



Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

**Construcción de una aplicación web como sistema de
recomendación de problemas para programación competitiva.**

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela por los
Br. German Blandin
Br. Rodolfo Miquilarena
para optar al título de Licenciado en Computación

Tutor
Prof. Hector Navarro

Caracas, Mayo del 2017
Universidad Central de Venezuela




Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica


ACTA DE VEREDICTO


Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación de la Universidad Central de Venezuela, para evaluar el Trabajo Especial de Grado titulado **Construcción de una aplicación web como sistema de recomendación de problemas para programación competitiva**, presentado por los bachilleres Germán Blandin, cédula 21.283.057 y Rodolfo Miquilarena, cédula 20.244.091, como requisito académico para optar al título de Licenciado en Computación, dejamos constancia de lo siguiente:

1. Una vez leído el trabajo por los miembros del jurado, se fijó el día 19 de Mayo de 2017 a las 11:00 am, en la sala 1 del Centro de Computación Gráfica, Facultad de Ciencias de la Universidad Central de Venezuela, para que la aspirante presentara mediante una exposición oral y pública su contenido, respondiendo luego satisfactoriamente a las preguntas formuladas por el jurado.
2. Finalizada la defensa del Trabajo de Grado, el jurado decidió APROBARLO con una calificación de 20 puntos.
3. Para dar este veredicto, el jurado estimó que el trabajo examinado cumple con los requisitos exigidos para su aprobación en fe de lo cual se levanta la presente acta, el día 19 de Mayo de 2017.


Eugenio Scalise
C.I. 10.184.983

Jurado


Hector Navarro
13.113.800
Tutor


Rhadamés Carmona
C.I. 10.804.242

Jurado

Resumen

Construcción de una aplicación web como sistema de recomendación de problemas para programación competitiva.

German Blandin

Rodolfo Miquilarena

Prof. Hector Navarro, Tutor.

Universidad Central de Venezuela

El objetivo del presente trabajo de investigación es la implementación de una aplicación web dedicada al entrenamiento en programación competitiva haciendo uso de minería de datos y un sistema de recomendación que se encargue de sugerir ó recomendar problemas a los usuarios basados en las necesidades de los mismos, esto con el objetivo de lograr un mejor proceso de aprendizaje en el ámbito de la programación competitiva en los usuarios, sin la necesidad de un experto que lo asesore.

Para esto se implementó una aplicación web usando el framework Django y distintos métodos de recomendación, además de una implementación del algoritmo a priori, para esto se extrajeron datos de páginas web como SPOJ, UVA, Coj y Codeforces a través de técnicas de crawling y se procesaron estos datos para posteriormente realizar las recomendaciones. Finalmente se realizaron pruebas de rendimiento y de aceptación con el objetivo de mejorar la experiencia del usuario y determinar la calidad de la aplicación debido a la subjetividad de la misma.

Palabras claves: Programación Competitiva, sistemas de recomendación, minería de datos, aplicación web.

Germán: Le dedico a este trabajo a mis padres y a mi abuela Gladis que ya no es encuentra entre nosotros pero que siempre quiso ver que este día llegara.

Rodolfo: Le dedico este trabajo a mi abuelo José Paez porque siempre fue su sueño ver a uno de sus hijos ó nietos varones graduado y aunque no me podrá ver, haré que su sueño se cumpla. Le dedico este trabajo también a todos los competidores que sueñan con alcanzar la final mundial ACM, le dedico esto también a mis padres y espero que se sientan orgullosos de mí.

Agradecimientos

Germán: en primer lugar le agradezco a mis padres ya que por ellos es quien soy y por ellos estoy escribiendo estas palabras, ya que siempre me han dado su apoyo para seguir adelante, ¡¡GRACIAS!! A mi tía Liseth debido su apoyo, sin ella no hubiese podido realizar este trabajo. A todos aquellos que hayan ayudado a mi formación tanto de manera educativa como de manera personal, entre ellos destaco a los profesores que he tenido la oportunidad de recibir sus enseñanzas, como todos los compañeros y amigos que tuve desde el inicio de la carrera. Agradezco a mis compañeros y amigos Rodolfo Miquilarena y Samuel Nacache por ser mis compañeros de equipo, por todos aquellos buenos momentos que pasamos y todas las cosas que aprendimos en esos momentos. A Rodolfo agradecimientos aparte por invitarme a desarrollar junto a el, el trabajo especial de grado cuya idea me pareció excelente. A mi Tutor, profesor en varias ocasiones y coach Hector Navarro por todas las enseñanzas en todos las ocasiones donde recibí su guía. Por ultimo a todos aquellos que probaron e hicieron criticas a nuestro trabajo especial de grado con la intención de que fuera mejor, en especial a Alfredo Viera que apporto bastantes ideas durante el desarrollo y mostrara su entusiasmo durante el mismo.

