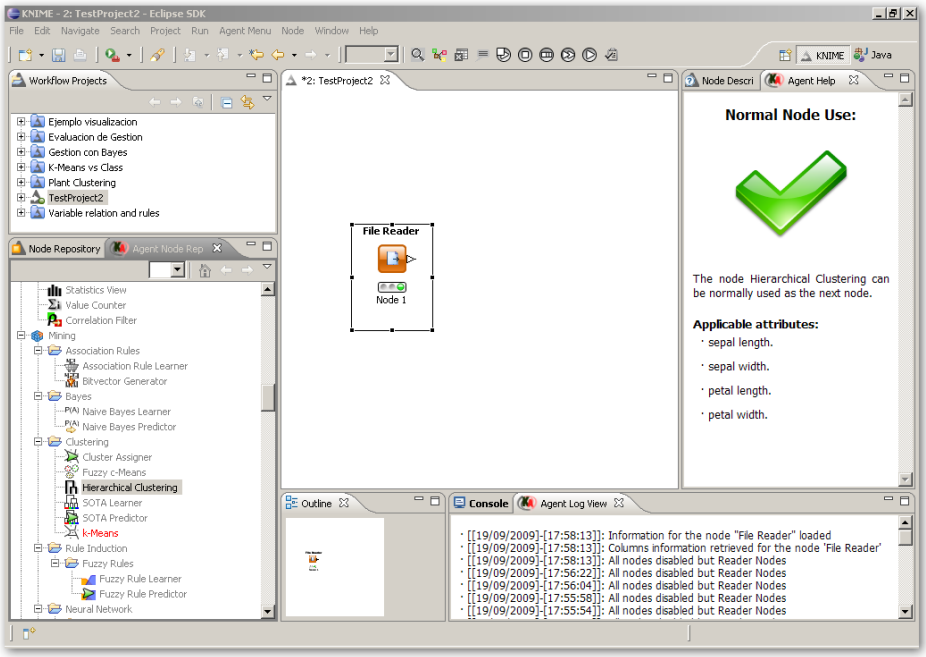

Figura 13. Análisis de los tipos de datos para continuar con el operador Clustering

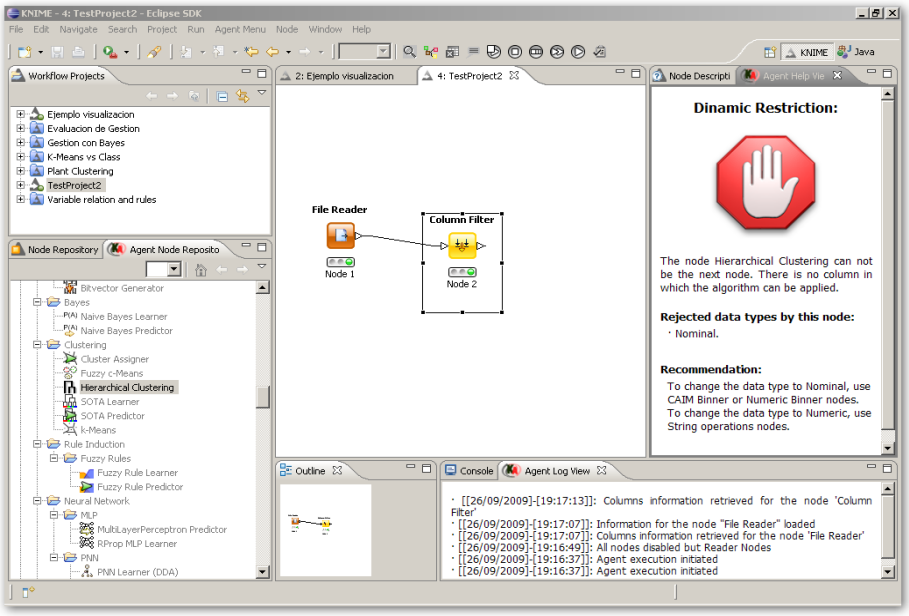

Figura 14. Restricción de operador Clustering por variables nominales

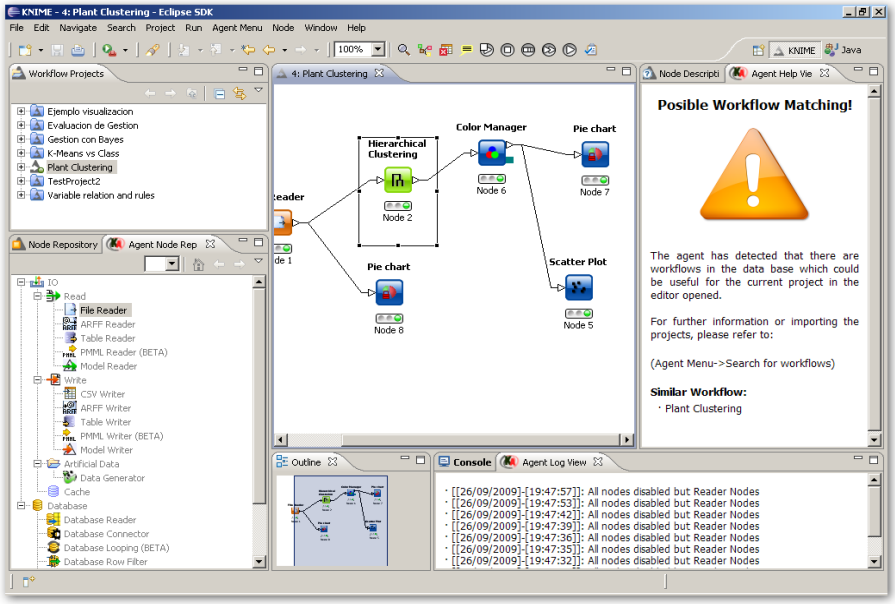

Figura 15. Recomendación de un proceso similar por AHP

Apéndice C: Cuestionario de evaluación de la herramienta

1. ¿Cómo calificaría la utilidad de las recomendaciones de operadores?

Muy buena	Buena	Media	Mala	Muy mala

2. ¿Cómo calificaría la utilidad de las restricciones (tanto dinámicas como estáticas)?

Muy buena	Buena	Media	Mala	Muy mala

3. ¿Con qué nivel calificaría la molestia ocasionada por el asistente durante su trabajo?

Muy bajo	Bajo	Media	Alto	Muy alto

4. ¿Cómo evaluaría la experiencia del usuario siendo asistido por el esquema propuesto?

Muy buena	Buena	Media	Mala	Muy mala

5. ¿Como calificaría la mejora general introducida por el asistente con respecto a KNIME original?

Muy buena	Buena	Media	Mala	Muy mala

6. ¿Por cuantos años ha cursado la carrera de Ingeniería en Sistemas?

7. ¿Cuántos finales de la carrera de Ingeniería en Sistemas posee aprobados?

8. ¿Ha utilizado alguna otra herramienta para Minería de Datos?

Si	No	¿Cuál/es?