IDENTIFICACIÓN DE PROBLEMAS DE APRENDIZAJE DE PROGRAMACIÓN CON EXPLOTACIÓN DE INFORMACIÓN

Jiménez Rey, E., Rodríguez, D., Britos, P., García-Martínez, R.

Centro de Ingeniería de Software e Ingeniería del Conocimiento. ITBA
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. UBA
Laboratorio de Sistemas Operativos y Base de Datos. Facultad de Ingeniería. UBA

ejimenezrey@yahoo.com.ar, drodrigu@itba.edu.ar, pbritos@itba.edu.ar, rgm@itba.edu.ar

Resumen

Presentamos un trabajo de investigación en progreso que se focaliza en herramientas de explotación de información para ayudar a los docentes a aplicar un proceso de descubrimiento del conocimiento en tres pasos para diagnosticar las dificultades de aprendizaje de los alumnos (y sus causas) relacionadas a sus errores de programación.

Palabras clave. Algoritmos TDIDT, Redes Bayesianas, Minería de Datos, Diagnóstico de Dificultades de Aprendizaje de los Alumnos.

1. INTRODUCCIÓN

La explotación de información ha sido señalada como una manera efectiva de descubrir nuevo conocimiento desde conjuntos de datos de procesos educacionales, datos generados por sistemas de aprendizaje o experimentos de aprendizaje, así como la información descubierta puede ser usada para probar la personalización y la adaptación. Entre los problemas interesantes que la explotación de información puede ayudar a resolver: determinación de cuáles son los estilos o estrategias de aprendizaje comunes, predicción del conocimiento e intereses de un usuario basado en su comportamiento previo, particionamiento de un grupo heterogéneo de usuarios en clusters homogéneos o detección de conceptos no comprendidos en procesos de aprendizaje.

Una de las técnicas más comunes de explotación de información son los árboles de decisión (TDIDT) usados para descubrir conocimiento en forma de reglas las cuales constituyen un modelo que representa el dominio del conocimiento subyacente a los ejemplos disponibles sobre el mismo. Una red bayesiana es un gráfico acíclico direccionado en el cual cada nodo representa una variable y cada arco representa una dependencia probabilística la cual especifica la probabilidad condicional de cada variable dada a sus padres; la variable a la cual el arco apunta es dependiente (causa – efecto) de la variable en el origen de esta.

Para descubrir concepciones erróneas de aprendizaje comunes de aprendices, el trabajo de investigación descripto, emplea la regla de asociación al perfil del aprendiz de explotación de información para diagnosticar conceptos no comprendidos comunes a los aprendices durante los procesos de aprendizaje. Las reglas de asociación que la ocurrencia del concepto no comprendido A implica la ocurrencia del concepto no comprendido B pueden ser descubiertas utilizando la aproximación diagnóstica de aprendizaje de la regla de asociación propuesta.

En este artículo presentamos resultados parciales de un proyecto de investigación en desarrollo que se focaliza en la utilización de herramientas de explotación de información para ayudar a los

docentes a diagnosticar las dificultades de aprendizaje de los alumnos (y sus causas) relacionadas a sus errores de programación con base en un proceso de descubrimiento del conocimiento de tres pasos. Para ello se introducen algunas consideraciones de qué medir y por qué (sección 2), se propone un proceso de descubrimiento del conocimiento en tres pasos (sección 3), se exploran a modo de ejemplo casos reales tomados de una población de estudiantes de un curso de introducción a la programación (sección 4), y se muestran conclusiones preliminares e investigaciones futuras (sección 5).

2. DOMINIO DEL PROBLEMA

Una lista de componentes de conceptos no comprendidos del modelo del estudiante a ser evaluado ha sido identificada. Algunos de estos componentes, su justificación y el rango de valores posibles se describen en los siguientes parágrafos, clasificándolos teniendo en consideración tres aspectos fundamentales en la creación de programas: metodología de desarrollo, funcionalidad del programa y calidad del diseño.