Rodolfo: Quiero agradecer a mi Mamá por todo su apoyo y comprensión durante toda mi vida y su gran cariño, a mi Papá, a mis hermanos y a Gioconda por su apoyo y su compromiso para que yo pudiera estudiar y culminar mis estudios en la UCV y también para estar ahí siempre que los necesitara, a mis abuelas por todas las historias de éxito de la familia que me contaron, a mi abuelo por siempre haberme dado todo y siempre estar ahí para darme consejo y comprensión, a mi tía María Angelica por dejarme quedarme con ella los días previos a los parciales, a mi primo Luis Alfredo y a María Teresa por siempre estar ahí para darme algún apoyo y hacerme pasar un buen rato, a mi primo José Miguel y a mi amigo Raúl Sanchez por ser otras voces de apoyo, a todos los competidores de programación competitiva que me motivaron a hacer este trabajo, a todos los profesores de la universidad de los que aprendí muchísimo, a mi amiga Barbara Rojas por siempre ser una fuente de apoyo y comprensión en todo momento, a Germán Blandin por unirse a mí en este trabajo especial de grado que fue una experiencia que nunca olvidaré junto con todos los años que competimos juntos en ACM junto a nuestro gran amigo Samuel Nacache, a toda la gente que probó la aplicación y votó en las encuestas, a todos los compañeros y amigos que tuve en la universidad, y a nuestro tutor Héctor Navarro por ser un amigo y mentor para mí desde que inició mi quinto semestre de carrera.

Índice general

Resumen	I
Agradecimientos	III
Introducción	VIII
1. Problema y Objetivos	1
1.1. Problema	2
1.2. Objetivo General	2
1.3. Objetivos Específicos	3
2. Marco Teórico	4
2.1. Programación competitiva	4
2.2. Problemas	5
2.3. Técnicas y recursos	6
2.3.1. Ad-hoc	7
2.3.2. Recursión y backtracking	7
2.3.3. Estructuras de datos	7
2.3.4. Algoritmos Greedy	10
2.3.5. Programación Dinámica	10
2.3.6. Divide y Conquista	11
2.3.7. Algoritmos sobre grafos	11
2.3.8. Algoritmos matemáticos	12
2.3.9. Geometría computacional	13
2.3.10. Algoritmos sobre strings	13
2.4. Competencias	14
2.4.1. ACM International Collegiate Programming Contest (ACM- ICPC)	15
2.4.2. International Olympiad in Informatics (IOI)	16
2.4.3. Top Coder Single Round Match	17
2.4.4. Codeforces	19
2.4.5. Google Code Jam	20
2.4.6. Facebook Hacker cup	21
2.5. Jueces y veredictos	22

ÍNDICE GENERAL

2.5.1.	BOCA	23
2.5.2.	Kattis	23
2.5.3.	DOMjugde	24
2.5.4.	UVA online jugde	24
2.5.5.	SPOJ	24
2.5.6.	Moose	25
2.6.	Minería de datos	26
2.7.	Algoritmos de aprendizaje	28
2.7.1.	Aprendizaje supervisado	28
2.7.2.	Aprendizaje no supervisado	29
2.7.3.	Aprendizaje semi-supervisado	29
2.7.4.	Aprendizaje por refuerzo	29
2.8.	Técnicas de Minería de datos	29
2.8.1.	Clustering	30
2.8.2.	Clasificación	31
2.8.3.	Reglas de asociación	31
2.8.4.	Regresión	31
2.8.5.	Algoritmo Apriori	32
2.9.	Sistemas de recomendación	32
2.9.1.	Entradas y Salidas	34
2.9.2.	Métodos de Generación de Recomendaciones	35
2.9.3.	Grado de Personalización	36
2.9.4.	Modelo de los sistemas de recomendación	37
2.9.4.1.	Matriz de Utilidad	37
2.9.4.2.	Aplicaciones de los sistemas de recomendación	37
2.9.4.3.	Manejo de la matriz de utilidad	38
2.9.5.	Filtrado basado en contenido	39
2.9.5.1.	Perfil de artículos	39
2.9.5.2.	Obtención de características de los artículos a través de etiquetas	39
2.9.5.3.	Representación de perfil de artículos	40
2.9.5.4.	Perfil de Usuarios	40
2.9.5.5.	Algoritmos de clasificación	41
2.9.6.	Filtrado Colaborativo	41
2.9.6.1.	Algoritmo de los K-Vecinos	42
2.9.7.	Métricas para recomendaciones	43
2.10.	Aplicaciones Web	45
2.10.1.	Arquitectura de las Aplicaciones Web	45
2.10.1.1.	Modelo-Vista-Controlador	46
2.11.	Capas de la Arquitectura	48
2.11.1.	Servidor de bases de datos	48
2.11.2.	Servidor Web	49
2.11.3.	Lenguajes de Programación	50

