FACULTAD 4
DEPARTAMENTO DE TÉCNICAS DE PROGRAMACIÓN

UN JUEZ EN LÍNEA AJUSTADO A LAS NECESIDADES DE LA DOCENCIA

Tesis presentada en opción al título de Máster en Informática Aplicada

Autor: Lic. Tomás Orlando Junco Vázquez

Tutores: DrC. Rafael Arturo Trujillo Rasúa

MSc. Dainys Gainza Reyes

La Habana, Noviembre de 2012

Dedicatoria

A mis amigos.

A mi novia Yamila.

A mi hermana.

A mis padres.

A la Iniciativa Xtreme y al MPC-TLJ.

Agradecimientos

Sencillamente no cabe en ochenta páginas la lista de personas que durante mi vida me han apoyado. Y ciertamente todos esos empujoncitos han resultado en que hoy me haga Máster. Entre los de siempre agradezco a mis padres, a mi hermana y a toda mi familia en general que siempre han confiado más en mí que yo mismo. A Yamila por su mucho apoyo, por su cariño y por su invaluable ayuda incluso en la confección de esta tesis.

Conocí los jueces en línea gracias a Karel uno de mis compañeros en la etapa de adiestramiento que me exhortó a asistir a una Copa Void, por eso le agradezco. Me motivé tanto con esa competencia, donde podía ejercitar y mejorar los conocimientos de algoritmia aprendidos en la Universidad de La Habana, que no me pude perder una edición a partir de ese momento. La Copa Void evolucionó hasta que su fundador Dovier se ha convertido en el líder del ACM-ICPC en todo El Caribe. Dovier sin dudas ha sido un elemento clave en la culminación de este trabajo.

Nunca podré dejar de reconocer que quien me dio esta idea viendo lo ensimismado que estaba yo con los jueces en línea, fue Yuselys. Mi antiguo Decano Alcides cuando vio mi primera exposición sobre el tema, me dio mucho ánimo diciéndome que estaba "…en la autopista de conocimiento…".

Agradezco a mis muchachos de siempre, principales eslabones a lo largo de mi investigación sobre los jueces en línea, los que hoy ya son profesores y algunos mis compañeros de trabajo. En especial y en orden cronológico mi agradecimiento eterno para Enrique, Amado, Raciel, Lara y Leandro. A Leandro en especial le debo muchísimo por el apoyo final y decisivo… todos lo saben.

Agradezco a Roque y Lobaina, creadores del COJ, sin cuyo apoyo no hubiera sido este trabajo posible.

Agradezco a mis amigos de siempre y a todas las personas que con intención de apoyarme y darme el ánimo que necesité tantas veces me preguntaron ¿cómo va la maestría? A mi hermana Surelys siempre preocupada por mí, a Yorgelys, al Joe que fue el primero en darme un par de Gb de RAM para la máquina del jurado y que me apoyó mucho otras tantas veces. Agradezco a Maritza que un día me habló como una madre dándome ánimos para que terminara esta maestría…

A mi tutores Dainys y Trujillo agradezco especialmente. Supieron darme siempre, entre su tremendo cúmulo de trabajo, la guía necesaria…

Declaración de autoría

Declaro por este medio que yo, Tomás Orlando Junco Vázquez, con carné de identidad 81120701701, profesor de la Universidad de las Ciencias Informáticas, soy el autor principal del trabajo final de maestría “Un juez en línea ajustado a las necesidades de la docencia” desarrollado como parte de la Maestría en Informática Aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año _____.

Autor: Lic. Tomás Orlando Junco Vázquez

Tutor: DrC. Rafael Arturo Trujillo Rasúa

Tutor: MSc. Dainys Gainza Reyes

Resumen

El uso de los jueces en línea como aplicaciones web que automatizan el proceso de evaluación de soluciones algorítmicas a problemas de programación, ha promovido el surgimiento de propuestas de su inclusión en los procesos docentes en las carreras de perfil informático. Estas ideas consisten en añadir funcionalidades y asimilar otras ya presentes en los principales jueces en línea existentes en el mundo. En este trabajo se revisan varias experiencias que corroboran el hecho de que la inclusión en el ambiente docente de estas herramientas puede tributar a un mejor desarrollo cognitivo de los estudiantes en las asignaturas de Programación a partir de una ejercitación ininterrumpida con retroalimentación constante sobre la calidad de las soluciones implementadas. Se propone un juez en línea personalizado para ser integrado en el proceso enseñanza-aprendizaje de la disciplina Programación de la carrera Ingeniería en Ciencias Informáticas. Además, por el importante valor agregado que aportan a la solución, se describen en detalle un subsistema *recomendador de problemas a resolver* y un *módulo detector de soluciones plagiadas* adjuntos en la herramienta.

Índice

Introducción

En carreras con perfil computacional, la enseñanza de asignaturas relacionadas con la Programación es uno de los aspectos más debatidos y de más controversia. Al respecto diversos maestros y autores de distintas universidades del mundo han expuesto sus maneras y modelos de enseñanza (Robins, 2003) (Oviedo, 2005) (Gries, 2006) (Petit, 2009) (Shun-xin, y otros, 2011) (Othman, y otros, 2012). Sin embargo, ésta sigue y seguirá siendo un elemento recurrente, pues a diferencia de como ocurre con otras disciplinas no existe para la Programación, aún, un modelo de enseñanza de éxito demostrado. A diferencia de las matemáticas, la Programación es una disciplina relativamente joven que no se imparte en enseñanzas precedentes.

La importancia de este punto está dada por la premisa de que un buen profesional de la Computación, o la Informática, debe haber adquirido las habilidades del sistema de esta disciplina desde los años tempranos de la carrera, lo cual es un prerrequisito para cursar con éxito varias asignaturas posteriores así como para un adecuado desempeño del futuro profesional (Oviedo Galdeano, y otros, 2005) (Tiantian, 2011). Por tal motivo la literatura existente, respecto a la enseñanza de la programación, hace énfasis en los cursos introductorios (Kurnia, y otros, 2001).

A nivel mundial, cada año se han ido acentuando en los alumnos de nuevo ingreso algunas carencias relacionadas con el dominio de esta materia (Petit, 2009). Entre estas están la falta de habilidades de algoritmización, el desconocimiento total de los paradigmas y herramientas de programación, así como la falta de organización y disciplina a la hora de programar (Othman, y otros, 2012).

Los resultados del cuestionario aplicado cada año en la Universidad de las Ciencias Informáticas (UCI) a estudiantes de nuevo ingreso, igualmente permiten concluir que la preparación inicial de los estudiantes que ingresan es baja: ninguno de los ingresados en la Facultad 4 ha tenido anteriormente buenos resultados en concursos de matemática o computación, apenas un 5% admite tener conocimientos previos sobre algún lenguaje de programación y menos del 15% se ha vinculado con algún movimiento de jóvenes relacionado con la Computación (ver anexo 1). Esta

caracterización inicial permite tener apreciaciones previas sobre el bajo desempeño de los estudiantes en esta disciplina.

Información brindada por el Departamento Docente Central de Técnicas de Programación muestra como los resultados en las asignaturas de esta disciplina no son los deseables. El curso 2011-2012 concluye con un 60.46% de aprobados en la asignatura Introducción a la Programación (IP), un 57.38% en la asignatura Programación I (P1) y peor aún con 40.79% de aprobados en la asignatura programación II (P2). En el anexo 2 se pueden revisar con más detalle estos datos, además de los también mejorables resultados del curso 2010-2011.

Varias opiniones coinciden en que se aprende a programar programando (Oviedo, 2005), (Nghi, 2007), (Joy, y otros, 2005), claro está, basando todo este proceso de mucha práctica en elementos como la motivación de los estudiantes y el papel orientador-controlador del profesor.

Debe citarse que una de las más estimulantes estrategias asumidas para incrementar las aspiraciones de los estudiantes en el área de la programación y mejorar su rendimiento, ha sido el desarrollo de los concursos de programación (Warsaw, 2012). En particular, surgiendo de una competición que hubo en la Universidad A&M de Texas en 1970 y devenida actualmente en el evento de programación más grande y prestigioso del mundo, el Concurso Internacional Universitario ACM[1] de Programación (ACM-ICPC[2]) constituye el mejor ejemplo para ilustrar cómo la competitividad sana puede motivar, de manera natural, a un sinnúmero de estudiantes universitarios a mejorar sus habilidades como programadores. Este evento pasó desde 1977 a ser una competición con varias rondas clasificatorias que culminan en la Final Mundial y desde entonces se han incrementado cada año los números de participantes hasta el caso de 2011 con la participación de 25016 concursantes de más de 2200 universidades. En estos eventos, los concursantes tienen que resolver un conjunto de problemas donde aplican conocimientos que van desde el diseño y análisis de algoritmos hasta las más complejas teorías matemáticas, desarrollando habilidades como el pensamiento algorítmico, la resolución de problemas, el trabajo en equipo y la laboriosidad; valores importantes en un profesional de la Computación.

[1] Association for Computing Machinery (http://www.acm.org/)
[2] ACM International Collegiate Programming Contest (http://www.acmicpc.org y http://icpc.baylor.edu)

En nuestro país el objetivo de este tipo de concursos fue logrado rápidamente. A partir del año 2009, en que Cuba se incluyó con más regularidad en el movimiento ACM-ICPC, se han ido organizando y consolidando los eventos a nivel de universidad, nacional y regional. En pocos años se ha logrado incluir cuatro equipos cubanos en finales mundiales.

Como eventos homólogos existen en la UCI las reconocidas *Copa Pascal*, *Copa Xtreme Programming* y *Copa Void*, además de (como se muestra en el anexo 3) otras competencias que de manera regular se efectúan en las distintas facultades. A nivel nacional se tienen referencias de distintas competencias en casi todas las instituciones que han logrado sumarse al ACM-ICPC (ver en http://coj.uci.cu/contest/past.xhtml). Este tipo de experiencias se sincronizan totalmente con el concepto de *aprendizaje competitivo* (Revilla, 2008), y aunque muestran más bien un carácter de élite por la participación casi total de estudiantes de alto rendimiento, resultan interesantes para ser extrapoladas al proceso de enseñanza de la programación.

En varias instituciones, incluida la UCI, donde se imparten distintos cursos de programación con grandes matrículas, constituyen aspectos críticos el complejo tema de la evaluación (Joy, y otros, 2005) y la dificultad para retroalimentar a los estudiantes sobre la calidad de las actividades realizadas. Para que un estudiante pueda practicar lo suficiente, debe tener varias tareas y ejercicios propuestos, y debe recibir una retroalimentación constante. Sin embargo, en universidades como la UCI donde los profesores en general tienen además de la docencia otras actividades que realizar en las esferas de la investigación y la producción, el recurso tiempo se vuelve muy limitado, lo cual repercute en la orientación de menos tareas que las necesarias para los estudiantes debido a la imposibilidad del profesor de dar la adecuada y temprana retroalimentación a sus educandos. En el mejor de los casos las actividades se asignan, pero varias quedan opcionales y normalmente la mayoría de los estudiantes no las realizan pues, en general, no cobra sentido para los estudiantes realizar una actividad que su profesor probablemente no podrá supervisar (Kurnia, y otros, 2001).

Es fundamental que el profesor juegue bien su papel como ente orientador-controlador en el proceso de enseñanza, teniendo plena constancia del avance real de cada uno de sus estudiantes en la asimilación de los contenidos impartidos, y que a partir de una observación detallada trace sus estrategias de seguimiento. Los alumnos de alto rendimiento (aquellos que realizan sus tareas sin dificultad) deben recibir tareas

adecuadamente más complejas para mejorar su aprovechamiento de los cursos; se debe ser cuidadoso en cuanto al nivel de complejidad de las nuevas tareas para no frustrar el avance. De la misma forma los estudiantes de rendimiento bajo deben recibir tareas que faciliten la asimilación previa de contenidos base.

La orientación, respondiendo a la pregunta de cómo y qué actividades debe realizar cada estudiante en particular, exige del profesor un completo dominio de cada una de las actividades vencidas por cada alumno para seleccionar qué nuevas actividades orientar a cada educando. Lograr lo planteado anteriormente es ideal, sin embargo pocas veces se materializa en plenitud pues resulta complicado saber qué ha hecho cada estudiante y sobre todo saber de entre tantas posibles tareas las más adecuadas a reasignar, de manera que cumplan el papel desarrollador deseado. Por demás, este tratamiento diferenciado exige más tiempo del profesor también con el tema de la evaluación pues no tendría sentido proponer nuevos ejercicios si no se va a controlar su realización. Es importante que las actividades asignadas a un estudiante se adecuen al nivel de adquisición de los conocimientos que este tenga.

En estos momentos la UCI está implementando un proceso de rediseño de las asignaturas de la carrera, obedeciendo al modelo de formación centrado en el aprendizaje. La disciplina Técnicas de Programación ha sido rediseñada casi en su totalidad, sobre todo las asignaturas IP, P1 y P2. El rediseño realizado propone varias actividades a realizar por el estudiante cada semana de manera independiente, y varias actividades no presenciales o semipresenciales. Otras asignaturas de la disciplina como Programación III (P3) y Programación IV (P4) proponen la resolución de muchos ejercicios prácticos de programación.

Este cúmulo de tareas independientes agrava una situación que es bastante general en los entornos académicos y en especial en los cursos de programación: el plagio. Informes semestrales de los colectivos de asignatura reportan la existencia de esta situación entre sus estudiantes. Desafortunadamente, este es un fenómeno bastante extendido a nivel internacional. Típicamente, existen estudiantes que motivados solamente por obtener una buena calificación copian soluciones de otros compañeros. Por otra parte la existencia en la UCI de siete facultades impide que una misma tarea sea revisada a nivel de toda la Universidad por el mismo equipo de profesores, por lo que es hoy una situación real y complicada la detección de soluciones plagiadas.

Considerando la anterior situación problemática se presenta el siguiente problema a resolver**:** ¿Cómo hacer más efectivo y menos costoso en cuanto a tiempo el proceso de orientación, control y evaluación de actividades prácticas independientes de la disciplina Técnicas de Programación?

Con el fin de dar respuesta a este problema, se selecciona como objeto de estudio los jueces en línea de programación y como campo de acción la aplicación de los jueces en línea en la enseñanza de la programación.

El objetivo general de la investigación es desarrollar un juez en línea personalizado al contexto docente de la UCI, que automatice procesos tales como evaluación de soluciones, recomendación de problemas y detección de plagios, de modo que pueda servir de apoyo al proceso de enseñanza-aprendizaje de la Disciplina Técnicas de Programación.

Como objetivos específicos de la investigación se definieron los siguientes:

1. Elaborar el marco teórico sobre las herramientas de apoyo a la enseñanza de la programación de computadoras existentes en el mundo, haciendo hincapié en su aplicación con fines docentes en otras instituciones.
2. Desarrollar una aplicación informática que corresponda con las características de un juez en línea y que integre características deseables para asegurar eficacia a partir de su inclusión en la docencia.
3. Validar la aplicación desarrollada.

Se tiene como hipótesis que el desarrollo de un juez en línea personalizado para el contexto docente de la UCI, facilitará el proceso de enseñanza de la Programación permitiendo la ejercitación a tiempo completo y automatizando de manera efectiva los procesos de orientación a los estudiantes y detección de plagio, en las actividades de esta disciplina.

Estructura del documento

El presente documento se encuentra estructurado en tres capítulos. En el primero se hace un análisis de los principales conceptos relacionados con los jueces en línea y posibles herramientas o utilidades que pudieran complementar sus prestaciones. Se realiza un estudio del estado del arte de los más importantes jueces en línea y de su empleo en varias instituciones del mundo, principalmente en el clásico contexto docente que son las asignaturas de una carrera.

En el segundo capítulo se describe la propuesta de solución, enfatizando los elementos más importantes que permiten su inclusión en el ambiente docente.

En el tercer y último capítulo se valida el resultado obtenido teniendo en cuenta principalmente los resultados de la aplicación desarrollada, y se enfatiza por ser el principal aporte de la solución, en la validación de los algoritmos implementados para el desarrollo de los módulos *Recomendación de Problemas* y *Detección Automática de Plagio*.

Capítulo 1. Fundamentación Teórica

La programación es considerada un arte y una ciencia que se cultiva, fundamentalmente, con la práctica constante en los momentos iniciales del aprendizaje. La calidad y éxito de los cursos de programación, dependen en gran medida del número de actividades prácticas que de manera independiente realicen los estudiantes (Giménez, 2012). Sin embargo, este número e incluso el número de actividades asignadas, disminuye notablemente respecto a lo que es deseable cuando las matrículas de los cursos son grandes y el claustro permanece igual, derivando esto es un cociente *cantidad de alumnos / cantidad de profesores* realmente elevado. La alta complejidad del proceso de evaluación de programas, la cual han sufrido varios departamentos de Ciencias de la Computación a nivel mundial, conllevó en los últimos años a considerar la posibilidad de automatizar este proceso (Cheang, y otros, 2003).

Son varias las instituciones que han tenido que enfrentarse con el problema de evaluar cursos introductorios de programación con grandes matrículas que convierten el tema evaluación en un punto crítico dentro del proceso de enseñanza de esta disciplina (Joy, y otros, 2005). Así mismo y como se refiere en la introducción de este trabajo también la evaluación ha sido un tema necesariamente abordado en la preparación de participantes del ACM-ICPC. Comoquiera que estas dos vertientes (docencia estándar y concursos al estilo ACM-ICPC) son las más preocupadas en la ejercitación de los estudiantes en conocimientos de programación, las herramientas que se describen en este trabajo están asociadas fundamentalmente a estas áreas.

En la definición del estado del arte asociado al contexto de este trabajo constituye un aspecto esencial abundar sobre qué son los jueces en línea, qué funciones principales ofrecen estos, cuáles de estas herramientas son las más importantes a nivel mundial y experiencias de su aplicación en otros contextos similares al contexto docente de la Universidad de las Ciencias Informáticas.

1.1 Jueces en línea

Si bien son variados los trabajos e intentos de solución al problema de automatizar la evaluación y la ejercitación de cuestiones de la programación, puede decirse que las

aplicaciones más conocidas y más probadas debido al gran auge de Internet en los últimos años son los jueces en línea.

Un juez en línea es una aplicación web para entornos generalmente académicos, que incluye varios problemas de distintas materias para ser resueltos mediante técnicas de programación y, lo más importante, evalúa automáticamente las soluciones de sus usuarios en los varios lenguajes de programación disponibles. Se caracterizan por contener una interfaz bien definida para la interacción con el usuario y un alto nivel de disponibilidad, permitiendo el libre acceso a la aplicación en todo momento a las personas registradas (Revilla, 2008).

El hecho de ser aplicaciones en línea les aporta varias ventajas con respecto a otros tipos de aplicaciones de escritorio concebidas también para la automatización de la evaluación en concursos:

Portabilidad: Al presentar una interfaz construida a base de estándares como HTML, pueden ser ejecutadas desde cualquier plataforma, con el único requisito de contar con un navegador que soporte dichos estándares.

Accesibilidad: Al estar publicadas en un servidor de Internet en el cual guardan todos sus datos y configuración, generalmente pueden ser accedidas desde cualquier computadora con acceso a la red de redes.

Colaboración: Al centrar todas las conexiones de los usuarios en un mismo servidor central, se facilita la colaboración y comunicación entre los mismos, haciendo énfasis en el trabajo en equipo y la distribución de la información.