2.1. Metodología de desarrollo

El estudiante aplica método de refinamientos sucesivos: este componente busca diagnosticar si el estudiante ha adquirido la habilidad analítica de descomponer un problema complejo en partes más simples (estrategia divide y vencerás). Las respuestas tabuladas pueden ser: <sí, no, en forma incompleta>.
El estudiante describe el significado de variables: este componente busca detectar si el estudiante tiene la capacidad de explicar para qué define los recursos que usa en el desarrollo del programa. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante tiene estilo de escritura: este componente busca detectar si el estudiante tiene en cuenta el mantenimiento de un programa, es decir, que el programa puede ser leído y/o utilizado por otros programadores o incluso por sí mismo en un futuro cercano, por lo cual debe ser escrito en forma clara e inteligible. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante usa enunciados de documentación interna: este componente busca detectar si el estudiante ha adquirido la noción de programa autodocumentado, es decir, autoexplicativo. Las respuestas tabuladas pueden ser: <sí, no, en forma incompleta>.

2.2. Funcionalidad del programa

El estudiante descubre el algoritmo: este componente busca evaluar si el estudiante ha desarrollado la habilidad de disponer las sentencias primitivas de programación en una secuencia lógica de acuerdo con el objetivo del problema a ser resuelto. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante generaliza la solución: este componente busca detectar si el estudiante ha incorporado el concepto de algoritmo como procedimiento de resolución de una clase de familia de problemas a la cual pertenece el problema propuesto. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante utiliza delimitadores begin-end correctamente: este componente busca explorar si el estudiante ha incorporado el concepto de ablocamiento de sentencias primitivas o composición de

sentencias para ser ejecutadas como sentencia única. Las respuestas tabuladas pueden ser: <sí, no, no se pudo evaluar>.
El estudiante descubre la naturaleza del problema: este componente busca explorar si el estudiante ha realizado un análisis apropiado del problema identificando el tipo o tipos de estructuras de control que permiten resolverlo y las condiciones que gobiernan la solución. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante comprende el objetivo del problema: este componente busca identificar si el estudiante ha comprendido claramente cuál es el problema que debe resolver el programa. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante logra funcionamiento del programa: este componente busca indagar si el estudiante logra encontrar una solución productiva al problema obteniendo el resultado correcto. Las respuestas tabuladas pueden ser: <sí, no, sí con algún error>.
El estudiante realiza prueba de escritorio: este componente busca identificar si el estudiante ha completado las fases del proceso de creación de un programa evaluando si produce el resultado esperado de acuerdo con el enunciado del problema. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante asigna correctamente: este componente busca detectar si el estudiante ha comprendido genuinamente cómo el programador puede alterar el contenido de un espacio de memoria. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante comete errores de sintaxis: este componente busca controlar si el estudiante se ha entrenado suficientemente en la escritura de programas en el entorno de desarrollo de Pascal. Las respuestas tabuladas pueden ser: <sí, no>.

2.3. Calidad del diseño

El estudiante obtiene una solución lógica: este componente busca indagar si el estudiante ha alcanzado la madurez necesaria para diseñar una solución de buena calidad. Las respuestas tabuladas pueden ser: <muy buena, buena, regular, mala>.
El estudiante controla finalización ciclo repetitivo: este componente busca evaluar si el estudiante ha incorporado el concepto de algoritmo como un procedimiento que provee una solución en un tiempo finito verificando que una estructura iterativa efectivamente termine. Las respuestas tabuladas pueden ser: <sí, no, no siempre, no evaluado>.
El estudiante usa conectores lógicos correctamente: este componente busca diagnosticar si el estudiante ha detectado la relación entre el tipo de problema a resolver y las condiciones que controlan la repetición como reguladores de la generalización de la solución del problema. Las respuestas tabuladas pueden ser: <sí, no, no evaluado>.
El estudiante desarrolla un ciclo infinito: este componente busca detectar si el estudiante controla ell cuerpo de los ciclos iterativos para verificar el cambio de valor de las variables que conforman la expresión condicional asociada a la finalización del ciclo iterativo. Las respuestas tabuladas pueden ser: <sí, no, no evaluado>.
El estudiante comprende el objetivo del problema: este componente busca identificar si el estudiante ha comprendido claramente cuál es el problema que debe resolver el programa. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante inicializa variables de condición antes del while: este componente busca explorar si el estudiante ha comprendido el funcionamiento de esta estructura iterativa como pretest, es decir, con control de condición al inicio. Las respuestas tabuladas pueden ser: <sí, no, no evaluado>.
El estudiante usa indistintamente while y repeat: este componente busca detectar si el estudiante ha adquirido la habilidad de usar correctamente cualquiera de las dos formas de estructuras