ÍNDICE GENERAL

2.11.3.1. Python	50
2.11.3.2. JavaScript	51
2.11.4. Lenguajes de Marcado	51
2.11.4.1. HTML	51
2.11.4.2. CSS	52
2.12. Frameworks	53
2.12.1. Django	53
2.12.2. Bootstrap	54
2.12.3. Scrapy	55
2.13. Web scraping	55
2.13.1. Técnicas	56
2.14. Librerías y paquetes	58
2.14.1. jQuery	58
2.14.2. NumPy y SciPy	59
3. Trabajos previos	60
3.1. Páginas de Programación competitiva	60
3.1.1. Top Coder	61
3.1.2. Spoj	62
3.1.3. A2 Online Judge	63
3.1.4. USACO	64
3.1.5. Uhunt	65
3.1.6. COJ	66
3.1.7. Codeforces	66
3.2. Páginas web con sistemas de recomendación	68
3.2.1. Amazon	68
3.2.2. Hackerrank	68
3.2.3. Youtube	69
3.2.4. LinkedIn	69
3.2.5. NetFlix	69
4. Diseño	71
4.1. Metodología	71
4.2. Arquitectura	72
4.3. Modelo de datos	73
4.3.1. Cuentas de Usuario	74
4.3.2. Problemas y Categorías	75
4.3.3. Resolución de Problemas	76
4.3.4. Habilidades	78
4.3.5. Tutoriales	79
4.4. Procesos y Vistas	79
4.4.1. Registro	80
4.4.2. Inicio de sesión	80

ÍNDICE GENERAL

4.4.3.	Cerrar sesión	81
4.4.4.	Perfil de Usuario	81
4.4.5.	Clasificación	84
4.4.6.	Buscador de problemas	85
4.4.7.	Tutoriales	86
4.4.8.	Recomendaciones	87
5.	Implementación	90
5.1.	Back-end	90
5.1.1.	Crawling	91
5.1.1.1.	Crawler SPOJ	91
5.1.1.2.	Crawler UVA	94
5.1.1.3.	Crawler COJ	96
5.1.1.4.	Crawler Codeforces	99
5.1.2.	Mapping	102
5.1.3.	Clasificación	102
5.1.4.	Calculo de habilidades	103
5.1.5.	Recomendaciones	104
5.1.5.1.	Recomendador utilizando el algoritmo apriori	104
5.1.5.2.	Recomendador utilizando filtros colaborativos	106
5.1.5.3.	Recomendador basado en contenido	108
5.2.	Front-end	111
5.2.1.	Interfaz	111
5.2.2.	Log-in con Google y Facebook	113
5.3.	Seguridad	113
6.	Pruebas y resultados	115
6.1.	Pruebas eficiencia en tiempo	115
6.1.1.	Prueba uno	116
6.1.2.	Prueba dos	117
6.1.3.	Conclusiones	118
6.2.	Pruebas de aceptación	119
6.2.1.	Conclusiones	126
7.	Conclusiones y trabajos a futuro	127
7.1.	Conclusiones	127
7.2.	Trabajos a futuro	130
	Bibliografía	131

Introducción

Con el desarrollo de las ciencias de la computación, ha venido en aumento el número de competencias de programación. Comenzando con las competencias ACM International College Programming Contest hacia finales de los años 70, luego con la Olimpiada Internacional de Informática hacia finales de los 80, el número de competencias ha ido en aumento así como la dificultad de las mismas. Tal y como se ha elevado la popularidad y la dificultad de estas competencias, también ha elevado el nivel que un competidor debe tener para participar adecuadamente en este tipo de eventos de gran importancia.