En (Kurnia, y otros, 2001) se hace una de las mejores descripciones en lo que refiere a un modelo de juez en línea. Se aborda la aplicación de una herramienta llamada *Online Judge* en la preparación de concursantes del ACM-ICPC y como apoyo en cursos de programación en la Universidad Nacional de Singapur. Constituye una de las bases para varios desarrollos posteriores de jueces en línea en el mundo pues los autores describen en detalle las múltiples desventajas de la evaluación manual cargadas de subjetivismo y dependientes de las circunstancias en que se lleva a cabo la evaluación, así como las, también múltiples, ventajas de la evaluación automática que reduce notablemente el tiempo empleado y hace mucho menos preocupante el crecimiento de la matrícula de los cursos. Se describe el modelo básico, recomendado y genérico que utilizan los jueces en línea para evaluar. Se detallan algunas

experiencias de su aplicación en la preparación de concursantes y en apoyo a la impartición de asignaturas, así como detalles de la arquitectura deseada para un sistema de este tipo. La principal dificultad de la herramienta *Online Judge* para considerarla un modelo ideal es que no cuenta con un mecanismo de orientación a los estudiantes, para su avance en el aprendizaje.

La manera en que un juez en línea comprueba la corrección de un programa es sencilla: para cada problema a resolver existen uno o varios ficheros que representan las entradas para probar la corrección de las soluciones enviadas por los usuarios, a su vez existen igual cantidad de ficheros que constituyen las salidas correctas asociadas a cada una de esas entradas. Se compila la solución enviada por el usuario, escogiendo de manera transparente el compilador a utilizar según el lenguaje de programación utilizado. De no haber errores de compilación, el paso siguiente es ejecutar el programa para cada entrada definida y verificar que la salida de cada ejecución coincida exactamente con la salida correcta definida previamente para la entrada en cuestión. La solución enviada se considerará *correcta* si cada una de estas verificaciones fue exitosa y en caso contrario se considerará *incorrecta* (Walker, y otros, 2005).

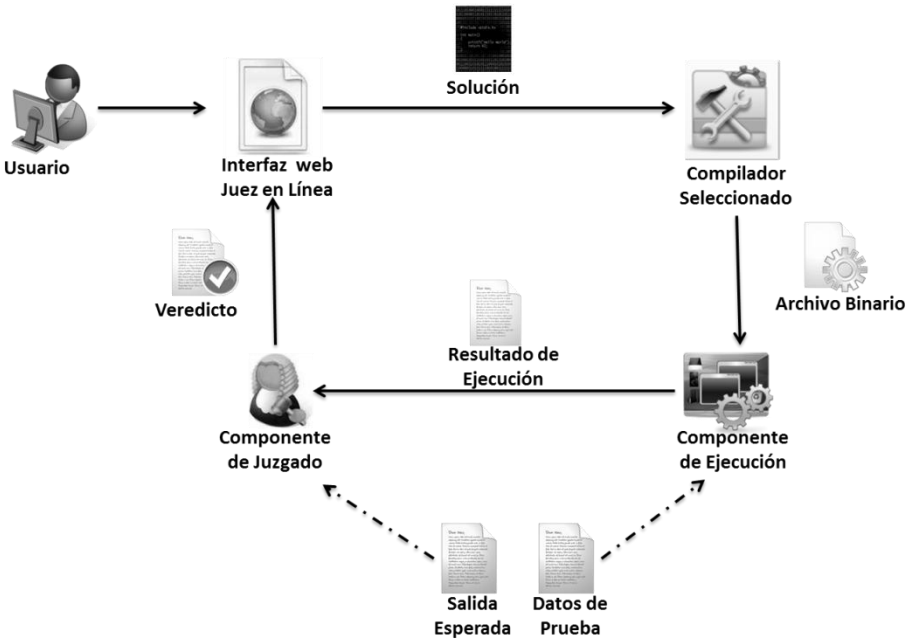


Figura 1. Esquema de funcionamiento general de los jueces en línea

Para garantizar un veredicto objetivo es muy importante que las entradas de prueba sean lo suficientemente abarcadoras teniendo en cuenta todos las posibles situaciones que se puedan presentar según el problema. En la figura 1 se representa el esquema general de funcionamiento de un juez en línea.

Correcta e *Incorrecta* no son los únicos veredictos que brinda un juez en línea (Petit, 2009). Generalmente incluyen otras variantes en el juzgado que sugieren al programador qué aspecto de su solución debe mejorar. Si en el momento de la compilación se detectara un error que no permitiera completar este proceso, la evaluación se interrumpe y el veredicto ofrecido es *Error de Compilación*. Para cada problema en el juez en línea se define un tiempo de ejecución que permite controlar que se implementen algoritmos eficientes para la resolución y, no menos importante, asegurar que el servidor no se sobrecargue con soluciones que pretendan ejecutarse por mucho tiempo. Si al ejecutar la solución enviada con alguna de las entradas de prueba el proceso tarda más del tiempo máximo definido, el sistema se encarga de forzar el paro de la ejecución y el veredicto es *Tiempo Límite Excedido*. Por último, si durante la ejecución con una entrada existe algún problema de funcionamiento (errores en tiempo de corrida) como una división por cero o acceso a una posición de memoria no reservada, el proceso también es detenido y el veredicto es *Error en Tiempo de Ejecución*. Más detalles sobre los posibles veredictos y sus causas típicas pueden revisarse en (Roque Alvarez, 2010).

1.1.1 Jueces en línea para entornos competitivos

Entre los jueces en línea de programación más conocidos e importantes a nivel global pueden mencionarse el Juez en Línea de la Universidad de Valladolid (UVA[3]), el Juez en Línea de la Universidad de Tecnología de Gdansk (SPOJ[4]), y el Juez en Línea de la Universidad de Pekín (PKU[5]).

En la UCI desde hace aproximadamente seis años se ha comenzado a emplear y desarrollar jueces en línea de programación propios. Actualmente los más importantes

[3] http://uva.onlinejudge.org/
[4] http://www.spoj.pl/
[5] http://poj.org/

son el Juez en Línea de la Cátedra de Programación Avanzada (CPAV[6]) y de manera relevante el Juez en Línea del Caribe (COJ[7]).

En materia del entrenamiento de concursantes ACM-ICPC así como en el aporte pedagógico el **UVA** es, a nivel mundial, el que goza de más prestigio en la comunidad universitaria internacional, quizás por ser uno de los más antiguos y el mejor surtido en ejercicios. Este sistema tiene como característica fundamental su organización de los problemas en extensos volúmenes nutridos normalmente de los concursos ACM-ICPC, constituyendo referencia obligada por parte de los interesados en el tema (Revilla, 2008). Este juez en línea solo soporta el envío de soluciones para los lenguajes de programación permitidos en el ACM-ICPC (C, C++, Java, Pascal).

Por su parte **SPOJ** es una solución tecnológicamente elegante, muy flexible y que soporta todas las funcionalidades básicas de un juez en línea para preparación de concursantes. Es un proyecto del *Sphere Interest Project*, entidad afiliada a la Universidad de Tecnología de Gdansk, en Polonia. Se destaca en el hecho de que permite la evaluación de soluciones en 47 lenguajes de programación diferentes. Sus problemas, traducidos al ruso, al portugués y al vietnamita, cubren varios de los presentados en las diversas ediciones del ACM-ICPC, así como otros aportados por los mismos usuarios y administradores del sitio. SPOJ mantiene varios *rankings*, en los cuales influye la puntuación de cada usuario. Presenta un sistema muy dinámico de puntuación asegurando que los problemas más complejos valgan más, y que los usuarios que pasen cierto tiempo sin realizar envíos bajen en el ranking a favor de los usuarios más activos. Cuenta con un módulo especializado de competencia, muy versátil y eficiente. Presenta un sistema de juzgado especial digno de mención, dado que tiene varias implementaciones de la forma de evaluación, estrictamente necesaria para algunas tipologías de problemas.

El Juez en Línea de la Universidad de los Urales (**Timus**[8]), en Rusia, es un proyecto desarrollado por estudiantes que ofrece las funcionalidades básicas de su tipo, con una interfaz intuitiva y fácil de usar. Mantiene dos versiones del mismo software (ruso e inglés), presenta una solución interna de comunicación (los jueces anteriores recaen en soluciones de terceros para implementar un foro de usuarios) y brinda la posibilidad

[6] http://cpav.uci.cu
[7] http://coj.uci.cu
[8] http://acm.timus.ru/

de recibir el reporte de la solución por correo. Las principales características de este juez en línea son sus buenos tiempos de respuesta y su solución de comunicación interna a través de una *webboard*. Permite al usuario filtrar los problemas por tipo de técnica de programación (por ejemplo *Programación Dinámica),* enfoque que podría extrapolarse en la docencia para permitir que los estudiantes eligieran entre un tipo de ejercicios en particular (por ejemplo *Estructuras de Datos Lineales, Estructuras repetitivas*…)

El **PKU** tiene varias funcionalidades en común con el Timus y otras muchas más que lo distinguen. Ofrece correo interno como complemento a su *webboard* (Wen-xin, 2005) (Luo, 2008). Es destacable el módulo de estadísticas, muy interesante y que permite apreciar el devenir de la herramienta desde que fue creada en 2003. El funcionamiento de este juez en línea es muy suave, intuitivo y rápido. El tráfico es alto, con un pico de 1199865 envíos de soluciones desde enero hasta septiembre de 2012, lo cual junto al número de envíos diarios lleva a suponer que este es el más visitado de todos los analizados, algo que respalda la calidad asiática en los concursos ACM-ICPC. Este sistema marcó los primeros pasos del desarrollo y despliegue de jueces en línea en la UCI mediante la versión descargable libremente que proporcionaran en 2006.

En especial, el **COJ** se encuentra disponible desde Junio de 2010 en la red nacional del Ministerio de Educación Superior (MES) y en Internet, con fines de entrenamiento de todos los concursantes de la Región Caribeña. El Equipo Desarrollador del COJ (CDEVT[9]) está compuesto por nueve personas, en su mayoría de la UCI. Juez en Línea del Caribe es una solución elegante que surge sobre todo de la necesidad ante los problemas de conectividad que limitaban en Cuba el entrenamiento de los concursantes ACM-ICPC. La versión actual (2.0) incluye archivo 24 horas, posibilidades de revivir competencias anteriores, solución de comunicación interna entre los usuarios, un excelente y completo módulo de administración, ejecución en ambiente seguro de las soluciones de los usuarios y posibilidad de realizar la evaluación de soluciones de manera distribuida lo cual le ha permitido un muy buen desempeño en las 212 competencias efectuadas hasta la fecha y dando soporte a más

[9] http://coj.uci.cu/general/about.xhtml#about.18
Parte del CDEVT pertenece al equipo de desarrollo del antiguo Juez en Línea de la Iniciativa Xtreme el cual ya no se encuentra disponible, pero fue el primer juez en línea que se conoce a nivel nacional.

de 7850 usuarios. Desafortunadamente actualmente no cuenta con funcionalidades de orientación a los usuarios ni detección automática de plagio.

El juez **CPAV** desarrollado en la UCI y disponible solo en la intranet de la Institución, rompe con los esquemas de los jueces en línea anteriores, en filosofía, herramientas y objetivos. Está desarrollado solo para Windows, usando el CMS PHP-Fusion. Para evaluar las soluciones utilizan un *daemon* aparte construido en C++. Este sistema ha acogido varias competencias, por ejemplo las copas GrundyPC, She Programmer y Troya, aunque en sus mejores momentos de explotación no soportó el gran número de usuarios conectados en las grandes competencias de la UCI. Ofrece varios tipos de ranking, por año, por facultades, por lenguajes y general, así como una serie de servicios que derivan de su calidad de CMS, incluyendo galería, *shoutbox*, anuncios e información de los últimos usuarios conectados. Se destaca su sistema de clasificación de los problemas en volúmenes por temas lo cual, si bien para entrenamiento de concursantes algunos lo consideran contraproducente, para la docencia es un aspecto interesante extrapolando un poco estos volúmenes a las categoría de asignaturas o temas estudiados en la carrera.

1.1.2 Jueces en línea para entornos docentes

Una experiencia importante de inclusión de un juez en línea en la docencia se tiene en la Universidad de Warwick (Heng, y otros, 2005), pionera y líder en estos temas de la evaluación automática, donde el claustro se encontró ante la dificultad de los cursos introductorios relacionados con la programación. En (Joy, y otros, 2005) se describe esta experiencia sustentada por la herramienta de código abierto **BOSS**[10] la cual tiene en cuenta tres aspectos fundamentales para la evaluación de programas de los estudiantes: corrección, estilos de código y autenticidad. Hacen un resumen de atributos que, según la literatura, son unánimemente considerados para evaluar un programa, atributos que no son todos susceptibles de evaluarse automáticamente. Concuerdan en que la corrección de un programa es uno de los aspectos más importantes a considerar cuya evaluación automática es posible. BOSS se destaca por incluir chequeo antiplagio apoyado en un sistema externo llamado Sherlock, algo de lo que los demás jueces en línea no disponen. Cuenta con un módulo para incluir los listados de estudiantes y demás aspectos relacionados con las características

[10] http://boss.org.uk/

específicas de la institución, lo cual constituye un elemento interesante para la inclusión de un juez en línea al contexto general universitario. La principal limitación de BOSS es que no tiene en cuenta ninguna funcionalidad de orientación a los estudiantes y además por estar tan ligado a los conceptos *módulo* y *tareas para los estudiantes* no permite disponer de actividades competitivas o del concepto de ranking, aspectos típicos y motivadores existentes en otros jueces en línea.

Utilizado por más de seis cursos en actividades docentes de la Facultad de Informática de Barcelona y la Facultad de Estadística de la Universidad de Cataluña, el entorno virtual de aprendizaje **Jutge.org**[11] ha sido especializado para la enseñanza de la Programación. En este sistema los estudiantes pueden resolver más de 800 problemas utilizando 22 lenguajes de programación distintos recibiendo retroalimentación instantánea. Los instructores pueden crear sus propios cursos e incluir documentos, bibliografía, orientar tareas, así como preparar exámenes y competencias. Es gratis para estudiantes y actualmente también para los instructores (Giménez, 2012). En la guía de uso no se hace explícito en la guía de uso cómo obtener el rol de profesor por parte de externos, solo se da una variante de ser instructor por 48 horas. Este sistema no brinda ningún tipo de orientación a los estudiantes. Tampoco brindan alguna posibilidad de detección automática de plagio. Los problemas están algunos redactados en catalán lo cual podría ser una dificultad para su uso en la UCI.

Practice-It[12] es una aplicación web que permite la resolución de problemas de programación utilizando el lenguaje Java. Varios de los problemas que reta a resolver provienen de los textos usados en el Curso Introductorio a Java, de la Universidad de Washington. Incluye un importante conjunto de problemas bien agrupados que comprenden los principales elementos de algoritmia y conocimiento de dicho lenguaje de programación. Ha sido usado en exámenes de los varios cursos que actualmente tiene configurados. Su principal limitación es que solo soporta Java como lenguaje de programación, lo cual lo hace solo utilizable para ambientes docentes (como ocurre en la UCI) donde sea Java el lenguaje seleccionado para instruir. No incluye funcionalidades para orientación a los estudiantes ni para detección de plagio.

En (Tiantian, 2011) se describe **AutoLEP**, un sistema automático para la evaluación y el aprendizaje desarrollado en el Instituto Tecnológico de Harbin. El propósito de este

[11] https://www.jutge.org/
[12] http://webster.cs.washington.edu:8080/practiceit/index.jsp

sistema es evaluar automáticamente los programas de los estudiantes además de ayudarlos a obtener mejores habilidades en la programación. Entre sus aspectos novedosos está su capacidad de chequear si un problema cumple con especificaciones predefinidas de codificación, característica que podría ser muy útil en el apoyo a las clases de Programación sobre todo en los cursos introductorios. Para un problema que requiera imprimir todos los valores de 1 hasta 20 los estudiantes pudieran imprimir estos valores uno por uno sin hacer uso de estructuras repetitivas lo cual, en específico, es el objetivo que probablemente quiera controlar el profesor. Permite evaluar parcialmente las soluciones enviadas, distinto a lo que hacen la mayoría de otros jueces en línea cuyo veredicto es menos flexible. Más interesante aún, esta herramienta es capaz de evaluar programas con algunos errores sintácticos y semánticos, y hasta sugerir a los estudiantes qué deben arreglar. Esto ayuda en el aprendizaje y mantiene contentos a los estudiantes. Los desarrolladores del sistema y los profesores aseguran que con la herramienta han incrementado la calidad de sus cursos de programación y reducido el esfuerzo.

CourseMaker es una plataforma desarrollada en Java por la Universidad de Nottingham que ha sido utilizada por varias instituciones sobre todo europeas. Realiza pruebas estáticas sobre las soluciones que incluyen análisis de estilo y detección de plagio. Permite que el número de intentos y el nivel de retroalimentación sean definidos por el autor de cada problema y registra las estadísticas de errores por alumno para posteriormente determinar los contenidos del curso que deben ser reforzados. Puede incluso recomendar la lectura de materiales del curso (López, 2011), (Higgins, 2003).

1.1.3 Plataformas para la enseñanza de la Programación

Para extender las prestaciones de SPOJ, recientemente el *Sphere Interest Project* anunció **SPOX**[13], una plataforma dedicada a la enseñanza de la Programación, desarrollada sobre el mismo motor del juez original.

Esta plataforma adjunta es gratis pero se encuentra en su versión beta. Permite disponer del conjunto de problemas del SPOJ. Durante esta investigación se probó la disponibilidad para simular un curso de programación de la UCI sin dificultad[14]. Los desarrolladores recomiendan su uso en los cursos introductorios. Permite crear cursos

[13] http://spox.spoj.pl
[14] http://uci-judge.spox.spoj.pl

y lecciones de distintos niveles de complejidad convenientemente separados en lecciones, así como restringir cuáles estudiantes tienen acceso a los cursos creados. La mayor dificultad de la plataforma es que implica contar con acceso internet y la limitación de contar solo con los problemas ya existentes. Por demás como mismo ocurre con SPOJ no implementan ningún mecanismo para la detección de plagio ni la orientación a los estudiantes.

Varias instituciones universitarias, entre ellas la UCI, utilizan la plataforma **Moodle**[15] como soporte en línea para el Proceso de Formación. En (Zhigang, 2011) se informa sobre la implementación de plugins para Moodle en función de incluir en la plataforma algunas actividades similares a las que operativamente muestran los jueces en línea existentes y el aporte enriquece la herramienta con la inclusión de un detector de plagio.

Aunque esta funcionalidad de juez en línea para la docencia queda enriquecida por varias funcionalidades que ya tiene incluida la plataforma, como la gestión de los profesores y grupos, se considera que las prestaciones que ofrecen los jueces en línea siguen siendo más ricas, en lo que a interfaces y utilidades se refiere. Sería conveniente que Moodle integrara la funcionalidad de realizar recomendaciones de actividades a realizar por los estudiantes según su trayectoria. De cualquier manera no deja de ser esta plataforma y sus nuevos *plugins* una variante sobre la cual investigar y mejorar en futuro cercano para su adecuación.