iterativas gobernadas por control de condición al inicio y al final. Las respuestas tabuladas pueden ser: <sí, sólo Repeat, sólo While>.
El estudiante usa símbolo de asignación en expresión condicional: este componente busca identificar si el estudiante ha adquirido la capacidad de diferenciar el significado de un símbolo de asignación del significado de un operador de comparación. Las respuestas tabuladas pueden ser: <sí, no, no evaluado>.
El estudiante optimiza uso de recursos: este componente busca detectar si el estudiante ha adquirido la habilidad de depurar el diseño del algoritmo logrando una buena solución al problema mediante el uso de recursos necesarios. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante evita repetición de código: este componente busca explorar si el estudiante ha incorporado la noción de algoritmo como una secuencia de sentencias ejecutables. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante utiliza apropiadamente estructura case: este componente busca detectar si el estudiante ha comprendido el funcionamiento de la estructura case como sentencia de selección múltiple y sus limitaciones de uso. Las respuestas tabuladas pueden ser: <sí, no>.
El estudiante anida estructuras selectivas: este componente busca detectar si el estudiante ha logrado identificar las distintas situaciones de uso de las estructuras selectivas (con opción única, con opción doble y con opción múltiple). Las respuestas tabuladas pueden ser: <sí, no>.

3. PROCESO DE DESCUBRIMIENTO DEL CONOCIMIENTO

Para descubrir conceptos mal comprendidos de programación de los estudiantes, esta investigación se focaliza en el uso de herramientas de bases de datos de sistemas inteligentes: inducción de reglas por algoritmos TDIDT y redes bayesianas. Estas reglas son utilizadas en un proceso de descubrimiento del conocimiento en tres pasos:

Primer Paso: Construir una base de datos sobre una caracterización estándar de cada estudiante y su estilo y conceptos mal aprendidos de programación. Esta caracterización se focaliza en aspectos de: metodología de desarrollo del programa escrito por el estudiante, funcionalidad del programa desarrollado por el estudiante y calidad del diseño del programa creado por el estudiante.

Segundo Paso: Proceder al descubrimiento de reglas (mediante algoritmos TDIDT) las cuales establecen una relación entre conceptos no comprendidos de programación y sus posibles causas. Las reglas obtenidas son aplicadas para revisar la estructura propuesta para el curso y las hipótesis de los profesores. Los resultados experimentales indican que aplicando este paso, el profesor puede descubrir posibles causas de los conceptos mal aprendidos por los estudiantes.

Tercer Paso: Descubrir el peso que cada causa tiene sobre cada concepto mal aprendido (mediante redes bayesianas). Esto permite establecer un rango de importancia de las causas de conceptos no comprendidos para proponer estrategias de enseñanza remediales.

4. CASOS DE EJEMPLO

Se ha realizado un estudio preliminar de un curso introductorio de programación con una población de 88 estudiantes. El profesor se ha focalizado en los siguientes problemas:

1.- ¿Cuáles son las posibles causas por las cuales el estudiante no logra encontrar un algoritmo que solucione el problema planteado en el enunciado?

2.- ¿Cuáles son las posibles causas por las cuales el estudiante no logra diseñar una solución lógica de buena calidad para resolver el problema planteado en el enunciado?

4.1 CASO: Problema 1

El Profesor de este curso desea explorar qué conceptos mal aprendidos están relacionados con el hecho de que los estudiantes no logren descubrir el algoritmo correcto asociado al problema propuesto *("El estudiante descubre el algoritmo=no").*
Se definió como variable de clase el componente ¿Logra Descubrir el Algoritmo? y como categorías de predicción los componentes ¿Descubre la Naturaleza del Problema? ¿Comprende el Objetivo del Problema? ¿Consigue Generalizar la Solución? ¿Aplica Método de Refinamientos Sucesivos? ¿Logra Funcionamiento del Programa? ¿Obtiene una Solución Lógica? ¿Controla Condición Fin de Ciclo Repetitivo? ¿Usa Conectores Lógicos en forma correcta?
A partir de esta base de datos desarrollada como Paso 1 del estudio del curso sobre evaluaciones de programas, se aplicó el Paso 2 y usando algoritmos TDIDT se obtuvo el conjunto de reglas de predicción que se muestran en la Tabla 1.