Además de esto, a menudo es necesario seleccionar una entre varias alternativas sin tener un conocimiento exacto de cada una de ellas. En estas situaciones, la decisión final puede depender de las recomendaciones de otras personas. Los sistemas de recomendaciones son herramientas cuyo objetivo es asistir a los usuarios en sus procesos de búsqueda de información, ayudando a filtrar objetos de información, usando recomendaciones propuestas sobre esos objetos. Dichas recomendaciones se generan a partir de las opiniones proporcionadas por otros usuarios sobre esos objetos en búsquedas previas o bien a partir de las preferencias del usuario objeto de la recomendación, dando lugar a los dos grandes grupos de sistemas de recomendación, los colaborativos y los no colaborativos o basados en contenidos. El uso de estos sistemas se está poniendo cada vez más de moda debido a su utilidad para evaluar y filtrar la gran cantidad de información disponible en la Web para asistir a

INTRODUCCIÓN

los usuarios en sus procesos de búsqueda y recuperación de información.

Débito a esto, el siguiente trabajo plantea la investigación de estos dos tópicos, la programación competitiva y los sistemas de recomendación, con el objetivo de aprovechar todas las características de este último y utilizarlas con el fin de sugerir problemas a los competidores para que esto que ayude a mejorar su desempeño en las competencias de programación y sus habilidades.

En el primer capítulo se presenta el planteamiento del problema y los objetivos tanto general como específicos, orientados a la construcción de una aplicación web dedicada al entrenamiento de la programación competitiva, implementando en dicha aplicación un sistema de recomendación que se encargue de sugerir o recomendar problemas a los usuarios basados en las necesidades de los mismos, esto con el objetivo de lograr un mejor progreso de aprendizaje en el ámbito de la programación competitiva en los usuarios.

En el capítulo 2 se incluye el marco teórico en el cual nos basamos para realizar la implementación de la aplicación web que surge de nuestra propuesta. En este capítulo se explica lo que es la programación competitiva y sus diferentes aspectos a tener en cuenta tales como las técnicas básicas para poder competir, así como explicar la temática de los jueces y exponer algunos ejemplos variados. También hubo enfoque en especificar que es un sistema de recomendación, los tipos de estos sistemas, los aspectos a tener en cuenta al momento de su implementación junto con los algoritmos y técnicas de aprendizaje, minería de datos e inteligencia artificial usada en este tipo de sistemas. Por último se explica cómo funciona una aplicación web, ya que los sitios de entrenamiento de programación competitiva están hechos en su mayoría con este tipo de aplicación, también se define algunas tecnologías usadas comúnmente en la implementación de las mismas que además serán usadas para llevar a cabo la propuesta de tesis, incluyendo los distintos frameworks, lenguajes y

INTRODUCCIÓN

librerías a utilizar.

En el capítulo 3 hablaremos sobre trabajos previos que se han realizado tanto en el ámbito de la programación competitiva, como de los sistemas de recomendación, en él, se exponen sitios dedicados al entrenamiento para estas competencias, ya sea por medio de ofrecer competencias semanales, gran variedad de problemas para resolver y aprender de ellos, o presentar tutoriales. Además se hace una breve mención a algunos sitios ajenos a la programación competitiva, pero que hacen uso de algún sistema de recomendación para llevar a cabo sus objetivos.

En el capítulo 4 se presenta el diseño utilizado para la realización de la aplicación. En esta detallamos la arquitectura usada en la aplicación, el modelo de datos y los procesos que son llevados a cabo por la misma.

El capítulo 5 trata sobre la implementación de la aplicación, explicando detalladamente las 2 partes esenciales de la aplicación como lo son el back-end y el front-end. En el back-end se menciona la extracción de los datos a través de los crawler, su posterior procesamiento y clasificación para finalmente realizar las recomendaciones con los distintos métodos allí expuestos. Por último en el front-end se explica la interfaz y algunas funciones que hacen uso de tecnologías como JavaScript y jQuery.

Por último en el capítulo 6 se exponen los resultados de las pruebas de rendimiento y aceptación realizadas en la aplicación.