A partir de las demandas de los usuarios del UVA y de la necesidad de mejorar las competencias matemáticas, científicas y tecnológicas a nivel europeo en la educación secundaria y superior, surge el proyecto **EduJudge**[16], consistente en la integración del Juez en Línea de Valladolid al *e-learning* efectivo. EduJudge ha sido desarrollado y probado en entornos reales y consta de tres principales componentes:

- El UVA como un servidor de evaluación de las soluciones enviadas por los estudiantes. Esto incluye las mismas limitaciones en los lenguajes de programación.

[15] http://moodle.org.
[16] http://eduvalab.uva.es/tags/edujudge

- QUESTOURnament[17] como nueva actividad que permite el desarrollo de concursos en la plataforma Moodle, durante los cuales los estudiantes compiten entre sí tratando de resolver un conjunto de desafíos en un límite de tiempo. (Verdú, 2010)
- CrimsonHex como un repositorio especializado en problemas de programación (Leal, José, 2008).

YoungCoder[18] es otra solución innovadora para la enseñanza de la Programación, específicamente de los lenguajes C++ y Java, que oferta tutoriales y varios problemas de programación, incluyendo un juez en línea para evaluar las soluciones. Permite a los profesores preparar rápidamente sus lecciones y organizar exámenes o competencias para sus estudiantes. Según sus creadores, la plataforma fue diseñada para incrementar el desarrollo del aprendizaje en el área de los algoritmos y la programación, guiando al usuario paso a paso por varios aspectos intrínsecos del arte de programar. Puede ser empleada por profesores para enseñar a sus respectivos estudiantes o bien para el autoaprendizaje de aquellos usuarios que de manera individual le guste aprender algoritmos o lenguajes de programación desde el principio. Aunque es perfecta según la descripción de sus creadores, tiene el importante inconveniente de que no es gratis, no incluye detección de plagio ni orientación a los estudiantes. Puede accederse a prestaciones limitadas de esta plataforma mediante las credenciales (`tom_ajunc, internetpass`).

1.1.4 Resumen sobre revisión de las herramientas existentes

A partir del análisis realizado se procede a continuación a resumir las características encontradas en cada uno de las tres clasificaciones de herramientas evaluadoras estudiadas (ver tabla 1).

Tabla 1. Resumen de características de los jueces en línea revisados

Característica	Jueces en línea entornos competitivos	Jueces en línea entornos docentes	Plataformas enseñanza de la programación
1. Accesibilidad	Demandan de acceso a Internet (excepto el COJ	Demandan de acceso a Internet (excepto BOSS que	Demandan acceso a Internet

[17] http://itastdevserver.tel.uva.es/dev/moodle-projects
[18] www.youngcoder.eu

	que está disponible en intranet nacional)	puede ser descargado)	
2. *Detección de Plagio*	No incluyen	Solo BOSS incluye esta funcionalidad	No incluyen
3. *Motivación de la Competitividad*	Incluyen varios rankings, sistemas de puntuación estimulantes y gestión de competencias	No	No
4. *Orientación a los Usuarios*	Solo SPOJ y UVA de manera muy básica	No	No
5. *Gestión Docente*	No	Sí	Sí
6. *Posibilidades de Administración*	No	Parcialmente	Parcialmente
7. *Disponibilidad del Código Fuente*	solo del COJ	solo de BOSS	No
8. *Adecuación de los Problemas para la Docencia*	Algunos pocos se adecuan, pero sobre todo incluyen problemas del ACM-ICPC y redactados en idioma extranjero	Buen conjunto de problemas para incluir en la docencia, clasificados por asignaturas y redactados en varios idiomas, fundamentalmente inglés	Algunos. En general están soportados sobre los problemas de los jueces en línea competitivos.

Se consideran insuficientes las herramientas estudiadas sobre todo por la necesidad en general de acceso internet para su acceso, no todas retoman el enfoque del aprendizaje competitivo y fundamentalmente por no disponibilidad en casi todos los casos de funcionalidades para facilitar la orientación a los usuarios[19] ni para facilitar la detección de plagio.

1.2 Experiencias en la Docencia

La Universidad de Concepción, en Chile, utiliza una plataforma de evaluación automática que apoya el proceso de enseñanza/aprendizaje en cursos introductorios

[19] Desde este momento el concepto *usuario* queda ligado al concepto *estudiante* dado que serán los estudiantes los principales usuarios del sistema a desarrollar.

de programación de computadores para estudiantes de ingeniería. Esta plataforma utiliza mecanismos que combinan análisis estático/dinámico y aplican evaluación de comprensión/análisis en línea, permitiendo una retroalimentación personalizada a los alumnos. Los resultados obtenidos después de seis años de aplicación muestran que la evaluación automática afecta positivamente la motivación, el desempeño y la autoeficacia de los alumnos (López, 2011).

La Universidad Politécnica de Cataluña hace uso del sistema Jutge.org en sus cursos introductorios de Programación (Giménez, 2012) manteniendo orientadas varias actividades a los estudiantes sobre esta plataforma y basando todo el aprendizaje en el preciado conjunto de ejercicios que mantiene este sistema. Según los reportes, los estudiantes aprecian usar dicha herramienta.

También en la Universidad de Liverpool se tiene constancia del uso de un juez en línea como apoyo a los cursos relacionados con la Programación. A través de la correspondencia con uno de los profesores del curso *Principles of C and Memory Management*, se supo que el juez en línea que utilizan no es desarrollado en la propia universidad sino una de las versiones libres que son descargables en la red. El mismo profesor plantea que el uso de un sistema no propio le introduce limitación de varias funcionalidades deseables. El más sencillo y representativo ejemplo es que no tienen acceso a ver las soluciones de los estudiantes y se usa el sistema solo para que el estudiante reciba una preevaluación después de lo cual debe enviar mediante otro sistema las soluciones al profesor para la revisión manual (Guo, 2012).

Otras importantes evidencias relacionadas con el uso de los jueces en línea en la docencia, que permiten asegurarse de cuán efectivo, empleado y extendido ha sido el uso de estas herramientas pueden revisarse en (Kurnia, y otros, 2001), (Cheang, y otros, 2003), (Nghi, 2007), (Wen-xin, 2005), (Luo, 2008), (Montoya, y otros, 2009), (García, y otros, 2009), (Tiantian, 2011), (Zhigang, 2011) y (Mckinstry, 2012). En (Walker, y otros, 2005) se describe detalladamente la aplicación exitosa y las conclusiones de la experiencia de aplicación durante varios cursos del sistema OMEN en el Instituto Politécnico de Virginia.

Se tiene constancia de que en la Universidad de Matanzas (UMCC) varias asignaturas incluyendo los cursos optativos se apoyan en los recursos del Juez en Línea Caribeño para la evaluación y la preparación de sus estudiantes. Un trabajo relacionado con estas experiencias fue expuesto en el I Taller de Aprendizaje Colaborativo en el marco

de la Final Caribeña 2011 del ACM-ICPC. Además se han realizado varios intercambios al respecto entre los directivos del Movimiento de Programación Competitiva "Tomás López Jiménez" (MPC-TLJ) y los profesores de las asignaturas de programación en la UMCC fundamentalmente para lograr la clasificación por asignaturas de los problemas publicados en este juez en línea.

En la UCI, en la actual Facultad 4, fue empleado desde 2006 hasta el curso anterior el Juez en Línea de la Iniciativa Xtreme para las asignaturas IP y P2 fundamentalmente. El empleo fue bastante informal hasta el curso 2009-2010 debido a que el sistema no facilitaba la gestión del profesor sobre sus grupos al ser un juez en línea diseñado con fines de entrenamiento de concursantes.

1.3 Sistemas de Recomendación

Todos los estudiantes no tienen el mismo desempeño en el aprendizaje. En particular en las asignaturas de programación son comunes los estudiantes muy retrasados en el aprendizaje y también existen los muy avanzados. Cuando, como es típico de los jueces en línea, se tiene un numeroso conjunto de problemas, de tan distintos niveles de dificultad, de tantas asignaturas y tantos temas, puede resultar complicada la toma de decisión sobre ¿qué ejercicio hacer? o ¿qué ejercicio orientar? según sean los roles de estudiante o profesor, respectivamente.

Es aquí donde entran a jugar su papel los sistemas de recomendación. Estos utilizan la opinión de los miembros de una comunidad para ayudar a los individuos a identificar la información o los productos más relevantes o interesantes según sus necesidades actuales (Konstan, 2004). Constituyen una técnica de filtrado de información, la cual presenta distintos tipos de temas o ítems de información al usuario, basándose en la predicción del *ranking* o ponderación que este le daría a un ítem que el sistema aún no ha considerado; mediando, automatizando o soportando de esta forma, el proceso de realizar recomendaciones (Terveen, y otros, 2001).

Estos sistemas comienzan a emerger a mediados de los años 90 (Hill, 1995), como solución ante el desbordamiento de información que desde aquel entonces ya estaban ocasionando a nivel global las nuevas tecnologías de la información y las comunicaciones (TICs) y en particular la World Wide Web (www). La amplia utilización y aceptación de este nuevo paradigma ha devenido en que se pueda ya afirmar que se

esté abandonando la época de la información y se esté iniciando la época de la recomendación (Terveen, y otros, 2001).

1.3.1 El problema de la recomendación

El principio de funcionamiento de los sistemas de recomendación se resume en la formalización del problema de la recomendación (Resnick, 1994), el cual puede ser formulado de la siguiente forma:

Sea ***C*** un conjunto de usuarios, y ***S*** el conjunto de todos los ítems[20] posibles a ser recomendados, tales como libros, películas o restaurantes, pudiendo ser bien grande la cardinalidad de ambos conjuntos. Sea además ***u*** una función que mide la utilidad del ítem ***s*** para el usuario ***c***, o sea ***u*: *C x S* → *R***, donde ***R*** es un orden total (enteros no negativos o números reales en un determinado rango). Con estos elementos, se desea, para cada usuario $c \in C$, aquel ítem $s' \in S$ que maximice la utilidad al usuario (Adomavicius, y otros, 2005). Más formalmente:

$$\forall c \in C, \quad s'_c = argmax_{s \in S} u(c, s)$$

1.3.2 El modelo del proceso de recomendación

Dentro de la concepción de sistemas de recomendación, ocupan un papel central el individuo *buscador de recomendaciones*, el *proveedor de preferencias* y el *sistema recomendador* propiamente dicho. El sistema recomendador ofrece recomendaciones al buscador de recomendaciones incluso en ocasiones sin solicitud previa. Para la construcción de las recomendaciones, el sistema puede capturar de manera directa las preferencias específicas de los individuos, o puede inferirlas a través de preguntas indirectas. Una vez recopilados todos los datos sobre los usuarios, el sistema está listo para recomendar los ítems que el usuario probablemente prefiere, basándose en su perfil, el perfil de otros usuarios, los datos del proveedor de las posibles preferencias, entre otros criterios a tomar en cuenta, siendo utilizadas estas recomendaciones para seleccionar ítems del universo de alternativas.

En la concepción y diseño de un sistema recomendador suelen tenerse en cuenta la forma de representar las preferencias (Gemmis, 2009), la manipulación de los roles dentro del sistema y la comunicación usuario-usuario (Rodríguez, 2009), los algoritmos

[20] Particularmente, en el contexto de este trabajo el término *ítem* se refiere a un problema de los presentes en el juez en línea.

para manipular las recomendaciones (Vozalis, 2003) y la forma de interacción usuario-máquina a la hora de representar las recomendaciones (Swearingen, 2001).

1.3.3 Algoritmos en sistemas de recomendación

Con el fin de dar una solución eficaz y computacionalmente eficiente al problema de brindar recomendaciones que se correspondan con los intereses de un usuario determinado, se han utilizado diferentes enfoques que pueden ser agrupados en dos categorías fundamentales: los algoritmos basados en memoria y los basados en modelos.

1.3.3.1 Métodos basados en memoria

Los algoritmos basados en memoria son, en esencia, heurísticas que realizan predicciones de las preferencias de los usuarios por determinados ítems, basándose en la evaluación[21] previa de los ítems dada por todos los usuarios. Los datos normalmente son procesados en el instante en que son solicitadas las recomendaciones.

Generalmente emplean técnicas estadísticas para encontrar *vecinos* del usuario actual (aquellos con un similar historial de valoraciones respecto a los elementos). Determinados los *vecinos*, se combinan sus preferencias para generar una lista con los ***N*** elementos más recomendables para el usuario actual. Entre sus inconvenientes se encuentra la necesidad de disponer de un número mínimo de usuarios con un número mínimo de predicciones cada uno, incluyendo el usuario para el cual se pretende realizar la recomendación (Galán, 2007).

1.3.3.1.1 K–Nearest Neighbor

Los *algoritmos de vecinos más cercanos* o *k-nearest neighbor* están conformados por tres etapas fundamentales, que son la representación de los datos, la formación de vecinos y la generación de recomendaciones (Vozalis, 2003), siendo la más crítica la segunda de ellas, por poder afectar el rendimiento del sistema y la eficacia de las recomendaciones en los casos en que no se conciba de la manera más correcta.

Se han definido varias medidas de similitud de elementos para estos algoritmos con vistas a utilizarlos como métrica durante el proceso de formación de vecinos. La más

[21] Aunque es adecuado describir estos sistemas de manera general, particularmente, en el contexto de este trabajo los términos *evaluación* o *consideración* de un ítem se refieren a la resolución de un problema del juez en línea.

general de todas la constituye la *distancia euclidiana* la cual se define para toda estructura que pueda asumir la forma:

$$(a_1\ x\ , a_2\ x\ a_3\ x\ , \ldots , a_n\ x\ )$$

donde $a_r(x)$ denota el valor del atributo *r* de la instancia *x*. Para esta representación, la distancia entre dos instancias x_i y x_j, definida como $d(x_i,x_j)$ viene dada por:

$$d\ x_i,x_j\ = \sum_{r=1}^{n} (a_r\ x_i\ - (a_r\ x_j\ )^2$$

Otra de las funciones de semejanza más populares, con bastante utilización en el campo de los sistemas de recomendación, lo constituye la *similitud basada en el coseno* (Billsus, 2000), la que da una buena medida de similaridad de dos vectores en un espacio multidimensional, pudiendo este espacio describir características de usuarios o de ítems, tales como palabras clave, entre otras (Vélez, 2005). El cálculo de este parecido viene dado por el coseno del ángulo formado por la representación de estos dos vectores en el espacio *n*-dimensional, tal y como lo muestra la expresión:

$$sim_{ik} = cos_{ik} = \frac{r_i \cdot r_k}{r_i\ *\ r_k} = \frac{\sum_j r_{ij}}{\sqrt{\sum_j r_{ij}^2}} \frac{r_{kj}}{\sqrt{\sum_j r_{kj}^2}}$$

la cual permite calcular la proximidad entre dos usuarios u_i y u_k, siendo r_{ij} la evaluación dada por el usuario u_i al ítem i_j.

Una manera más sofisticada para establecer semejanzas dentro de los sistemas de recomendación son las similitudes basadas en correlación, las que no son más que medidas estadísticas que permiten establecer semejanzas entre dos elementos, dados determinados valores cuantitativos que los caracterizan. De particular efectividad resulta el *coeficiente de correlación de Pearson* (Resnick, 1994), (Su, y otros, 2009), que viene dado, para los usuarios a y b por:

$$w\ a,b\ = \frac{\sum_k (V_{a,k} - V_a)(V_{b,k} - V_b)}{\sqrt{\sum_k (V_{a,k} - V_a)^2}\ \sqrt{\sum_k (V_{b,k} - V_b)^2}}$$

donde $V_{i,k}$ es el voto emitido por el usuario *i* acerca del ítem *k*, V_i es la media de todos los votos para el usuario *i*, y *K* es el conjunto de elementos evaluados tanto por el usuario *a* como por el usuario *b*. Este método indica cuán linealmente correlacionados están los elementos de un vector con los del otro, lo que de manera gráfica se vería si se toma el plano cartesiano, en uno de los ejes se ubica los valores asociados a un

vector, en el otro eje los asociados al otro vector, y se determina la cercanía a la existencia de una recta que contenga todos los puntos de intersección de los valores de un eje con los de otro (Seagaran, 2005). En el caso de dar como resultado 1, esto se interpreta como que existe una correlación total, o en este caso relación de proporcionalidad directa total, entre un vector y otro; en el caso de obtenerse -1, esto indicaría que la relación es de proporcionalidad inversa total. En cambio, la obtención de un 0 como *coeficiente de Pearson* indica que la relación entre un vector y otro es nula o no existe.

Como una alternativa al coeficiente de correlación de Pearson, con mucha frecuencia se utiliza el *coeficiente de Jaccard-Tanimoto* por resultar más práctico en determinadas circunstancias. Este se centra en calcular la similitud entre dos conjuntos basándose en la cantidad de elementos comunes que poseen, y se define, siendo N_a la cantidad de elementos del conjunto A, N_b la cantidad de elementos del conjunto B, y N_c la cantidad de elementos de la intersección de A y B, de la siguiente forma:

$$T = \frac{N_c}{N_a + N_b - N_c}$$

Tras obtener *los vecinos más cercanos* a través de determinado criterio de similitud, sus datos asociados son utilizados para generar las recomendaciones para el usuario actual. Con dichos fines, la técnica más utilizada es la *Most-Frequent Item Recommendation*, consistente en llevar un conteo de la frecuencia en que cada uno de los ítems es evaluado o considerado por parte de los usuarios que están en el vecindario del usuario activo. Tras completar el listado de frecuencias de aparición de ítems, se procede a excluir aquellos ya evaluados o considerados por el usuario activo, ordenando los restantes de mayor a menor considerando esta frecuencia de aparición, siendo retornados los *N* primeros elementos de este listado final en calidad de recomendaciones para dicho usuario.

Otra variante a utilizar para la generación de las recomendaciones es la *Association Rule-based Recommendation*. En el dominio de este método, considerando la existencia de *n* ítems $I = \{i_1, i_2, \dots, i_n\}$, se define una transacción $T \subseteq I$ como un conjunto de ítems que son evaluados o considerados juntos, en una misma interacción del usuario con el sistema. Una regla de asociación entre dos conjuntos de elementos I_x y I_y, donde $I_x, I_y \subseteq I$ y $I_x \cap I_y = \emptyset$, establece que si están presentes los ítems I_x en

una transacción T, entonces existe una gran probabilidad de que los ítems de I_y también podrían estar presentes en T. Esta regla de asociación normalmente en lenguaje formal se denota como $I_x \rightarrow I_y$.

Una vez obtenidas las reglas de asociación, es posible la generación de recomendaciones dirigidas a un usuario en específico. Con este propósito, para cada uno de los usuarios que conforman el vecindario de este, es creada una transacción conteniendo todos los ítems valorados o considerados en el pasado. Teniendo como entrada estas transacciones, se define un algoritmo de descubrimiento de reglas de asociación para encontrar aquellas que sean soportadas por determinados usuarios, esto es, que todos los ítems presentes en la parte izquierda de la regla estén presentes en la transacción del usuario correspondiente. Posteriormente, se forma un listado de elementos no repetidos conformado por los ítems de la parte derecha de las reglas aplicables, que aún no hayan sido considerados por el usuario activo, entregándosele a este en calidad de recomendaciones finales, tras ser ordenado por el grado de confianza de la regla que lo generó. Para los ítems que se generan por más de una regla, se selecciona para el ordenamiento la regla de mayor confianza.