Regla 1	**SI**	**el alumno obtiene una solución lógica = mala**
	ENTONCES	**el alumno logra descubrir el algoritmo = no**
Regla 2	**SI**	**el alumno obtiene una solución lógica = buena**
	ENTONCES	**el alumno logra descubrir el algoritmo = sí**
Regla 3	**SI**	**el alumno logra el funcionamiento del programa = sí, con algún error**
	ENTONCES	**el alumno logra descubrir el algoritmo = sí**
Regla 4	**SI**	**el alumno aplica método de refinamientos sucesivos = en forma incompleta**
	ENTONCES	**el alumno logra descubrir el algoritmo = no**
Regla 5	**SI**	**el alumno aplica método de refinamientos sucesivos = sí**
	ENTONCES	**el alumno logra descubrir el algoritmo = sí**
Regla 6	**SI**	**el alumno consigue generalizar la solución = no**
	ENTONCES	**el alumno logra descubrir el algoritmo = no**

Tabla 1. Reglas obtenidas por TDIDT

Con relación a la calidad (interpretación) de las reglas, el 42,3% de las observaciones soportan la Regla 1 siendo la confianza en la misma del 100,0% y su representatividad del 70,2%, el 29,5% de las observaciones soportan la Regla 2 siendo la confianza en la misma del 95,7% y su

representatividad del 71,0%, el 6,4% de las observaciones soportan la Regla 3 siendo la confianza en la misma del 100,0% y su representatividad del 16,1%, el 56,4% de las observaciones soportan la Regla 4 siendo la confianza en la misma del 75,0% y su representatividad del 70,2%, el 34,6% de las observaciones soportan la Regla 5 siendo la confianza en la misma del 63,0% y su representatividad del 54,8% y el 53,8% de las observaciones soportan la Regla 6 siendo la confianza en la misma del 97,6% y su representatividad del 87,2%.
A partir de los resultados descriptos el Docente formula las siguientes reflexiones:

- Los alumnos han sido evaluados en forma individual en cuanto a la habilidad desarrollada para la creación de programas Pascal con estructuras de control básicas y tipos de datos simples en la clase 7 (de un total de 16 clases). En muchos alumnos la adquisición de la habilidad de resolver un problema mediante un algoritmo está aún en proceso. (Regla 1)

- Pocos alumnos logran el funcionamiento correcto del programa en todas las situaciones de prueba; se presenta algún error en situaciones extremas de uso. No obstante, se evalúan otros aspectos, además de la funcionalidad, de igual ponderación, como la metodología aplicada y la calidad de la solución, para calificar el programa desarrollado pro el alumno como solución al problema planteado. (Regla 3)

- La aplicación del método de refinamientos sucesivos en forma completa puede implicar que el alumno ha vislumbrado un camino de solución posible al problema siendo capaz de implementarlo en lenguaje Pascal mediante el refinamiento apropiado para aproximarse a la solución; sin embargo, siempre existe la posibilidad de que la vía de solución elegida no sea la apropiada para conducir a una solución del problema. (Reglas 4 y 5)

4.2 CASO: Problema 2

El Profesor de este curso desea explorar qué conceptos mal aprendidos están relacionados con el hecho de que los estudiantes no logren diseñar una solución lógica de buena calidad asociada al problema propuesto *("El estudiante obtiene una solución lógica buena=no").*
Se aplicó Redes Bayesianas (BN) en el Paso 3 para determinar cuáles componentes (del total de componentes antes referenciados) tienen mayor impacto sobre la calidad del diseño del programa desarrollado por el alumno. Los resultados obtenidos se muestran en la Tabla 2.
A partir de los resultados descriptos el Docente formula las siguientes reflexiones:

- Un refinamiento completo por pasos del problema propicia el alcance de una solución de buena calidad. (Causa 1)
- La experiencia previa en lógica de programación no constituye un obstáculo o interferencia en el logro de un diseño de buena calidad. (Causa 2)
- El dominio del funcionamiento y uso de las estructuras repetitivas son indicativas de la capacidad del alumno para arribar a una buena solución lógica. (Causas 3 y 4)
- El logro del descubrimiento del algoritmo y de la generalización de la solución deviene en el desarrollo de un programa de buena calidad. (Causas 7 y 9)

COMPONENTES	CALIDAD			
	Muy Buena	Buena	Regular	Mala
El estudiante aplica método de refinamientos sucesivos = sí	71%	50%	33%	14%
El estudiante aplica método de refinamientos sucesivos = incompleto	14%	38%	67%	69%
El estudiante aplica método de refinamientos sucesivos = no	14%	12%	-	17%
El estudiante programó alguna vez = no	86%	58%	60%	86%
El estudiante programó alguna vez = sí, en lenguaje Pascal	-	17%	20%	5%
El estudiante programó alguna vez = sí, en otro lenguaje	14%	25%	20%	10%
El estudiante controla finalización ciclo repetitivo = sí	86%	67%	67%	43%
El estudiante controla finalización ciclo repetitivo = no	14%	8%	13%	43%
El estudiante controla finalización ciclo repetitivo = no siempre	-	25%	20%	-
El estudiante usa conectores lógicos correctamente = sí	100%	79%	67%	67%
El estudiante usa conectores lógicos correctamente = no	-	17%	20%	14%
El estudiante usa conectores lógicos correctamente = no evaluado	-	4%	13%	19%
El estudiante descubre la naturaleza del problema = sí	86%	100%	87%	24%
El estudiante descubre la naturaleza del problema = no	14%	-	13%	76%
El estudiante comprende el objetivo del problema = sí	86%	100%	93%	79%
El estudiante comprende el objetivo del problema = no	14%	-	7%	21%
El estudiante generaliza la solución = sí	86%	100%	47%	-
El estudiante generaliza la solución = no	14%	-	53%	100%
El estudiante logra funcionamiento del programa = sí	57%	25%	7%	-
El estudiante logra funcionamiento del programa = no	29%	62%	87%	100%
El estudiante logra funcionamiento del programa = sí, con algún error	14%	12%	7%	-
El estudiante descubre el algoritmo = sí	86%	96%	20%	-
El estudiante descubre el algoritmo = no	14%	4%	80%	100%

Tabla 2. Pesos obtenidos por BN

5. CONCLUSIONES PRELIMINARES E INVESTIGACIÓN FUTURA

La Explotación de información aplicada para asistir a los profesores en el descubrimiento de las causas de conceptos mal comprendidos por los estudiantes es una nueva herramienta de diagnóstico a explorar en el área.

Los resultados actuales sobre un conjunto de 88 registros de evaluaciones de los estudiantes de un curso inicial a Pascal son prometedores pero no concluyentes.

El próximo paso consiste en ajustar el proceso sobre una población integrada por 300 estudiantes de un curso introductorio a la programación de la Facultad de Ingeniería de la Universidad de Buenos Aires.

6. REFERENCIAS

Beck, J. , Calders, T., Pechenizkiy, M., Viola, S. 2007. *Workshop on Educational Data Mining*. ICALT'05: 933-934.

Britos, P., Cataldi, Z., Sierra, E., García-Martínez, R. 2008. *Pedagogical Protocols Selection Automatic Assistance*. LNAI 5027 (in press).

Chen , C., Hsieh, Y. 2005. *Mining Learner Profile Utilizing Association Rule for Common Learning Misconception Diagnosis*. ICALT'05: 588-592.

Felgaer, P., Britos, P. and García-Martínez, R. 2006. *Prediction in Health Domain Using Bayesian Network Optimization Based on Induction Learning Techniques*. International Journal of Modern Physics C 17(3): 447-455.

Salgueiro, F., Cataldi, Z., Britos, P., Sierra, E. y García Martínez, R. 2006. *Selecting Pedagogical Protocols using SOM*. Research in Computing Science Journal, 21: 205-214.

Schulte, C., Bennedsen, J. 2006. *What do teachers teach in introductory programming?*. ICERW´06: 17-28.