1

Problema y Objetivos

Hoy en día la programación competitiva no tiene límites. Se ha convertido en un verdadero deporte intelectual visto por las principales corporaciones de tecnologías de información y todos aquellos que están interesados en el desarrollo de las tecnologías de computación. Esto se debe a que la programación competitiva ayuda a desarrollar aspectos como, el trabajo en equipo, mejorar las habilidades cognitivas, la habilidad investigativa, además de que resuelve problemas reales, destacar en estos concursos es un gran atractivo para las grandes empresas. No es sorpresa que empresas como Microsoft, Google o Facebook extiendan ofertas de trabajo a los mejores competidores de la ACM-ICPC, o que incluso organicen sus propias competencias, como el Code Jam de Google o el Hacker Cup de Facebook, para reclutar a los mejores programadores.

Es más, las entrevistas para estas empresas usualmente contienen problemas muy sencillos para el programador competitivo experto. Libros completos se dedican a los problemas de estas entrevistas, los cuales podrían resultar triviales después de adentrarse en el mundo de la programación competitiva.

1.1. Problema

Actualmente la problemática en Venezuela es que se carece de suficientes personas a nivel nacional con los conocimientos para impartirlo de una manera eficaz a la mayoría de las universidades venezolanas, además se carece de recursos para invitar profesionales de otras partes del mundo para que impartan clases, por lo general solo algunas universidades tienen los recursos, así que Venezuela está en una gran desventaja ante otros países que participan en estas competencias.

La mayoría de las páginas web dedicadas a programación competitiva no suelen ser suficiente para que un competidor desarrolle adecuadamente las habilidades necesarias para resolver problemas de programación, por lo general se necesita de un guía o persona dedicada que ayude a los competidores para facilitar su aprendizaje haciendo uso de estos sitios web.

Por estas razones, se propone el desarrollo de un sistema de entrenamiento de programación competitiva, en donde el competidor sea capaz de avanzar sin necesidad de expertos que lo asesoren.

1.2. Objetivo General

Debido a esta problemática el objetivo es realizar una aplicación web haciendo uso de un sistema de recomendación que facilite el aprendizaje y el desarrollo de las habilidades de cada competidor de una manera sencilla y confiable. Para lograrlo la aplicación debe cumplir con los siguientes objetivos específicos:

1.3. Objetivos Específicos

- Extraer de sitios web externos problemas de programación, de manera automática, utilizando herramientas de Web Scraping, más específicamente Crawling, para así realizar recomendaciones.
- Implementar un sistema de recomendación basado en contenido para recomendar problemas, basándose en las necesidades del usuario.
- Implementar un sistema de recomendación colaborativo, para hacer recomendaciones de problemas que otros usuarios similares hayan resuelto.
- Poner a disposición una biblioteca virtual que posea tutoriales de los temas referentes a la programación competitiva.
- Determinar y asignar características a los problemas, tales como la dificultad y las categorías, utilizando distintos métodos, como cálculo en base a estadísticas ó votación de los usuarios.
- Establecer compatibilidad multi-plataforma, para un buen funcionamiento en los navegadores mas usados.
- Ofrecer una interfaz agradable al usuario, que pueda ser manejada de manera sencilla y eficaz.
- Realizar pruebas usando distintos métodos de recomendación, para garantizar que se cumpla con las especificaciones del sistema.
- Realizar pruebas generales para garantizar el buen funcionamiento del sistema.

2

Marco Teórico

En este capítulo se hablará de las bases teóricas que sustentan el presente trabajo, desde la definición de la programación competitiva, pasando por minería de datos y sistemas de recomendación, hasta las distintas herramientas que se utilizaron para realizarlo.

2.1. Programación competitiva

La directiva central de la programación competitiva es “Dado problemas bien conocidos sobre Ciencias de la computación resolverlos lo mas rápido posible”, el termino “Problemas bien conocidos”, implica que en la programación competitiva se tratan con problemas que ya están resueltos y no se investigan los mismos (donde no se conoce la solución). Algunas personas (al menos el autor del problema) conocen la solución y han resuelto el problema antes. Para “resolver” los problemas implica tener los conocimientos en ciencias de la computación a un cierto nivel para que se pueda producir un código que lo pueda resolver, al menos en términos de conse-

guir el mismo resultado que el autor en los casos de prueba secretos en un tiempo estipulado. La necesidad de resolver el problema lo antes posible es donde radica el elemento competitivo, siendo un objetivo muy natural en el comportamiento humano [1].