1.3.3.2 Métodos basados en modelos

En contraste con los algoritmos basados en memoria, los basados en modelos utilizan la colección de evaluaciones o de ítems considerados, con el objetivo de *aprender* un modelo del usuario, el que es entonces usado para mostrar predicciones de evaluación o mostrar recomendaciones. La implementación efectiva de este aprendizaje usualmente implica la utilización de técnicas estadísticas y de aprendizaje automático dentro del sistema de recomendación.

Empíricamente se ha demostrado el incremento del rendimiento en términos generales de un sistema de recomendación con la utilización de métodos basados en el modelo, aunque algunos autores (Breese, 1998) señalan que pueden sufrir problemas de desbordamiento de memoria como precio a pagar ante el intento de producir recomendaciones de alta calidad.

1.3.3.2.1 Recomendación bayesiana

Uno de los modelos más utilizados para reflejar las preferencias de los usuarios lo constituyen las redes bayesianas (Pearl, 1988). Una red bayesiana es un grafo acíclico

dirigido en el que los nodos son variables y los arcos representan relaciones de influencia causal entre ellos. La información cuantitativa a suministrar para cargar la red está dada por las probabilidades a priori de los nodos que no tienen padres, y la probabilidad condicionada de los nodos con padres, tal como se muestra en la figura 2.

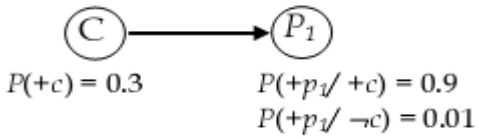


Figura 2. Red bayesiana con parámetros

Las redes bayesianas son aplicables al campo de los sistemas de recomendación, pudiendo servir cada uno de sus nodos como representación de los ítems a seleccionar, con la respectiva probabilidad por parte de estos, de ser seleccionado. Las probabilidades condicionales, por su parte, representarían la probabilidad de seleccionar un ítem determinado, conociéndose previamente la selección de un ítem anterior que guarde intrínseca relación con este.

1.3.3.2.2 Clustering

Las técnicas de *clustering* trabajan a través de la identificación de grupos o *clusters* con usuarios que evidencien tener semejantes intereses. Una vez que los *clusters* son creados, las predicciones de cualquiera de sus individuos miembros pueden ser hechas promediando las opiniones del resto de los usuarios en el *cluster* correspondiente.

Es importante resaltar que el principal objetivo de este enfoque no es el de ofrecer sugerencias, sino el de reducir la dimensión del espacio contenedor de todos los datos de los usuarios e ítems, particionándolos en varios espacios más pequeños con menor cantidad de ítems, menor número de preferencias pre-almacenadas y menor cantidad de usuarios. Entre los algoritmos más populares para la implementación del clustering de manera general, están el *k-means clustering* (Bock, 2007) y el cluster jerárquico (Seagaran, 2005).

1.3.3.2.3 Horting

Horting es una técnica basada en grafos, definida en (Aggarwal, 1999) específicamente para su aplicación en sistemas de recomendación. En esta, los nodos del grafo representan los usuarios y las aristas el grado de similitud entre un usuario y otro.

Difiere del método de los vecinos más cercanos al considerar, para la elaboración de las recomendaciones, las relaciones transitivas entre los usuarios como un factor importante a la hora de establecer la proximidad.

La idea central de este enfoque reside en predecir la evaluación del ítem i por parte del usuario j, siendo esta computada como el promedio de una serie de pesos calculados a partir de los correspondientes caminos dirigidos que unen al usuario en cuestión con un conjunto de usuarios ya previamente evaluadores de i. Desde el punto de vista del rendimiento, se considera que esta técnica es rápida, escalable, precisa y además, con una modesta curva de aprendizaje.

1.3.3.3 Limitaciones de los algoritmos en sistemas de recomendación

El hecho de trabajar en sentido general con información con alto grado de incertidumbre ocasiona que los métodos para establecer recomendaciones se encuentren con limitaciones que pueden dificultar la consecución de su objetivo final.

1.3.3.3.1 Arranque en frío

El *arranque en frío* constituye una de las cuestiones críticas a resolver al inicio del despliegue de todo sistema de recomendación, independientemente de la línea que se haya seguido en su desarrollo. Este problema hace referencia a las situaciones en las que se dispone solamente de un reducido número de evaluaciones o consideraciones para realizar las recomendaciones solicitadas, pudiendo ocurrir específicamente bajo tres escenarios diferentes:

- Nuevo usuario: Cuando un nuevo usuario se registra en el sistema de recomendación, existen muy pocos datos disponibles para definir con cierta precisión su perfil.
- Nuevo ítem: Cuando un nuevo ítem es adicionado al catálogo, inicialmente no presenta evaluaciones.
- Nuevo sistema: Cuando se pone a funcionar una nueva instancia de un sistema de recomendación, el número promedio de evaluaciones o consideraciones por usuario y por ítem es bajo, lo que puede afectar significativamente la calidad de las recomendaciones.

1.3.3.3.2 Dispersión de los datos

En los sistemas de recomendación el número de evaluaciones reales que se posee acerca de un ítem específico queda muy por debajo del número mínimo que se requiere para hacer recomendaciones efectivas, haciéndose particularmente crítico para aquellos usuarios con preferencias alejadas de las de la mayoría. Esto implica que la predicción efectiva a partir de un número reducido de ejemplos juegue un papel clave para el éxito del sistema.

1.3.3.3.3 Sobre-especialización

La sobre-especialización en sistemas de recomendación aparece cuando solamente se recomiendan ítems que se corresponden fuertemente con el perfil del usuario, limitando a estos a recibir únicamente sugerencias de elementos similares a aquellos que ya ha evaluado o considerado. Con vistas a garantizar cierta diversidad en cuanto a las recomendaciones brindadas a determinado usuario, en muchos sistemas de recomendación se ha dado cabida a la aleatoricidad, a través de técnicas como la de los algoritmos genéticos.

1.3.4 Herramientas recomendadoras en los jueces en línea

Los sistemas de recomendación han encontrado aplicación dentro de los jueces en línea, los cuales tampoco han estado exentos del sobredimensionamiento de la información. Con objetivo de facilitar la utilización de los jueces en línea se han concebido aplicaciones con características recomendadoras, específicamente para guiar a los usuarios a la hora de seleccionar los problemas a resolver. Entre las más representativas están el ACM Problem Grading[22], el Next2Solve[23] y el SPOJ Problems Classification[24]. A continuación se hace un breve análisis de cada una de ellas.

1.3.4.1 ACM Problem Grading y Next2Solve

El ACM Problem Grading es una herramienta implementada en apoyo al UVA por Sebastian Urbaniak que se limita a proporcionar un listado de problemas a resolver, el cual puede ser ordenado por diferentes criterios entre los que se pueden mencionar la cantidad de soluciones aceptadas para cada ejercicio, el porciento de soluciones correctas con respecto al total de soluciones y un *score* de simplicidad que se calcula

[22] http://www.angelfire.com/on4/surbaniak/acm_grading_score.htm
[23] http://shygypsy.com/acm/cgi-bin/next2solve.pl
[24] http://vnoi.info/index.php?option=com_voj&task=classify&site=spoj

tomando como base los dos anteriores criterios. No se enfoca en un usuario en particular, suponiendo que los problemas más sencillos a resolver para todos son aquellos que más soluciones correctas tengan. Los criterios que utiliza son completamente estáticos, siendo diametralmente opuestos a las actuales tendencias dentro de la web.

Por su parte, introduciendo un ligero aporte y desarrollado también para el UVA, Next2Solve es una herramienta basada en el ACM Problem Grading. Su mejor enfoque radica en que esta solicita la entrada del identificador de usuario para generar una sublista de la lista generada por la aplicación de Urbaniak, en la que sólo aparecen los problemas que el usuario aún no ha resuelto.

1.3.4.2 SPOJ Problems Classification

SPOJ Problems Classification es una herramienta de recomendación que utiliza un enfoque distinto al presentado en las dos aplicaciones anteriores. En vez de concentrarse en el perfil de los usuarios, se centra en los ejercicios, definiendo para cada uno de ellos una serie de datos o etiquetas que luego son utilizadas para agruparlos por categorías. Utilizando estas categorías y a través de opciones de filtrado, los usuarios pueden obtener propuestas de problemas a resolver, basándose en criterios de búsqueda, que coinciden generalmente con los valores de las etiquetas. Unido a esto, los propios usuarios también pueden agregarle nuevas etiquetas a los problemas. De esta manera se aprovecha la inteligencia colectiva y la experiencia de los distintos usuarios para orientar sobre los problemas a resolver.

Uno de los principales inconvenientes que tiene esta propuesta es que permite el acceso a todas las funcionalidades de manera anónima, lo que implica que la información que esta posee carezca de fiabilidad. Además, el no gestionar de forma individual la información de cada uno de los usuarios implica que no se pueda indicar un paquete específico y personalizado de actividades a realizar, imposibilitando un mejor aprovechamiento de la misma.

1.4 Detección de Plagio

El plagio es una violación grave que puede ser reflejada en diferentes esferas de la sociedad. La Real Academia Española[25] al definir la palabra “plagio”, nos remite

[25] http://www.rae.es/rae.html

directamente a la acción y efecto de "plagiar", entendiendo esta última como "copiar en lo sustancial obras ajenas dándolas como propias".

El auge de Internet facilita plagiar el trabajo de otros (Cedeño, 2008). Es común que muchos sitios copien y peguen información sin la autorización correspondiente, engañando a sus lectores al presentar los datos como propios; de igual manera las personas copian y pegan de sitios e investigaciones científicas sin referenciar la fuente; a estas acciones se les conoce como plagio de texto. Un caso particular lo constituye el plagio de algoritmos y códigos fuentes de programas informáticos. Sobre todo en el ámbito académico el plagio es un comportamiento inaceptable y afortunadamente existen varias herramientas informáticas que permiten detectarlo.

El plagio también está presente en los jueces en línea y es una de las principales dificultades a mitigar en un juez que se pretenda utilizar para el apoyo docente (Joy, y otros, 1999).

En SPOJ han abierto un hilo de fórum[26] solo para discutir el tema del plagio ya que preocupa a los usuarios que el ranking no sea lo suficientemente representativo. Particularmente en la UCI, en los jueces en línea usados para entrenamiento de concursantes ACM-ICPC se han identificado usuarios que envían soluciones de otros, dándolas como suyas. En septiembre de 2009 en el CPAV fueron detectados 948 posibles casos de plagio correspondientes a 374 usuarios. En una segunda corrida de la aplicación en mayo de 2010 fueron 271 los casos detectados involucrando a 211 usuarios.

La necesidad de detectar el plagio se ha tornado primordial para garantizar la fiabilidad de los resultados en las competencias y las estadísticas; además esta práctica antiacadémica y antiética atentaría contra el aprovechamiento que pueden hacer los estudiantes de una aplicación tan útil para las asignaturas de programación.

Los sistemas para la detección de plagio son herramientas diseñadas para buscar similitudes entre muestras con el fin de encontrar fragmentos plagiados total o parcialmente. Surgen a mediados de los años 70 como posible solución ante la inaceptable tendencia presente en investigaciones científicas, obras, algoritmos y códigos de programación.

[26] http://www.spoj.pl/forum/viewtopic.php?f=29&t=4465

Un juez en línea ajustado a las necesidades de la docencia

El problema de la detección automática de plagio puede considerarse un problema de búsqueda y/o clasificación (Cedeño, 2008). Los elementos de referencia ***D*** y el elemento sospechoso ***S*** son suficientes para describir el planteamiento de la detección de plagio:

Sean ***S*** un elemento sospechoso y ***D*** un conjunto de elementos de referencia, el objetivo de la detección automática de plagio es encontrar aquel elemento $d \in D$ que haya sido utilizado como fuente para obtener el elemento ***S***, el cual presumiblemente es un caso de plagio. Dicha búsqueda puede llevarse a un nivel más específico: Sea $S_i \in S$ un fragmento plagiado, el objetivo es encontrar aquel fragmento $d_j \in d$ tal que $\mathbf{d}_j$ es la fuente del fragmento plagiado S_i.

1.4.1 Algoritmos para la detección de plagio en código fuente

Los algoritmos generalmente utilizados en la detección de plagio en código fuente utilizan desde criterios muy sencillos basados en métricas de software como el conteo de variables, hasta complejas funciones de codificación y comparación de porciones de texto. Se identifican tres tendencias fundamentales encabezadas por el uso de los algoritmos Running-Karp-Rabbin y Greedy-String-Tiling, así como el empleo de técnicas basadas en funciones *hash*, para generar claves que representan de manera casi unívoca a un documento, texto o archivo (Barrón Cedeño, y otros, 2011) (Schleimer, y otros, 2003).

1.4.1.1 Algoritmo Running-Karp-Rabbin

La base del algoritmo Running-Karp-Rabbin es el *hash Karp-Rabin* utilizado para crear claves en porciones no comparadas del código fuente, las coincidencias son buscadas sobre dichas claves y el tiempo de ejecución es optimizado mediante el uso de tablas *hash*. Contrario a realizar una búsqueda exhaustiva de coincidencias, se pueden obtener de la tabla las posiciones iniciales de todas las porciones de código de longitud **S** procesadas en orden descendente para intentar encontrar primero coincidencias mayores que tengan el mismo valor asignado por la función *Karp-Rabin*. Luego de encontrar una coincidencia de la porción de código y la clave a ella asignada, se continúa buscando nuevos elementos hasta que el algoritmo se detiene debido a una diferencia o el fin de la entrada (Crochemore, y otros, 1997), (Wise, 1996).

1.4.1.2 Algoritmo Greedy-String-Tiling

El algoritmo Greedy-String-Tiling (ver Figura 3) intenta encontrar similitudes de código comenzando por valores extremos y disminuyendo la longitud hasta un mínimo elegido. En cada ciclo de búsqueda se analizan las porciones de tamaño **S** que coincidan en ambos códigos, las mismas son añadidas a la lista de coincidencias en una estructura del tipo: *(<posición fuente>, <posición destino>, <longitud>)* reflejando una similitud encontrada comenzando en la *<posición fuente>* del primer código y en la *<posición destino>* del segundo, con una longitud determinada por el tercer campo. La porción de código es marcada para no tenerla en cuenta en futuras iteraciones y se continúa el proceso con valores **S** menores hasta llegar al umbral inferior de la detección (Prechelt, y otros, 2000).

```
Greedy-String-Tiling(String A, String B) {
    tiles = {};
    do {
        maxmatch = MinimumMatchLength;
        matches = {};
        Forall unmarked tokens A_a in A {
            Forall unmarked tokens B_b in B {
                j = 0;
                while (A_{a+j} == B_{b+j} &&
                       unmarked(A_{a+j}) && unmarked(B_{b+j}))
                    j++;
                if (j == maxmatch)
                    matches = matches ⊕ match(a, b, j);
                else if (j > maxmatch) {
                    matches = {match(a, b, j)};
                    maxmatch = j;
                }
            }
        }
        Forall match(a, b, maxmatch) ∈ matches {
            For j = 0 ... (maxmatch − 1) {
                mark(A_{a+j});
                mark(B_{b+j});
            }
            tiles = tiles ∪ match(a, b, maxmatch);
        }
    } while (maxmatch > MinimumMatchLength);
    return tiles;
}
```

Figura 3. Algoritmo Greedy-String-Tiling

1.4.1.3 Algoritmos basados en funciones Hash

El termino *hash* no es más que la codificación de un texto en un valor acotado, mediante una función matemática. El objetivo principal de este procedimiento es aprovechar las capacidades de direccionamiento directo de estructuras como los arreglos, ya que, gracias a la conversión de un texto en su correspondiente clave numérica, esta puede ser utilizada como índice del arreglo y realizar operaciones de búsqueda en orden constante.

En la detección de plagio, el uso de funciones hash para convertir porciones del código en una clave numérica, tiene como principal objetivo ganar en velocidad de comparación. El funcionamiento de estos algoritmos es bastante simple y sin alejarse mucho de la comparación realizada por otros algoritmos para la detección de plagio, pero con una complejidad computacional menor (Kazim, 2008).

1.4.2 Herramientas para la detección de plagio en código fuente

Esta sección se revisa el plagio de códigos fuentes pues es el punto que concierne a los jueces en línea, al ser por su naturaleza almacenes clásicos de soluciones algorítmicas. Disponible en Internet aparece un importante número de herramientas para la detección de este tipo de plagio, cuyos más antiguos precedentes son descritos en (Ottenstein, 1976). Se destacan entre estas YAP3[27], MOSS[28] y Jplag[29].

1.4.2.1 YAP3

YAP3, siglas en inglés de ***Y**et **A**nother **P**lague*, es un programa desarrollado en la Universidad de Sydney en el año 1996. Este basa su funcionamiento en utilidades de la consola Unix, fue programado en Perl y los scripts pueden ser bajados desde la web del autor[30]. Es una herramienta gratuita con fines no comerciales disponible para los lenguajes C, Pascal y Lisp, para cuyo desarrollo se aplicaron los algoritmos Running-Karp-Rabbin y Greedy-String-Tiling. Esta herramienta comienza su procesamiento del código eliminando los comentarios y constantes de cadena, convierte todos los textos a letras minúsculas y reemplaza los sinónimos por una palabra común para simplificar el engaño basado en remplazos de este tipo. Su estrategia para evadir los reordenamientos del código se fundamenta en procesar las funciones de acuerdo a su

[27] http://luggage.bcs.uwa.edu.au/~michaelw/YAP.html
[28] http://theory.stanford.edu/~aiken/moss/
[29] https://www.ipd.uni-karlsruhe.de/jplag/
[30] http://www.pam1.bcs.uwa.edu.au/~michaelw

orden de llamada y sus principales desventajas vienen dadas por la rústica representación de los resultados y los pocos lenguajes de programación para los que está disponible (Wise, 1996).

1.4.2.2 MOSS

MOSS, siglas en inglés de ***M**easure **O**f **S**oftware **S**imilarity,* es un servicio de Internet desarrollado en el año 1994 por la Universidad de Stanford. Funciona sometiendo a su consideración un grupo de ficheros y responde con un conjunto de páginas HTML detallando las similitudes detectadas. Es una herramienta gratuita aunque requiere obtener una cuenta vía correo electrónico del propietario. Se encuentra disponible para más de 20 lenguajes de programación incluidos C, C++, Java, C# y Python (Zhigang, 2011) (Aiken, 1994). El principal inconveniente de usar MOSS es que funciona como un servicio cuyo uso implica dependencia de Internet.

Basa su funcionamiento en la comparación de huellas digitales de los documentos. Divide los conjuntos de huellas en porciones basándose en algoritmos n-gramas[31], a las cuales se adjunta información estructural como el número de la página referenciada. Inicialmente el algoritmo crea un índice a distintos lugares dentro de los documentos, identificados con su respectiva huella, para luego buscar las coincidencias con la huella distintiva del documento (Schleimer, y otros, 2003).

1.4.2.3 JPlag

JPlag desarrollado en Alemania en la Universidad de Karlsruhe tiene su nombre de una mezcla entre los prefijos **J** de Java y **Plag** de *Plagiarism*. Es un sistema para la detección de plagio en conjuntos de códigos fuentes que se encuentra en forma de servicio web desde el año 1996. Su uso es gratuito aunque limitado por la creación de una cuenta. Toma como entrada un conjunto de programas y para cada par computa las regiones similares, generando un reporte en forma de página HTML. Se basa en un algoritmo similar al usado por YAP3 pero con algunas optimizaciones para mejorar su eficiencia. Trabaja convirtiendo el código fuente en *tokens* para luego tratar de completar su estructura con subcadenas de estos símbolos correspondientes al programa con el que se está comparando. El porciento de plagio es deducido a partir

[31] La **n-grama** es una representación redundante de texto que consiste en fragmentos solapados, ya sea a nivel de carácter o de palabra, cuya longitud es **n**. Las 3-gramas de "ejemplo" son [eje, jem, emp, mpl, plo] y los 2-gramas a nivel palabra de "este es solo un ejemplo" son [este es, es solo, solo un, un ejemplo] (Barrón Cedeño, y otros, 2011).

de la cantidad de subcadenas que puedan completarse. Su principales desventajas, además de la dependencia que implica de Internet, son que la cuenta necesaria para su uso está sujeta a la aceptación del propietario y el hecho de detectar plagio solo para cuatro lenguajes de programación (Prechelt, y otros, 2000).

1.5 Conclusiones

En el presente capítulo fueron revisados los principales jueces en línea del mundo, algunos en particular aplicados en contextos docentes. Fueron revisadas, además, algunas de las plataformas educativas más importantes relacionadas con la enseñanza de la Programación. Aunque varias de estas herramientas y entornos son excelentes candidatas para su eso en la docencia, existen algunos aspectos que exhortan a la implementación de un sistema propio:

- Las dificultades de conectividad en Cuba podrían afectar la calidad de los cursos impartidos al depender del acceso a Internet. Un trabajo futuro podría ser extender el uso de los jueces en línea a la docencia universitaria en todo el país y para eso sería importante tener presente desde los inicios de este trabajo las prestaciones de conectividad existentes en la red del Ministerio de Educación Superior (MES), la cual enlaza a todas las universidades a nivel nacional.
- Se considera muy importante que la autenticación en un juez en línea académico se realice mediante los usuarios del dominio de la institución, teniendo en cuenta de que sobre este sistema incluso podrán realizarse exámenes y evaluaciones decisivas para la promoción de los estudiantes, además de que esto permitirá integrar en un futuro otros servicios con el sistema.
- Los problemas y las interfaces de la mayoría de las aplicaciones están en idioma inglés algo que no sería deseable en algunas circunstancias. Además muchas de las plataformas no permiten la inclusión de nuevos ejercicios.
- El estudio describe una tendencia a nivel mundial de que los jueces en línea concebidos para entrenamiento de concursantes se integren a los perfiles educativos. En el caso de la UCI, donde se desarrolla el COJ (a cuyo equipo de desarrollo pertenece el autor de este trabajo) se considera conveniente el desarrollo de una plataforma educativa usando la misma arquitectura y tecnologías.

Un juez en línea ajustado a las necesidades de la docencia

Siendo una característica común de los jueces en línea el hecho de contar con un creciente conjunto de problemas a resolver, se considera necesario incluir un subsistema que permita orientar a los estudiantes que usen este tipo de herramientas en cuanto a qué problema tienen más oportunidad de resolver. A partir del estudio realizado se detectó que existen deficiencias en los jueces en línea para realizar recomendaciones a los usuarios. Dado esta problemática se decide concebir un sistema recomendador que tenga en cuenta el perfil de los usuarios y se base en la información almacenada en el juez en línea para determinar recomendaciones precisas y efectivas.

Se analizaron los principales enfoques algorítmicos utilizados para dar solución efectiva al problema de la recomendación, definiéndose claramente dos filosofías distintas para enfrentar este: los algoritmos basados en memoria y los algoritmos basados en modelo. Como parte de los enfoques se analizaron los métodos de vecinos más cercanos y reglas de asociación, asociados al primero; y las redes bayesianas, el clustering y el horting, asociados al segundo.

Se determina también la importancia de incluir un detector de plagio teniendo en cuenta que en otros ambientes docentes ha sido altamente necesario, además de que se ha comprobado la existencia de plagio en nuestros propios jueces en línea aún sin estar insertados en la docencia, contexto más propenso para ello.

Tras revisar las principales limitaciones de las herramientas existentes para detección de plagio, conviene desarrollar un sistema de este tipo. Para soporte de la detección de plagio se revisaron los algoritmos Greedy-String-Tiling y Karp-Rabin dos de los más eficientes y probados en otras herramientas de este tipo.

Capítulo 2. Propuesta de Solución

El Juez en Línea Académico consistirá en una aplicación web que tendrá como propósito principal apoyar la enseñanza de la programación. Tendrá varias funcionalidades similares a los varios jueces en línea existentes en el mundo dedicados al entrenamiento de concursantes ACM-ICPC e incluirá además otros elementos que lo adecuen para su introducción en el proceso docente.

Atendiendo fundamentalmente a las necesidades de incorporar cuanto antes las facilidades que se desprenderán de incluir el juez en línea en la docencia UCI, se describen en este capítulo las prestaciones y/o características que son deseables en la herramienta para posibilitar su inclusión en la docencia universitaria, para aplicarlo en cualquiera de las actividades relacionadas con la evaluación, el seguimiento, la orientación, la valoración y la auto superación de los estudiantes.

Para su ajuste a la docencia un juez en línea debe incluir:

- **Sistema de detección de plagio**. El módulo anti plagio debe ser adaptable a la incorporación de nuevos y/o varios algoritmos para la detección de similitud.
- **Clasificación de los problemas propuestos**. Se propone categorizar los problemas por asignaturas de la carrera y temas correspondientes a esas asignaturas; esos elementos permitirían realizar una selección de problemas para un año determinado. Se incluyen en la clasificación las técnicas de diseño de algoritmos a emplear en la resolución en caso de estar determinado.
- **Filtrado de problemas.** Los estudiantes y profesores podrán determinar los problemas que corresponden a una asignatura dada o a un tema determinado. Esto facilitará el autoestudio y animará sobre todo a los colectivos de asignaturas a incluir problemas relacionados con las materias que imparten, facilitando en consecuencia el trabajo interdisciplinario entre la Disciplina de Programación y las demás.
- **Recursos de apoyo al aprendizaje**. Se propone incluir en el juez en línea ejemplos de soluciones a algunos problemas sencillos incluidos entre los posibles a resolver, así como materiales relacionados con la operatividad en el sistema. Teniendo en cuenta que el Entorno Virtual de Aprendizaje es la plataforma

utilizada en la Universidad para estas cuestiones sería adecuado que, en caso de ser necesario indicar bibliografía a consultar o materiales semejantes, solo se incluyan enlaces a estos en la Plataforma.

- **Foro de discusión y mensajería**. Los nuevos esquemas de aprendizaje hacen protagonistas de los sistemas de e-learning a los foros de discusión y los sistemas de mensajería, recursos de la web 2.0 que permiten materializar el concepto del aprendizaje colaborativo. Mediante el foro, y más aún si se activa un hilo de foro por cada problema en particular, se pueden comentar y aclarar las dudas existentes para la resolución de algún ejercicio, tanto profesor-estudiante como estudiante-estudiante.
- **Historial de envíos de soluciones al sistema**. Es muy útil para hacer de manera asíncrona el seguimiento, control y evaluación del aprendizaje de cada estudiante, tener almacenado todo lo que ha hecho, mal o bien, cuándo, desde dónde, etc. El profesor deberá tener acceso a ver las soluciones enviadas y podrá realizar señalamientos a cada estudiante mediante el fórum o la mensajería interna.
- **Planificación de actividades con tiempo límite de resolución**. Una de las actividades más comunes realizadas en los jueces en línea son las competencias. Una competencia tiene un momento de inicio, tiempo de duración y problemas a resolver en ese tiempo. Esta característica se mantendrá en el juez en académico permitiendo a los profesores realizar actividades de laboratorio, exámenes, tareas extraclase en un tiempo determinado. El uso de esta práctica estaría directamente sincronizado con los elementos de aprendizaje competitivo y podría incentivar mucho la preparación de los estudiantes.
- **Soporte para los lenguajes usados en la docencia**. Queda claro que la posible utilidad de un sistema como este depende de que los lenguajes utilizados en los distintos cursos sean soportados por el sistema. En el caso de la UCI habría que incluir principalmente Java, C++, C#, Python, PHP, Ensamblador y Prolog.
- **Estadísticas a petición del colectivo de profesores y estudiantes**. La existencia de un historial de sumisiones permitirá la recuperación de información interesante sobre el aprendizaje de los estudiantes y las principales dificultades existentes. Reportes de este tipo incluidos en el juez permitirán constantemente la mejora de la asignatura.

- **Gestión de roles**. Como mismo ocurre en el EVA y otras plataformas educativas el conjunto de interfaces y funcionalidades disponibles para un usuario depende de su función en el proceso. Se identifican inicialmente los roles de Estudiante, Profesor, Profesor Editor y Juez de Plagio.
- **Gestión del proceso docente**. El sistema deberá suministrar o permitir incluir la información de qué usuarios son profesores y cuáles estudiantes. Se controlará qué grupos de clases son atendidos por cada profesor y en qué asignatura. Se tendrá al tanto la composición de los grupos. El manejo de toda esta información sustentará varias de las prestaciones de la herramienta.
- **Facilidad para la revisión de soluciones**. Teniendo en cuenta el flujo de procesos de funcionamiento de un juez en línea, detallado anteriormente, conociendo que para cada problema y sus datos de prueba las soluciones de los estudiantes generarán una salida (correcta o no), será deseable que el profesor tenga acceso a revisar esos ficheros salida pues puede señalar rápidamente el tipo de error que se esté cometiendo y facilitará la retroalimentación.
- **Mecanismo recomendador de problemas**. Cuando el conjunto de problemas aumente el estudiante necesitará asesoramiento de sus profesores para emprender la resolución de un nuevo reto. Incluso para un profesor experimentado esta decisión puede no ser suficientemente objetiva por lo que es deseable aplicar técnicas para apoyar estas orientaciones.

Este es un resumen de las características que marcan la separación entre un juez en línea docente con respecto a los demás jueces para entrenamiento de concursantes ya descritos. De cualquier manera el Juez en Línea Académico incluirá varias de las prestaciones estándares y otras que a partir de su puesta a punto en la docencia se irán necesitando para la gestión de los procesos.

2.1 COJ como punto de partida para el desarrollo

Teniendo en cuenta la tendencia mundial identificada el primer capítulo de esta tesis, de convertir o incluir los jueces en línea creados para entrenamiento de concursantes en jueces en línea con un concepto más pedagógico dirigido a la enseñanza de la programación a nivel de cursos en distintas escuelas, se consideró partir para el

desarrollo del Juez en Línea Académico del actual desarrollo del Juez en Línea del Caribe (COJ[32]).

El COJ ha estado disponible desde Junio de 2010 en Internet como herramienta para el entrenamiento de varios concursantes de todo El Caribe. El desempeño de este sistema ha sido adecuado y reconocido por varias figuras conocedoras del tema a nivel mundial, como Miguel Revilla uno de los desarrolladores del UVA en la Universidad de Valladolid. Varias de las estadísticas de este juez en línea, accesibles en Statistics for coj.uci.cu demuestran su cargado y correcto funcionamiento durante 2012 y constituyen créditos favorables para comenzar un desarrollo a partir de dicha herramienta..

Es destacable para el COJ su disponibilidad en la intranet del MES, lo que lo hace accesible para la mayoría de las instituciones cubanas. De hecho las tecnologías y metodologías utilizadas en su desarrollo estuvieron intencionadas a crear un sistema ligero que no dificultara la interacción con el sistema por parte de las universidades con dificultades de conectividad.

Varias de las funcionalidades necesarias para el Juez en Línea Académico ya están presentes en el COJ y tomarlo como punto de partida permitirá concentrarse en el desarrollo de las funcionalidades que son realmente novedosas y que aportan significativamente a una herramienta de este tipo para ser incluida en la docencia.

El autor de este trabajo es desarrollador del COJ y entiende, ratificado por los demás miembros del equipo de desarrollo[33], que por razones de recursos, mantenibilidad y soporte es recomendable que estos dos jueces en línea compartan la misma arquitectura y el mismo equipo de desarrollo.

2.2 Concepción del módulo recomendador

El subsistema recomendador desarrollado se centra fundamentalmente en sugerir problemas a resolver, basándose en la trayectoria y el perfil de los usuarios. En esta sección se expone la propuesta del perfil algorítmico del sistema desarrollado.

2.2.1 Sistema de recomendación para jueces en línea de programación

Aunque el módulo de recomendación desarrollado se incluye en el Juez en Línea Académico como una funcionalidad más, accesible por todos los usuarios una vez

[32] http://coj.uci.cu
[33] http://coj.uci.cu/general/about.xhtml#about.18

autenticados, este puede considerarse como un sistema independiente. Ha sido desarrollado de manera que su adaptación a otro juez en línea no implica muchas dificultades. Este actúa como punto intermedio entre el usuario y el juez en línea (ver Figura 4). A la hora de solucionar ejercicios, en lugar de dirigirse directamente al juez en línea, el usuario puede interactuar con el recomendador <1>, el que tendría almacenado previamente su perfil, a partir de la información almacenada del desempeño de este y los demás usuarios en el juez en línea <2>. A partir de la información asociada al usuario, relacionada con nuevas soluciones o intentos de solución a ejercicios <3>, el recomendador sugiere los problemas de mayor certeza de éxito ante intentos de solución <4>, procediendo posteriormente el referido usuario a su lectura <5>. Este proceso se comporta de manera cíclica, realizándose cada vez que el usuario se conecta a la aplicación o solicita orientación sobre un problema a resolver.

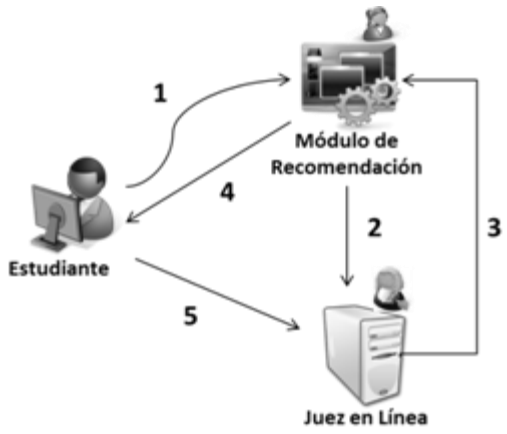


Figura 4. Sistema de recomendación

Para establecer una correcta correspondencia entre los posibles problemas a resolver y los usuarios, es necesario asociarle a ambos una serie de metadatos que permitirán identificarlos dentro de un conjunto de semejantes. En el caso de los usuarios, los datos asociados son recolectados en el momento del primer acceso al sistema, y consisten en la especificación de los temas de las distintas asignaturas (*arreglos unidimensionales*, *sucesiones*, *límites*, *ciclos*, *estructuras de datos no líneales*…) que se deben ejercitar con más fuerza. Opcionalmente, el usuario puede indicarle al sistema que sea este el que le sugiera los temas a ejercitar (basándose en el perfil asociado a cada uno de los ejercicios que ya ha resuelto)

A cada problema también se le han de asociar los temas que comprende, siendo denominada esta actividad, en el marco de la solución, "etiquetado del problema". De esta tarea se encargan los profesores autores de los problemas.

2.2.2 Perfil algorítmico propuesto para el Módulo de Recomendación

El estudio de varias de las tendencias líderes a la hora de definir el perfil algorítmico para un sistema recomendador así como los diferentes puntos neurálgicos que suelen aparecer asociados a estos, permitieron definir la estrategia a aplicar en el módulo en cuestión.

Se concibieron dos algoritmos fundamentales para acoplar a la aplicación. Uno de ellos tiene un carácter *ad-hoc* y se basa en el modelo de filtrado colaborativo (Linden, y otros, 2003), apoyándose en la definición de una función de semejanza entre los usuarios, mientras que el segundo es una implementación de la recomendación basada en reglas de asociación.

2.2.2.1 Recomendación ad-hoc basada en filtrado colaborativo

La filosofía de la recomendación basada en filtrado colaborativo aplicada a los jueces en línea de programación, se basa en sugerir al usuario aquellos problemas no resueltos por él que han sido resueltos por sus semejantes.

Para representar la información asociada a un usuario en específico, se decide asumir un vector con la forma:

$$< cantidadProblemasResueltosTema\ 1, cantidadDeProblemasResueltosTema\ 2, \ldots >$$

Una vez obtenidos los vectores asociados a todos los usuarios se cuenta con todos los datos necesarios para aplicar la estrategia de recomendación.

El principio de funcionamiento de esta se basa en el hecho de que si dos usuarios tienen una cantidad semejante de ejercicios por cada uno de los temas, implica que estos tienen intereses semejantes, pudiendo ser considerado para generar las recomendaciones. Expresándolo de una forma un poco más técnica: dos usuarios cuyos vectores asociados tengan una alta correlación. Específicamente, el algoritmo *ad-hoc* para ofrecer recomendaciones al usuario X, es el siguiente:

1- Aplicar *correlación de Pearson*[34] entre el vector asociado a X y los asociados a los restantes usuarios del juez en línea.
2- Se multiplica cada grado de semejanza por la razón de problemas aceptados sobre total de envíos de cada uno de los usuarios. A los resultados de la multiplicación se le denomina *nivel de confiabilidad con respecto a X*.
3- Se crea, con los problemas del juez en línea, una sublista con aquellos que tengan al menos una tema en común con el perfil de X.
4- Para cada problema de esta sublista, se establece una suma de los niveles de confiabilidad con respecto a X, de todos los usuarios que le han dado solución correcta. A esta suma se le denomina *ponderación del problema*.
5- Se muestra, en calidad de recomendaciones finales esta sublista ordenada descendentemente por la ponderación de los problemas.

2.2.2.2 Recomendación basada en reglas de asociación

Con el fin de incorporarle características híbridas (Sobecki, 2006) al sistema de recomendación se define un segundo algoritmo, en este caso basado en reglas de asociación, las que en su conjunto conforman un modelo. Para esto, se realizó una adaptación de la técnica relacionada con este tema, descrita previamente.

El objetivo de este método es generar reglas de la forma P → Q, donde tanto P como Q son conjuntos de ejercicios. El significado de estas sería el siguiente: si un estudiante X logró resolver todos los problemas del conjunto P, esto implica que está capacitado para resolver todos los problemas del conjunto Q. Una vez obtenidas un número de reglas de este tipo, para realizarle recomendaciones a un estudiante basta con revisar si para alguna regla *r* se cumple que el estudiante haya resuelto todos los ejercicios presentes en el antecedente P_r de la misma. En este caso, se recomiendan los ejercicios implicados en la parte derecha de la regla que aún no han sido resueltos por el estudiante.

[34] Correlación de Pearson y Similitud basada en el coseno son las métricas más empleadas en sistemas de recomendación. En (Su, y otros, 2009) se manejan algunas deficiencias de la segunda que permitieron decidir por la aplicación de Correlación de Pearson. Se recomienda estudiar el comportamiento de las recomendaciones con una u otra métrica en el contexto actual.

De manera secuencial, la estrategia sería la siguiente:

1- Obtener las reglas de asociación.
2- Para cada regla de asociación, verificar si se cumple la condición de que el estudiante haya resuelto todos los ejercicios que forman parte del antecedente.
3- En caso de cumplirse la condición, revisar si existe algún ejercicio de los implicados que aún no haya resuelto.
4- En caso de existir, insertar el problema en una lista de posibles recomendaciones, así como el nivel de confianza de la regla que lo genera.
5- De haber sido anteriormente insertado el problema, actualizar el nivel de confianza, si el valor de este es superior al que tenía asociado dicho problema previamente.
6- Se devuelven, en calidad de recomendaciones, el listado ordenado de manera descendente según los niveles de confianza.

Para obtener las reglas de asociación se utilizó el algoritmo *Apriori*, definido en (Agrawal, y otros, 1994), considerado como el método de referencia a la hora de generar este tipo de estructuras.

Apriori permite la obtención de reglas de asociación con un nivel de soporte y de confianza mínimo determinado. En el dominio del problema actual, *nivel de soporte* de una regla es la probabilidad de que un estudiante haya resuelto todos los ejercicios presentes tanto en el antecedente como en el conjunto de ejercicios implicados en la regla, mientras que *nivel de confianza* es la probabilidad condicional de que un estudiante que haya resuelto los ejercicios del antecedente, haya resuelto también los ejercicios implicados.

La corrida de *Apriori* en determinados momentos de la ejecución de la aplicación garantizará la actualización continua de las reglas, permitiendo que las recomendaciones sean realizadas acorde al estado actual del juez en línea, agregándole dinamismo al proceso.

2.2.2.3 Análisis de los algoritmos de recomendación

Se puede afirmar que las dos estrategias de recomendación a utilizar como parte de la aplicación se complementan mutuamente. Por un lado, el algoritmo *ad-hoc* presenta una propuesta dinámica para la generación de recomendaciones que considera a todos los usuarios basándose en una función de semejanza definida y utilizada para

comparar a cada uno de ellos con el usuario a recomendar. Para establecer las sugerencias, considera el desempeño de cada uno de estos usuarios en el momento de recomendar determinados problemas resueltos por estos.

Haciendo referencia a la recomendación basada en reglas, algunos autores como los de (Pazzani, y otros, 2007) han considerado que estas no ofrecen el nivel de detalle y personalización que otros métodos garantizan. A pesar de esto, se considera que si estas reglas son regeneradas de manera periódica y automática, las recomendaciones derivadas ganan en precisión y dinamismo. Por estas razones se decidió la inclusión de este tipo de recomendación.

A través de la combinación de estos dos algoritmos se logra mitigar el efecto de algunas limitaciones de los sistemas de recomendación. El hecho de poseer un paquete de reglas de asociación como base para la generación de sugerencias posibilita enfrentar de manera exitosa muchas situaciones asociadas al arranque en frío en el caso de la existencia de nuevos usuarios, dándose más detalles de esto en la siguiente sección. El papel clave de un método de filtrado colaborativo dentro de la arquitectura de recomendación evita la sobre-especialización de los usuarios y, finalmente, con respecto a la limitante de la dispersión, aun cuando por determinado estado de la base de datos del juez en línea esta pueda presentarse afectando la calidad de las recomendaciones del filtrado colaborativo, el otro mecanismo de recomendación garantiza que el usuario no deje de recibir buenas sugerencias.

2.2.2.4 Tratamiento del arranque en frío

Los algoritmos presentados anteriormente no muestran ninguna recomendación en el caso que el usuario sobre los cuales se corran no tenga ningún ejercicio resuelto. Para esta situación se decide recomendar aquellos problemas que más apariciones tengan dentro de los conjuntos de antecedentes en las reglas generadas. Por la propia definición de regla de asociación, son estos problemas la antesala de la resolución de un buen número de ejercicios en el juez.

2.3 Concepción del módulo detector de plagio

El subsistema detector de plagio será accesible solo para los profesores y administradores registrados en Juez en Línea Académico. Este deberá satisfacer los siguientes requisitos funcionales:

Un juez en línea ajustado a las necesidades de la docencia

1. Permitir inspeccionar plagio entre un conjunto de sumisiones.
2. Permitir inspeccionar plagio entre las sumisiones de un problema.
3. Permitir inspeccionar plagio entre una sumisión y un conjunto de sumisiones.
4. Permitir inspeccionar plagio en un rango de sumisiones.
5. Mostrar los resultados de la inspección de plagio realizada.
6. Mostrar el resultado de la detección de plagio entre dos sumisiones.
7. Calcular porciento de similitud entre dos sumisiones.
8. Mostrar similitudes entre códigos.
9. Permitir al profesor enviar una notificación al usuario ante su posible implicación en un caso de fraude detectado.
10. Permitir al profesor emitir un criterio sobre los resultados de las detecciones.
11. Permitir al profesor eliminar los criterios emitidos.

El profesor o los administradores del sistema pueden interactuar con el detector de plagio <1> para iniciar el proceso de detección entre las soluciones almacenadas en el juez en línea <2>. Una vez encontradas las soluciones sospechosas <3>, el profesor visualiza las respectivas interfaces mostrando los niveles de similaridad entre las soluciones y otros detalles importantes que facilitan la confirmación de que es un caso de plagio confirmado <4>. Después de esto el profesor puede proceder a la comunicación vía email con los usuarios implicados o a realizar otras acciones sobre el sistema <5> (ver Figura 5).

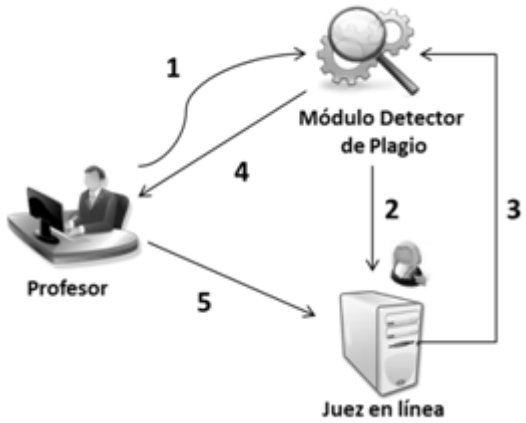


Figura 5. Sistema de detección de plagio

2.3.1 Perfil algorítmico propuesto para el sistema detector de plagio

La base algorítmica se sustenta en tres criterios que al complementarse generan el dictamen. Se implementó un detector para el análisis de la estructura del código fuente basado en el algoritmo Greedy-String-Tilling (Kazim, 2008), otro basado en las características del problema y un último detector que evalúa la probabilidad de que un usuario, según su historial, pueda haber cometido plagio.

2.3.1.1 Detector de plagio basado en la estructura del código fuente

Para la implementación del detector de plagio basado en la estructura del código fuente, se implementaron analizadores léxicos para los lenguajes de programación C, C++, Java, Pascal, Python y CSharp, que incluyen los de mayor importancia para la docencia de la UCI.

Con respecto a las parejas de fragmentos de código detectados, el dictamen final queda formulado como sigue: Sean $\boldsymbol{A_c}$ y $\boldsymbol{B_c}$ los códigos asociados a dos soluciones ***A*** y ***B*** respectivamente, ***S*** el conjunto de similitudes detectadas entre tales soluciones y ***l*** la longitud de la similitud entonces

$$similitud(A,B) = \frac{2 \quad x \in S\, x_l}{A_c + B_c}$$

Ecuación 1: Similitud de acuerdo a las coincidencias detectadas por el algoritmo Greedy String Tilling

2.3.1.2 Detector de plagio basado en el perfil del problema objetivo de la solución

Un segundo detector fue implementado basado en el perfil del problema objetivo de la solución. Este detector tiene en cuenta la clasificación que los autores de los problemas dan a sus ejercicios. Si por ejemplo, al resolver un problema este supone encontrar la distancia mínima de un nodo a los demás y sobre esa distancia realizar algún tipo de procesamiento, por ejemplo ir y volver a la mayor cantidad de nodos contando con un límite de distancia total a recorrer, lo cual podría verse desde una óptica golosa; entonces será muy probable encontrar entre las clasificaciones de este problema: Dijkstra y Goloso. La comparación de las estructuras de problemas solubles mediante algoritmos clásicos necesariamente arrojará altos niveles de similitud.

Se implementó un procedimiento almacenado que permite estimar regularmente, según los datos reales del sistema, la longitud promedio de código que se emplea para implementar determinado algoritmo dado su clasificación. De acuerdo a las clasificaciones coincidentes en los problemas asociados con las soluciones

comparadas, el dictamen se calcula como sigue: Sean A_c y B_c los códigos asociados a dos soluciones ***A*** y ***B*** respectivamente, A_f y B_f respectivamente los conjuntos de clasificaciones asociados a los problemas, F_l la longitud promedio estimada para una clasificación ***F***, I_c el código de la solución I-ésima entonces

$$similitud\ A,B\ = \frac{A_c\ +\ B_c\ -2\ \ _{F\ \epsilon\ [A_f\ \ B_f]}\ F_l}{2\ \ \underset{1\leq i\leq\infty}{max}\ I_c}$$

Ecuación 2: Similitud de acuerdo a las coincidencias detectadas con el comparador de problemas

2.3.1.3 Detector de plagio basado en el perfil del usuario

Por último se realiza un estudio del historial de cada usuario con el objetivo de determinar la probabilidad de que hayan cometido plagio. Dicho criterio se basa en los dictámenes realizados a otros ejercicios en los que el usuario ha estado involucrado.

Sean ***B*** una solución de un usuario ***u***, B_d los dictámenes de las detecciones en que ***u*** ha estado implicado previamente, B_r la cantidad de ejercicios que ha resuelto entonces:

$$similitud\ A,B\ = \frac{\frac{x\ \epsilon\ B_d\ x}{B_d} + \frac{B_d}{B_r}}{2}$$

Ecuación 3: Similitud de acuerdo al algoritmo comparador de acuerdo al historial del usuario

Finalmente, los resultados se ponderan, se unen y se evalúan por los administradores y profesores que juegan el rol de jueces de plagio, los cuales son los encargados de emitir criterios sobre el dictamen final y los dictámenes parciales. El objetivo de esos criterios es enriquecer la calidad de los dictámenes del módulo en futuras versiones y determinar las acciones a llevar a cabo con los usuarios involucrados en un plagio, detectado por el módulo y ratificado por los jueces. Se brinda la funcionalidad de notificación a los usuarios, sanción temporal de denegación de servicios y sanción permanente, pérdida de su cuenta en el sistema.

Sean ***A*** y ***B*** dos soluciones, ***D*** el conjunto de detectores aplicados a estas soluciones, ***d*** el dictamen de cada detector aplicado y ***p*** la ponderación del detector, entonces:

$$Dictamen(A,B) = \frac{\sum_{x \in D} x_d * x_p}{\sum_{x \in D} x_p}$$

Ecuación 4: Dictamen general del módulo para la detección de plagio

2.4 Conclusiones

En el presente capítulo se realizó una descripción general del Juez en Línea ajustado a la docencia que fue desarrollado. Se exponen las razones por la que es considerado adecuado desarrollar el nuevo sistema partiendo de la solución actual para el Juez en Línea del Caribe.

Se enfatiza, por ser el principal aporte de este trabajo, en la concepción del Módulo de Recomendaciones y el Módulo de Detección Automática de Plagio.

Se describen dos algoritmos un basado en métricas de similaridad entre usuarios y el otro en reglas de asociación para brindar recomendaciones a los usuarios del sistema. Son analizadas las principales limitaciones de los sistemas recomendaciones sobre todo referidas al avance en frío y cómo con la estrategia desarrollada se cubren las mismas. El desarrollo de un módulo recomendación de problemas en jueces en línea constituye un importante aporte sobre todo si se tiene en cuenta que prácticamente no existen soluciones de este tipo en este campo.

Por último de detalla el Módulo para la Detección de Plagio así como los algoritmos y enfoques que se tendrán en cuenta para detectar soluciones plagiadas de una manera efectiva. En particular el algoritmo Greedy-String-Tiling implementado para encontrar semejanzas entre dos soluciones no es una novedad pero permite disponer de un enfoque de detección semejante al de otras importantes herramientas como Jplag y YAP. Sin embargo los enfoques basados en el perfil del usuario y el perfil del problema sí constituyen un importante aporte.

Capítulo 3. Validación de la Propuesta

La validación del juez en línea desarrollado está sustentada primeramente en la calidad de funcionamiento del COJ, su sistema base, el cual ha brindado servicios durante dos años a los usuarios caribeños y de otros países sin presentar dificultades. Desde su publicación el 5 de junio del 2010, se han registrado 7860 usuarios de 491 instituciones pertenecientes a 109 países, se han enviado 315635 intentos de solución y fueron celebradas 212 competencias.

En particular, el nuevo sistema ha sido utilizado del septiembre de 2011 hasta la fecha apoyando la asignatura IP en la Facultad 4. Durante el curso pasado se sumaron algunos grupos de las facultades 5 y 6 con objetivo de ejercitar para el examen extraordinario.

Aunque puede inferirse de sus características de herramienta que automatiza los procesos de evaluación, detección de plagio y recomendación de problemas; el Juez en Línea Académico desarrollado disminuye en tiempo de trabajo y el esfuerzo del profesor a prácticamente realizar las descripciones de los problemas asociados a cada una de las asignaturas. Esto es una actividad que en las experiencias de la Facultad 4 se ha desarrollado durante las preparaciones metodológicas de los colectivos de asignaturas. Por otro lado disminuye el tiempo necesario a un estudiante para recibir la retroalimentación sobre la calidad de los algoritmos e implementaciones desarrolladas, aspecto que anteriormente se hacía bastante complejo para el profesor.

Teniendo en cuenta su importancia, se procede en este capítulo, además, a la validación de los módulos que constituyen el aporte principal de este trabajo.

3.1 Evaluación del Módulo de recomendación

Evaluar un algoritmo de recomendación constituye una de las tareas de mayor complejidad dentro del área de investigación y desarrollo que constituyen los sistemas de recomendación. Esto viene dado por la dificultad para determinar si una estrategia determinada genera recomendaciones provechosas para los usuarios finales, debido a la infinita cantidad de perfiles distintos que pueden asumir cada uno de dichos usuarios. Por esta razón, muchos prestigiosos autores han priorizado dentro de su

línea de investigación la obtención de métricas eficaces para evaluar a los sistemas de recomendación.

Para determinar la precisión de un sistema de recomendación, en la mayoría de los casos se han utilizado métricas que a pesar de provenir de otras áreas del conocimiento, tienen aplicación en el campo de este tipo de sistemas. Estas se dividen en dos grupos fundamentales: aquellas que están enfocadas a evaluar cuán cerca está la predicción, por parte de la aplicación, del *rating* que daría un usuario a un determinado ítem con respecto a la evaluación real dada por este usuario; y un segundo grupo que se dedica a medir la frecuencia con la que un sistema hace suposiciones correctas con respecto a si un ítem es realmente apropiado o no para un usuario en específico (Herlocker, y otros, 2004).

Dada las características del sistema en cuestión, las métricas pertenecientes al segundo grupo son las más apropiadas para realizar la evaluación de este. Estas son usualmente llamadas métricas de precisión de la clasificación.

A continuación se aplica una de estas métricas al listado de recomendaciones generadas por la combinación de los dos algoritmos del sistema de recomendación desarrollado, a través de un experimento *offline* (Gunawardana, y otros, 2009), utilizando los datos disponibles en el juez en línea de Programación.

Precision and Recall (TREC, 2001) es una de las métricas más populares para evaluar los sistemas de recuperación de información. *Recall* se encarga de medir la habilidad de un sistema de presentar *todos* los ítems relevantes, mientras que *Precision* mide la capacidad de presentar al usuario final *sólo* los ítems relevantes. Es considerada la métrica de precisión de la clasificación por excelencia a utilizar en los sistemas de recomendación.

La aplicación de *Precision and Recall* en este tipo de sistemas difiere ligeramente con respecto a la forma de aplicarla en otros marcos de propósito más general referidos en (TREC, 2001). Como parte de la evaluación *offline*, normalmente se tienen datos con los ítems que cada usuario ha utilizado o preferido. Para realizar la prueba, se selecciona un usuario específico, y se divide el conjunto de ítems asociados (sus problemas resueltos), en dos subconjuntos, llamados *conjunto de pre-recomendaciones* y *conjunto de pruebas*. Seguidamente al conjunto de pre-recomendaciones se le aplica el algoritmo a validar, y se seleccionan las n-primeras recomendaciones generadas por este. A su vez, los elementos que aparecen tanto en

el conjunto de pruebas como en el de las n-primeras recomendaciones, se consideran como parte de un conjunto llamado *hit set*.

Dentro de este contexto, el *Precision* y el *Recall* se definen de la siguiente manera (Cristache, 2009):

$$recall = \frac{tamaño\ del\ hit\ set}{tamaño\ del\ conjunto\ de\ prueba}$$

$$precision = \frac{tamaño\ del\ hit\ set}{Cantidad\ de\ n - primeras - recomendaciones}$$

Una de las formas más tradicionales de representar los valores del *precision* y el *recall,* para un usuario determinado, es a través de curvas de *precision y recall* que se representan en el plano cartesiano y en las que por cada uno de los ejes X y Y se ubican respectivamente los valores de *precision* y *recall.*

Es importante resaltar que *precision* y *recall* usualmente entran en conflicto a la hora de la práctica. Normalmente, a la hora de incrementar el valor de n en las n primeras recomendaciones a obtener, aparece una tendencia a incrementarse el valor del *recall* y de disminuir el valor de *precision*. Por esta razón, se han visto dirigidos los esfuerzos a obtener una métrica que unifique los dos componentes de la anterior. De los varios resultados alcanzados, la de más aplicación ha sido *F1* (Yang, y otros, 1999), que es a su vez la seleccionada para utilizarse en el asunto actual.

$$F1 = \frac{2 * Recall * Precision}{(Recall + Precision)}$$

Para el algoritmo ad-hoc del sistema de recomendación obtenido, se ejecutó el cálculo de los valores de *recall*, *precision* y F1 para los 50 usuarios punteros del COJ, considerados los de mejores criterios a la hora de seleccionar los problemas a resolver. El conjunto de pruebas, para cada uno de los usuarios, estuvo conformado por el 20% del total de sus ejercicios resueltos, seleccionados de manera aleatoria. Para cada uno de los usuarios se ejecutaron varias corridas en las que se varió, desde 5 hasta 25, el número N de recomendaciones a ser devueltas por el algoritmo de recomendación, con el objetivo de ver el comportamiento de los tres parámetros que se miden. En la figura 6 se muestran algunos de los resultados de estas corridas.

Para poder comparar el sistema de recomendación con otros similares, no es suficiente con tener los valores de estos parámetros para cada uno de sus usuarios. Se hace

necesario obtener valores globales de *precision*, *recall* y F1 que describan el funcionamiento global del sistema, y no el comportamiento de este ante determinado usuario en específico.

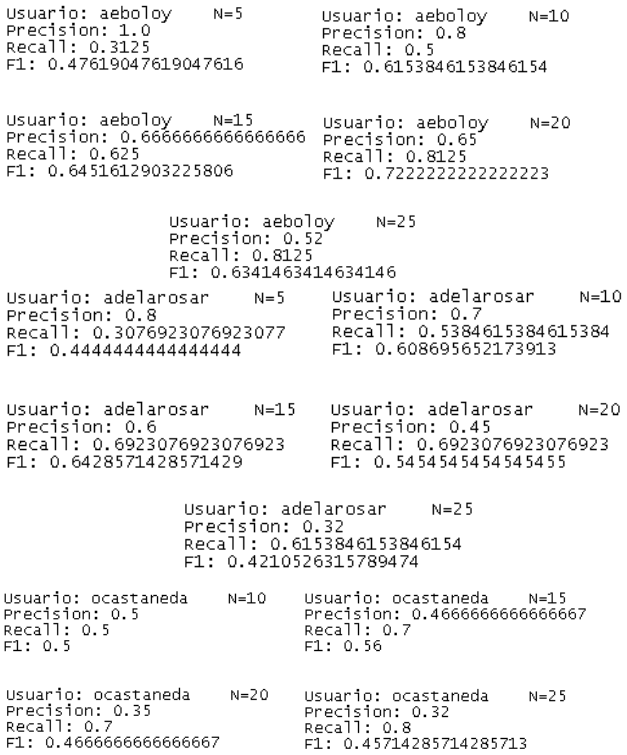


Figura 6. Valores de Precision, Recall y F1 para algunos usuarios del COJ

Se realizó el cálculo de nuevos valores utilizando las sugerencias brindadas en (Shani, 2009), que indican computar los promedios de los valores de *precision* y *recall* de todos los usuarios para cada valor de N, y con sus promedios, calcular el F1. Los resultados de este proceso se muestran en la figura 7.

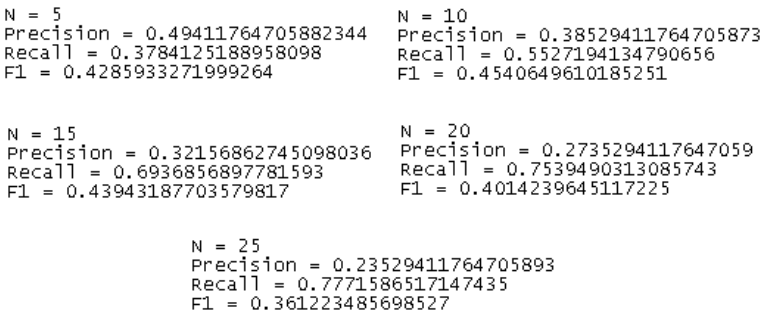


Figura 7. Valores globales de precision, recall y F1 para el sistema de recomendación para jueces en línea

Es válido mencionar que los valores de Precision, Recall y fundamentalmente F1 obtenidos de este último cálculo están en correspondencia con los obtenidos por otros sistemas recomendadores como los descritos en (Nasraoui, y otros, 2007), (Liang, y otros, 2009) y (Bedi, y otros, 2009).

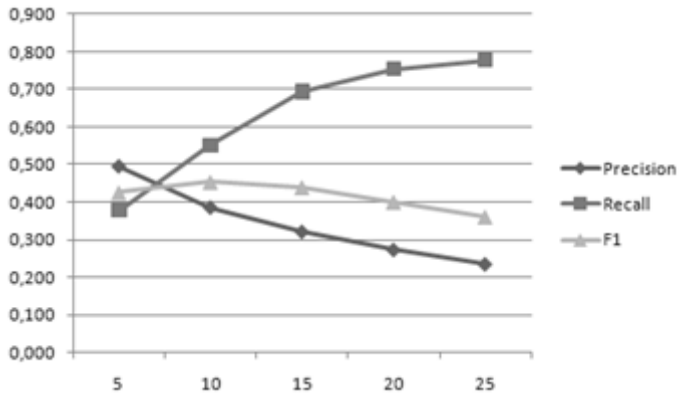


Figura 8. Representación gráfica de los valores globales de precision, recall y F1 para el sistema de recomendación

Resulta razonable el desbalance entre los valores obtenidos y los asociados a cada uno de los sistemas recomendadores mencionados. Esto viene dado por las características de las fuentes de datos utilizadas en cada caso, los algoritmos empleados, los objetivos finales del sistema en sí; y por estas razones, los sistemas no son comparables. No obstante, se referenciaron los datos con el objetivo de hacer ver que las recomendaciones brindadas por el sistema desarrollado poseen una calidad semejante a las que se pueden obtener de sistemas con un alto reconocimiento por la comunidad científica internacional. Este es uno de los resultados más importantes del trabajo.

Finalmente, es importante destacar que el resultado de que los mayores valores de F1 fueron obtenidos para las corridas en que se devolvieron de 5 a 10 recomendaciones, fue utilizado para determinar la cantidad de recomendaciones a mostrar al usuario en la funcionalidad “Recomendaciones Automáticas”.

3.2 Evaluación del Módulo de Detección de Plagio

La validación de la solución desarrollada asegura que el producto final cumpla con los requerimientos establecidos para el software, acredita además la calidad y eficiencia del módulo. En la actualidad existen disímiles métodos para determinar la validez de un

producto, los cuales presentan diferentes enfoques. Para la validación del Módulo de Detección de Plagio se procedió mediante la validación de los algoritmos y el grado de aceptación de los clientes. Para ello se aplica la técnica de ladov para medir el nivel de satisfacción del personal que utilizará el módulo implementado.

La técnica de ladov fue creada originalmente por Kuzmina en 1970, utilizada por López Rodríguez y González Maura en el 2002, para el estudio de la satisfacción por la profesión en carreras pedagógicas. Luego, González Maura y López Rodríguez en ese mismo año realizaron una transformación a esta técnica y la plantean como alternativa para el diagnóstico de la motivación profesional en profesores de Educación Física (Gómez López, y otros, 2006).

Constituye actualmente un camino indirecto para el conocimiento de la satisfacción, se utilizan criterios que establecen relaciones entre tres preguntas que se intercalan dentro del cuestionario y cuya existencia es desconocida por los involucrados. Estas cuestiones se relacionan a través del "Cuadro lógico de V. A. ladov" (ver tabla 2) desarrollado por los citados autores.

Para la recogida de los datos que permitirán conocer la satisfacción, alrededor de la creación y eficiencia del módulo para la detección de plagio, se confeccionó un cuestionario conformado por diez preguntas, de ellas siete cerradas y tres abiertas que posteriormente fueron analizadas mediante la técnica V. A. ladov. El cuestionario se aplica a los administradores del COJ (juez en línea donde se desplegó el subsistema para realizarle las pruebas de validación), el Colectivo de Entrenadores del Movimiento del MPC-TLJ y varios concursantes ACM-ICPC de la Universidad. Las preguntas están basadas principalmente en la necesidad de la creación de un módulo para la detección de plagio y la eficiencia de las detecciones de plagio realizadas por los algoritmos implementados (ver anexo 4).

Tabla 2: Cuadro lógico de V. A. ladov

<table>
<tr><td rowspan="3">5. ¿Qué usted cree del módulo para la detección de plagio en el Juez en Línea</td><td colspan="3">1. ¿Cree usted que el plagio es un comportamiento correcto?</td></tr>
<tr><td>No</td><td>No sé</td><td>Sí</td></tr>
<tr><td colspan="3">3. ¿Cree usted que la dificultad en la detección de plagio hace necesaria la creación de un módulo que facilite el proceso?</td></tr>
</table>

Un juez en línea ajustado a las necesidades de la docencia

Académico?	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta tanto	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

El resultado de la interrelación de tres preguntas cerradas (1, 3 y 5) conlleva a un número que indica la posición de cada uno de los encuestados en la escala de satisfacción siguiente:

1. Clara satisfacción
2. Más satisfecho que insatisfecho
3. No definida
4. Más insatisfecho que satisfecho
5. Clara insatisfacción
6. Contradictoria

La aplicación de la técnica de Iadov y de las preguntas adicionales mostradas en el cuestionario, constituyen un instrumento de gran utilidad para el estudio y evaluación de la satisfacción de los encuestados.

Para obtener el índice de satisfacción grupal (ISG) se establece una escala numérica entre +1 y -1 con los diferentes niveles de satisfacción como se muestra en la tabla 3 (López, 2002).

Tabla 3: Escala numérica de satisfacción

Valor	**Interpretación**
+1	Máximo de satisfacción
0.5	Más satisfecho que

	insatisfecho
0	No definido y contradictorio
-0.5	Más insatisfecho que satisfecho
-1	Máxima insatisfacción

El índice de satisfacción grupal se calcula por la siguiente fórmula:

$$ISG = \frac{A\ +1\ +B\ 0.5\ +C\ 0\ +D\ -0.5\ +E(-1)}{N}$$

En esta fórmula ***A***, ***B***, ***C***, ***D***, ***E***, representan el número de sujetos con índice individual 1, 2, 3, 4, 5 ó 6 respectivamente y ***N*** representa el número total de sujetos del grupo. El ISG arroja valores entre +1 y -1. Los valores que se encuentran comprendidos entre -1 y -0.5 indican insatisfacción; los comprendidos entre -0.49 y +0.49 evidencian indefinición o contradicción y los que caen entre 0.5 y 1 indican que existe satisfacción. Estos valores se pueden representar en un eje de coordenada como se aprecia en la figura 9.

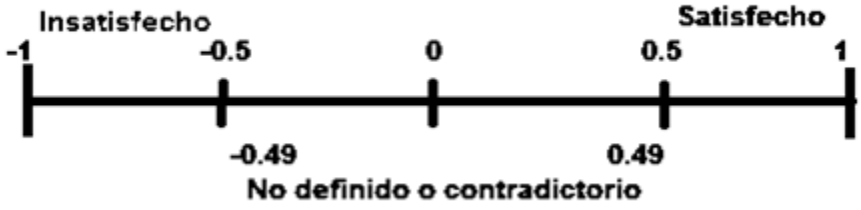


Figura 9: Eje numérico de satisfacción

El procedimiento de cálculo del ISG de la encuesta realizada a 15 personas queda reflejado a continuación:

$$ISG = \frac{A\ +1\ +B\ 0.5\ +C\ 0\ +D\ -0.5\ +E(-1)}{N}$$

$$ISG = \frac{10\ +1\ +4\ 0.5\ +0\ 0\ +1\ -0.5\ +0(-1)}{15}$$

$$ISG = 0.76$$

El resultado arrojado demuestra clara satisfacción con un ISG de 0.76 dentro del rango [0.5, 1].

3.3 Conclusiones

En el presente capítulo se realizó la validación del sistema de recomendación para jueces en línea de programación, acentuando en lo que constituyen los aportes sustanciales de este trabajo: Módulo Recomendador de Problemas y Módulo Detector de Plagio.

Para la evaluación del perfil algorítmico del Módulo Recomendador de Problemas se utilizó la métrica *Precision and Recall* para la evaluación del algoritmo *ad-hoc* de filtrado colaborativo y del algoritmo basado en reglas. Con esta se obtuvieron resultados satisfactorios.

Para la validación del Módulo Detector de Plagio se procedió con el uso de la técnica ladov para el cálculo de la satisfacción de los usuarios. Luego de aplicarles una encuesta a quince personas el índice de satisfacción grupal (ISG) resultante permitió determinar que el módulo para la detección de plagio cuenta con un buen nivel de aceptación entre sus usuarios.

Conclusiones Generales

1. A partir del estudio realizado de diferentes jueces en línea se determinó que ninguno cumplía con las características necesarias para solucionar la problemática planteada en la introducción de este trabajo.
2. Se desarrolló un juez en línea ajustado a las necesidades docentes, el cual incluye como principal aporte módulos para Recomendación de Problemas y Detección automática de plagio.
3. Se validó el Módulo de Recomendación haciendo uso de las métricas Precisión, Recall y F1, obteniendo resultados semejantes a los de otros sistemas recomendadores reconocidos.
4. Fue validado el Módulo de Detección de Plagio mediante la técnica de ladov creada para medir satisfacción, obteniéndose un índice de satisfacción grupal de 0.76 entre los expertos encuestados lo que supone un nivel de satisfacción adecuado.

Recomendaciones

Teniendo en cuenta el estado actual del desarrollo del Juez en Línea Académico se recomienda lo siguiente:

1. Integrar mediante una arquitectura orientada a servicios el Juez en Línea Académico con la plataforma Moodle. Esto unificará la gestión del profesor con respecto al seguimiento de los estudiantes y fundamentalmente mejorará la gestión de los grupos docentes.
2. Crear una comunidad de desarrolladores del Juez en Línea Académico con profesores interesados de la UCI. Este es un proyecto que requerirá cambios constantes.
3. Desplegar el sistema en otras universidades del país o extender su desarrollo de manera que permita centralizar esta herramienta para apoyar la enseñanza de la programación a nivel nacional en todas las carreras relacionadas con la Computación y la Informática.

Referencias Bibliográficas

Kurnia, Andy, Lim, Andrew y Cheang, Brenda. 2001. *Online Judge.* s.l. : Computers & Education, 2001. Thesis of Honor.

Adomavicius, Gediminas y Tuzhilin, Alexander. 2005. Towards the Next Generation of Recommender Systems:A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering.* 2005, Vol. 17, págs. 734-749.

Aggarwal. 1999. *Horting hatches an egg: a new graph-theoretic approach to collaborative filtering.* San Diego, Califormia : s.n., 1999. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining.

Agrawal, Rakesh y Srikan, Ramakrishnan. 1994. *Fast Algorithms for Mining Association Rules.* Santiago de Chile : s.n., 1994. Proceedings of the 20th VLDB Conference.

Aiken, Alex. 1994. Moss: A System for Detecting Software Plagiarism. [Online] 1994. [Cited: 10 1, 2012.] http://theory.stanford.edu/~aiken/moss/.

Barrón Cedeño, Alberto, Vila, Marta y Rosso, Paolo. 2011. *Detección automática de plagio: de la copia exacta a la paráfrasis.* Valencia : Natural Language Engineering Lab, 2011.

Bedi, Sharma y Kaur. 2009. Recommender System based on Collaborative Behavior of Ants. *Journal of Artificial Intelligence.* 2009.

Billsus, D. 2000. *User modeling for adaptive news access.* 2000.

Bock, Hans-Hermann. 2007. Clustering Methods: A History of k -Means Algorithms. *Selected Contributions in Data Analysis and Classification.* 2007, págs. 161-172.

Breese. 1998. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* 1998.

Cedeño, Luis Alberto Barrón. 2008. *Detección automática de plagio en texto.* Universidad Politécnica de Valencia. Valencia : s.n., 2008. Tesis desarrollada dentro del Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital.

Cheang, Brenda, y otros. 2003. On automated grading of programming assignments in an academic institution. *Computers & Education.* 2003, 41, págs. 121–131.

Cristache, Alex. 2009. *Hybrid recommender system using association rules.* Auckland, New Zealand : s.n., 2009.

Crochemore, Maxime y Rytter, Wojciech. 1997. *Efficient Algorithms on Text.* 1997. págs. 367-369.

Galán, Sergio Manuel. 2007. *Filtrado colaborativo y sistemas de recomendación.* Madrid, España : s.n., 2007.

García, Ginés y Fernández Alemán, José Luis. 2009. *A course on algorithms and data structures using on-line judging.* Murcia : s.n., 2009. págs. 45-49, Actas de Conferencia.

Gemmis, De. 2009. *Preference Learning in a Recommender System.* Bled, Slovenia : s.n., 2009. European Conference on Machine Learning and Principle and Practice of Knowledge Discovery in Databases.

Giménez, Omer. 2012. *Jutge.org: An Educational Programming Judge.* Catalunya : ACM, 2012.

Gómez López, Manuel, y otros. 2006. Las clases de Educación Física y el deporte extraescolar entre el alumnado almeriense de primaria. Una aplicación práctica mediante la técnica de ladov. *Revista Digital - Buenos Aires.* 2006.

Gries, David. 2006. *What Have We Not Learned About Teaching Programming?* s.l. : Computer, 2006. pp. 81-82. 10.

Gunawardana y Shani. 2009. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research.* 2009, 10.

Guo, Mingyu. 2012. Principles of C and Memory Management. *Comp281_Assignment 2.* [En línea] 2 de Marzo de 2012. [Citado el: 1 de 10 de 2012.] www.csc.liv.ac.uk/.../comp281_assignment2.pdf.

Heng, PeyShan, y otros. 2005. *Evaluation of the BOSS Online Submission and Assessment System.* Warwick : s.n., 2005.

Herlocker, Jonathan, y otros. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems.* 2004, Vol. 22, 1.

Higgins, Colin. 2003. *The CourseMarker CBA System: Improvements over Ceilidh.* Nottingham : s.n., 2003.

Hill, W. 1995. *Recommending and evaluating choices in a virtual community of use.* 1995. Proceedings of CHI'95.

Joy, Mike y Luck, Michael. 1999. Plagiarism in Programming Assignments. *IEEE Transactions on Education.* May de 1999, Vol. 2, 2.

Joy, Mike, Griffiths, Nathan y Boyatt, Russel. 2005. The BOSS Online Submission and Assessment System. 2005, Vol. 5, 3.

Kazim, Norulazmi Bin. 2008. *Plagiarims Detection System for Java Programming Assigments by Using Greedy String Tilling Algorithm.* Malaysia : College of Arts and Sciences, 2008. pág. 15, Tesis para obtar por el grado MSc. (ICT).

Konstan, Josep. 2004. IntroductionTo Recommender Systems: Algorithms and Evaluations. *ACM Transactions on Information Systems.* 2004, págs. 1-4.

Leal, José. 2008. *Integration of E-Learning Systems With Repositories of Learning Objects.* 2008.

Liang, Xu y Li. 2009. *Collaborative Filtering Recommender Systems based on Popular Tags.* Sydney, Australia : s.n., 2009.

Linden, Greg, Smith, Brent y York, Jeremy. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing.* 2003, págs. 76-80.

López, Alejandro. 2002. La técnica de ladov: Una aplicación para el estudio de la satisfacción de los alumnos por las clases de educación física. *Revista Digital - Buenos Aires.* 4 de 2002.

López, Jorge. 2011. Una plataforma de evaluación automática con una metodología efectiva. 2011, Vol. 19, 2, págs. 265-277.

Luo, Yingwei. 2008. *Programming grid: a computer-aided education system for programming courses based on online judge.* 2008.

Mckinstry, Jeff. 2012. Syllabus CSC 412 Topics in Computer Science (2 units). *PLNU.* [En línea] 2012. www.pointloma.edu.

Montoya, Francisco, Fernández Alemán, José y García Mateos, Ginés. 2009. *An experience on ada programming using on-line judging.* 2009. págs. 75-89, Actas de Conferencia.

Nasraoui, Olfa, y otros. 2007. *Performance of Recommendation Systems in Dynamic Streaming Environments.* 2007.

Nghi, Truong. 2007. *A web-based programming environment for novice programmers.* Queensland University of Technology. Australia : s.n., 2007. PhD Thesis.

Othman, Mahfudzah y Othman, Muhaini. 2012. The proposed model of collaborative virtual learning environment for introductory programming course. *Turkish Online Journal of Distance Education.* 2012, Vol. 13, 1.

Ottenstein, Karl J. 1976. An algorithmic approach to the detection and prevention of plagiarism. 1976, págs. 30–41.

Oviedo Galdeano, Mario, Ortiz Uribe, Frida Gisela y Oviedo Galdeano, Humberto. 2005. *Actitudes frente al aprendizaje y práctica de la Programación.* México D.F. : s.n., 2005.

Oviedo, Mario. 2005. *Técnicas efectivas de evaluación académica de grupos masivos en las asignaturas de Programación.* México D.F. : s.n., 2005.

Pazzani y Billsus. 2007. Content-based Recommendation Systems. *The adaptive web.* s.l. : Springer-Verlag, 2007.

Pearl, J. 1988. *Probabilistic Reasoning in Expert Systems: networks of plausible inference.* San Francisco : Morgan Kauffman Publishers, 1988.

Petit, Jordi. 2009. *Programación-1 Una asignatura orientada a la resolución de problemas.* Cataluña : s.n., 2009.

Prechelt, Lutz, Malpohl, Guido y Phlippsen, Michael. 2000. *JPlag: Finding plagiarisms among a set of programs.* Facultad de Informática, Universidad de Karlsruhe. Alemania : s.n., 2000. Technical Report.

Resnick, Paul. 1994. *GroupLens: An Open Architecture for Collaborative Filtering of Netnews.* 1994. págs. 175-186, Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work.

Revilla, Miguel A. 2008. Competitive Learning in Informatics: The UVa. *Olympiads in Informatics.* 2008, Vol. 2, págs. 131–148.

Robins, Anthony. 2003. *Learning and Teaching Programming: A Review and Discussion.* 2003. págs. 137-172.

Rodríguez, Marko. 2009. A recommender system to support the scholarly communication process. *The Computing Research Repository.* 2009.

Roque Alvarez, Jorge Luis. 2010. COJ: Frequently Asked Questions (FAQ). [Online] UCI, Junio 5, 2010. [Cited: 10 23, 2012.] http://coj.uci.cu/general/faq.xhtml#faq.12.

Schleimer, Saul, Wilkerson, Daniel S. y Aiken, Alex. 2003. *Winnowing: Local Algorithms for Document Fingerprinting.* s.l. : ACM, 2003.

Seagaran, Toby. 2005. *Programming collective intelligence.* s.l. : O'Reilly, 2005.

Shani. 2009. Evaluating Recommendation Systems. 2009.

Shun-xin, Wu y Bao-lan, Linag. 2011. Research on Teaching Method of Programming Design Course. *CNKI.* 2011.

Sobecki, Janusz. 2006. Implementations of Web-based Recommender Systems Using Hybrid Methods. *International Journal of Computer Science & Applications.* 2006, págs. 52-64.

Su, Xiaoyuan y M. Khoshgoftaar, Taghi. 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence.* 2009, Vol. 2009.

Swearingen, Kirsten. 2001. *Beyond Algorithms: An HCI Perspective on Recommender Systems.* 2001. ACM SIGIR 2001 Workshop on Recommender Systems.

Terveen, L y Hill, W. 2001. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium.* 2001, págs. 487-509.

Tiantian, Wang. 2011. *Programming, Problem Solving Ability Training and Assessment with AutoLEP.* Harbin : s.n., 2011. págs. 87-90.

TREC. 2001. *Common Evaluation Measures.* s.l. : NIST, 2001. The Tenth Text REtrieval Conference.

Vélez, Oswaldo. 2005. Aproximando a los sistemas recomendadores desde los algoritmos genéticos. *Revista Colombiana de Computación.* 2005, págs. 7-23.

Verdú, Elena. 2010. *Contribución a la aplicación de técnicas de inteligencia artificial para el diseño efectivo de sistemas adaptativos de aprendizaje competitivo.* Universidad de Valladolid. 2010. Tesis Doctoral.

Vozalis, Emmanouil. 2003. *Analysis of Recommender Systems' Algorithms.* Atenas, Grecia : s.n., 2003. The 6th Hellenic European Conference on Computer Mathematics & its Applications.

Walker, T.D.L. y Goodman, J.K. 2005. OMEN: An Online Grader for Engineering Programming Courses. [ed.] Northeast Consortium for Engineering Education. *Computers in Education.* 2005, Vol. 15, 4, págs. 47-52.

Warsaw, University of. 2012. *ICPC Fact Sheet – World Finals Release.* 2012.

Wen-xin, Li. 2005. Peking University Online Judge and Its Applications. 2005.

Wise, Michael. 1996. *YAP3: Improved detection of similarities in computer program and other texts.* Department of Computer Science, University of Sydney. Sydney, Australia : s.n., 1996.

Yang, Y y Liu, X. 1999. *A Re-examination of Text Categorization Methods.* 1999.

Zhigang, Sun. 2011. *Applying Online Judge and Anti-Plagiarism to Improve Programming Courses.* Harbin : s.n., 2011. págs. 26-28.

Anexos

Anexo 1. Resumen de Cuestionario Integrador a nuevo ingreso, Facultad 4

27. ¿Participó en concursos estudiantiles?

34	55.74 %	**Sí**
25	40.98 %	**No**
2	3.28 %	**(S / R)**

28. Marque con una X las asignaturas en las que participó, el nivel del concurso y si alcanzó premio.

	Centro	Municipal	Provincial	Nacional	Premio
Matemática	8 (40 %)	7 (35 %)	4 (20 %)	1 (5 %)	(0 %)
Computación	4 (20 %)	2 (10 %)	1 (5 %)	(0 %)	(0 %)

33. ¿Participó en su centro o municipio de residencia en algún movimiento de jóvenes en torno a la computación?

9	14.75 %	**Sí**
50	81.97 %	**No**
2	3.28 %	**(S / R)**

34. Especifique en ¿cuáles?

1	1.64 %	**Preselección computación**
3	4.92 %	Colaborador de los Joven Club de Computación y Electrónica
2	3.28 %	Aficionados de la computación
3	4.92 %	**Grupos de concurso de computación**

35. ¿Fue monitor de alguna asignatura?

51	83.61 %	**Sí**
8	13.11 %	**No**
2	3.28 %	**(S / R)**

36. Marque con una X las asignaturas en las que fue monitor.

30	49.18 %	**Matemática**
14	22.95 %	**Computación**

37. De las asignaturas en las que más usted se ha destacado durante su vida estudiantil, díganos aquellas 3 en las que se haya destacado más, otorgándole el valor 1 a aquella que haya sido la que más se ha destacado y así sucesivamente hasta llegar a las 3.

33	54.1 %	**Matemática**
6	9.84 %	**Computación**

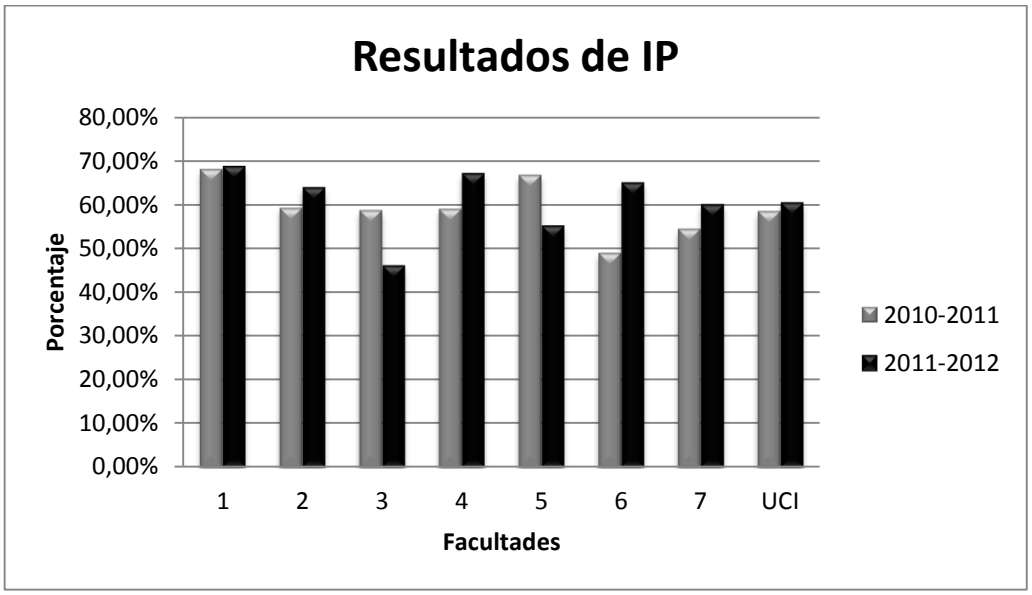
47. ¿Ha tenido usted algún tipo de experiencia con algún lenguaje de programación

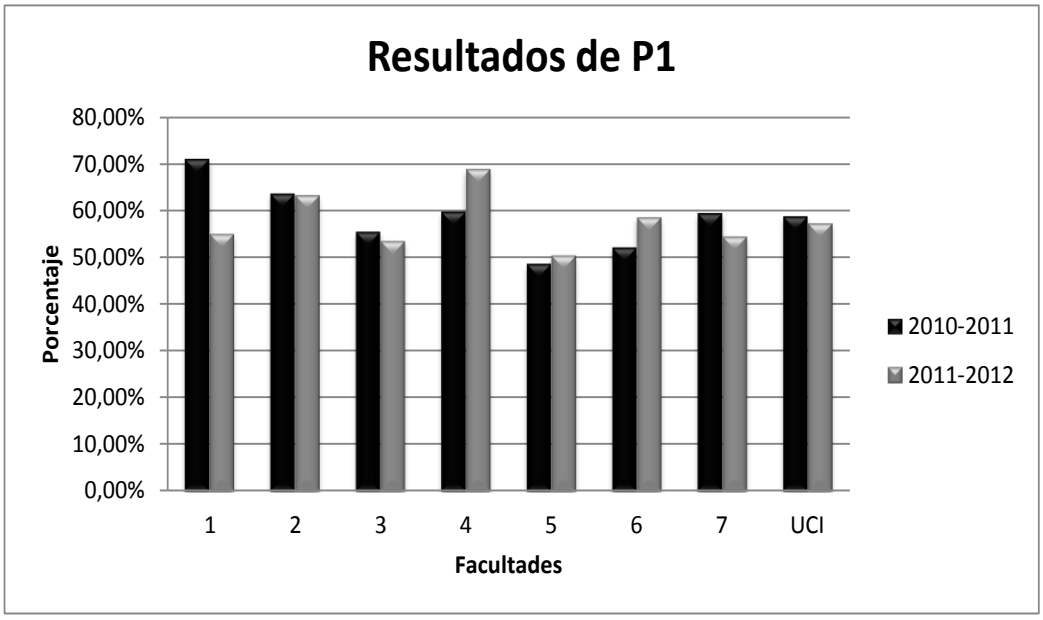
21	34.43 %	**Sí**
37	60.66 %	**No**
3	4.92 %	**(S / R)**

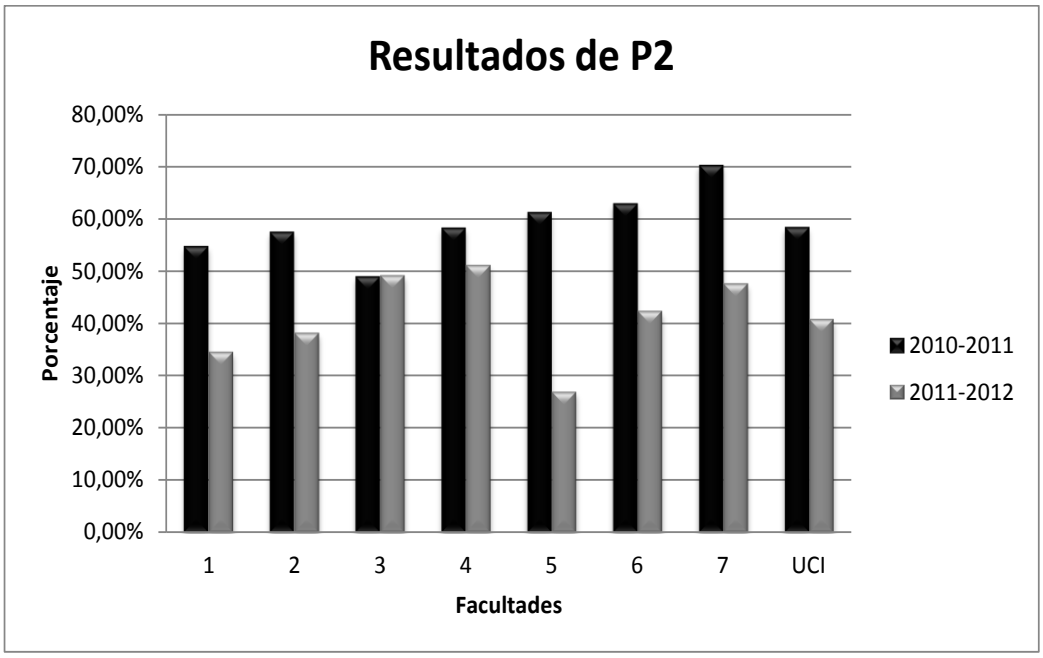
48. A continuación se muestran una serie de lenguajes de programación. Por favor especifique el nivel de dominio que tiene en ellos según la ayuda que se ofrece a continuación.

	1	2	3	4	5	S / R
C#	3 (4.92 %)	2 (3.28 %)	4 (6.56 %)	2 (3.28 %)	2 (3.28 %)	48 (78.69 %)
C++	1 (1.64 %)	1 (1.64 %)	6 (9.84 %)	6 (9.84 %)	2 (3.28 %)	45 (73.77 %)
Java	1 (1.64 %)	(0 %)	4 (6.56 %)	(0 %)	(0 %)	56 (91.8 %)
Phyton	3 (4.92 %)	(0 %)	(0 %)	(0 %)	(0 %)	58 (95.08 %)
VB	3 (4.92 %)	3 (4.92 %)	2 (3.28 %)	1 (1.64 %)	1 (1.64 %)	51 (83.61 %)
VB. Script	3 (4.92 %)	1 (1.64 %)	(0 %)	(0 %)	(0 %)	57 (93.44 %)
VB .NET	2 (3.28 %)	1 (1.64 %)	(0 %)	(0 %)	(0 %)	58 (95.08 %)
Object Pascal	3 (4.92 %)	(0 %)	1 (1.64 %)	1 (1.64 %)	(0 %)	56 (91.8 %)
Perl	3 (4.92 %)	(0 %)	(0 %)	(0 %)	(0 %)	58 (95.08 %)
Pascal	5 (8.2 %)	1 (1.64 %)	1 (1.64 %)	1 (1.64 %)	(0 %)	53 (86.89 %)
QBasic	3 (4.92 %)	(0 %)	(0 %)	(0 %)	(0 %)	58 (95.08 %)
Ensamblador	3 (4.92 %)	(0 %)	(0 %)	(0 %)	(0 %)	58 (95.08 %)
PHP	3 (4.92 %)	3 (4.92 %)	6 (9.84 %)	1 (1.64 %)	1 (1.64 %)	47 (77.05 %)
JavaScript	2 (3.28 %)	2 (3.28 %)	6 (9.84 %)	(0 %)	2 (3.28 %)	49 (80.33 %)
Fortran	3 (4.92 %)	(0 %)	(0 %)	(0 %)	(0 %)	58 (95.08 %)

Anexo 2. Resultados de la Disciplina Programación (2010-2012)







Anexo 3. Competencias de Programación identificadas en la UCI

Copa	Área	Responsable
Copa 50 Aniversario	UCI-MININT	Hamler Rodríguez González
Copa Void	Facultad 1	José Ernesto Torres Sanchez
Copa MaxCode	Facultad 1	José Miguel Argilagos Yis
Copa Ada Byron	Facultad 1	José Gustavo Suárez Matilla
Copa Olimpo	Facultad 2	Yaniel Alfredo Velázquez Bruceta
Copa Copa ByteCode	Facultad 3	Yadián Guillermo Pérez Betancourt
Copa Java	Facultad 3	Yadian Guillermo Pérez Betancourt
Copa TopKoder	Facultad 3	Yadián Guillermo Perez Betancourt
Copa Gauss	Facultad 4	Michael Orta Fleitas
Copa Arena	Facultad 6	Nelson Gonzalez Peñate
Copa Duke	Facultad 6	Yaima Fernández Segredo
Copa WFemale	Facultad 6	Julio César Desten Anaya
Copa Bit	Facultad 7	Dayán Roberto Pérez Barzaga
Copa Aniversario de la CPAV	CPAV	Yaniel Alfredo Velázquez Bruceta
Copa Euler	CPAV	Yaniel Alfredo Velázquez Bruceta
Copa Time vs Memory	CPAV	Yaniel Alfredo Velázquez Bruceta
Copa XXProgramming	Iniciativa Xtreme	José Ernesto Lara Rodríguez
Copa Xtreme Programming	Iniciativa Xtreme	José Ernesto Lara Rodríguez

Newbs		
Copa Xtreme Programming Local	Iniciativa Xtreme	José Ernesto Lara Rodríguez
Copa Xtreme Programming Pro	Iniciativa Xtreme	José Ernesto Lara Rodríguez
Copa Xtreme Programming (Xprog)	Iniciativa Xtreme	José Ernesto Lara Rodríguez
Copa Xtreme Programming Open Contest	Iniciativa Xtreme	José Ernesto Lara Rodríguez
Copa Xtreme Programming Master Challenge	Iniciativa Xtreme	José Ernesto Lara Rodríguez

Anexo 4. Cuestionario para la aplicación de la técnica de ladov

1. ¿Cree usted que el plagio es un comportamiento correcto?

   ___Sí ___No ____No Sé

2. ¿Cree usted necesario detectar plagio entre la soluciones enviadas a los jueces en línea?

   ___Sí ___No ____No Sé

3. ¿Cree usted que la dificultad en la detección de plagio hace necesaria la creación de un módulo que facilite el proceso?

   ___Sí ___No ____No Sé

4. ¿Cree usted que los *"include, import, using"* y otros fragmentos básicos de código en los ejercicios deben tenerse en cuenta para la detección?

   ___Sí ___No ____No Sé

5. Una vez vista la aplicación. ¿Qué usted cree del módulo para la detección de plagio en el Juez en Línea Académico?

   ___Me gusta mucho.

   ___No me gusta tanto.

   ___Me da lo mismo.

   ___Me disgusta más de lo que me gusta.

   ___No me gusta nada.

   ___No sé qué decir.

6. Una vez vista la aplicación. ¿Qué usted cree de la representación de los resultados del módulo para la detección de plagio en el Juez en Línea Académico?

___Me gusta mucho.

___No me gusta tanto.

___Me da lo mismo.

___Me disgusta más de lo que me gusta.

___No me gusta nada.

___No sé qué decir.

7. Una vez vista la aplicación. ¿Qué usted cree de la representación de las similitudes en el código utilizada por el módulo para la detección de plagio en el Juez en Línea Académico?

   ___Me gusta mucho.

   ___No me gusta tanto.

   ___Me da lo mismo.

   ___Me disgusta más de lo que me gusta.

   ___No me gusta nada.

   ___No sé qué decir.

8. Los algoritmos desarrollados para la detección de plagio en el Juez en Línea Académico se basan en tres criterios: el código fuente para reflejar la similitud estructural, la clasificación del problema teniendo en cuenta la similitud obvia en algoritmos clásicos como el Dijkstra y al usuario previniendo un comportamiento reincidente. ¿Cree usted que sea importante tener en cuenta algún otro criterio? Argumente su respuesta.

   Argumentación: _____

9. Cree usted que deba sancionarse al usuario reincidente en casos de plagio. ¿Qué tipo de sanción usted propone?

   Argumentación: _____

10. El módulo para la detección de plagio en el Juez en Línea Académico posee las funcionalidades fundamentales para facilitar el trabajo de detección a los profesores. ¿Considera usted que falte alguna funcionalidad? Argumente.

    Argumentación: _____