La programación competitiva es un tipo de concurso en el que varios programadores se someten a una prueba en un tiempo limitado para resolver la mayor cantidad de problemas de ámbito computacional posibles. La idea es escribir un programa que, dadas ciertas especificaciones, pueda resolver cualquier caso propuesto por los jueces, esto con la intención de verificar que dicho programa resuelva el problema en cualquier instancia posible. Además de esto se tienen otras restricciones como que el programa solo tiene unos cuantos segundos para encontrar la respuesta o se tienen recursos limitados como la memoria [2].

A continuación se describirán algunos conceptos y términos necesarios para conocer con más de detalle la programación competitiva, desde como son los problemas, las técnicas que se usan, las competencias más conocidas en la actualidad y la importancia en las ciencias de la computación.

2.2. Problemas

Los problemas usados en competencias de programación son de ámbito computacional y algorítmico, orientados al diseño e implementación de algoritmos eficientes que los resuelvan, también se suelen presentar problemas que requieren el desarrollo de habilidades como el análisis matemático. Los problemas son distintos dependiendo de la competencia y su formato, pero por lo general suelen ser variados en cuanto a su dificultad y su tipo, están basados en situaciones o problemas de la vida real, cosas como optimizar horarios del metro, control de tráfico, análisis de

CAPÍTULO 2. MARCO TEÓRICO

circuitos lógicos, simulaciones de distintos tipos, estimación de reservar de petroleo, entre otras [3].

La descripción del problema consiste en un texto donde se explica el mismo, luego se especifica como serán los datos de entrada que deberá recibir el programa, de igual manera se explica como tiene que ser el formato de la salida donde estará la respuesta otorgada por el programa de tal manera que sea aceptado [3]. Es importante mencionar que en estas competencias por lo general se asumen los conocimientos básicos para el manejo de los lenguajes de programación, conocimiento del uso de estructuras de datos y funcionalidades provistas por los mismos, habilidades de depuración de código, eficiencia, así como utilización de librerías particulares para algún problema [4].

Los problemas de programación generalmente se clasifican de acuerdo a la técnica que sea necesaria para su resolución, a continuación se darán a conocer las técnicas más comunes en programación competitiva.

2.3. Técnicas y recursos

Una técnica de programación es aquel método o conjunto de métodos utilizados para resolver un problema en específico, por lo que podemos decir que un problema de programación puede ser clasificado teniendo como referencia la técnica mediante la cual puede resolverse. Existe una enorme variedad de técnicas y recursos para la programación competitiva, profundizaremos en las más importantes ya que el resto de técnicas ó recursos posibles son directamente derivados de estas.

2.3.1. Ad-hoc

Ad-hoc no es como tal una técnica o un recurso, es la clasificación que se le da a un problema cuando no puede resolverse con una técnica general pues solo puede resolverse utilizando un algoritmo específicamente creado para resolver un problema en particular[1].

2.3.2. Recursión y backtracking

- Recursión: Es aquel proceso mediante el cual una función se llama a si misma con el objetivo de resolver algún problema en específico, la llamada hace que se cree un nuevo objeto en la pila del sistema, el cual es un clon de la función original con algún parámetro modificado[1].
- Backtracking: Es una búsqueda exhaustiva y recursiva, en otras palabras es una búsqueda que busca la solución a un problema probando todas las combinaciones posibles, una vez llega a una combinación inválida o a una solución al problema se rompe la recursión en ese estado para probar nuevas combinaciones[1].

2.3.3. Estructuras de datos

Una estructura de datos es aquella creada con el fin de almacenar datos, ya sea para almacenar en mayor volumen ó en generar un acceso más rápido a los datos, estas pueden dividirse en estructuras estáticas y estructuras dinámicas

- Estructuras estáticas:

Una estructura estática es aquella que trabaja en base a una memoria que se asigna antes de iniciar el programa y que no cambia de tamaño durante su ejecución. Entre ellas se encuentran:

CAPÍTULO 2. MARCO TEÓRICO

- Arreglos: Es la forma más simple de una estructura de datos, se basa en elementos almacenados de manera secuencial en memoria, que puede ser accedido por medio de índices[5].
 - Tabla Hash: Es una estructura de datos la cual almacena un conjunto de elementos similar a un arreglo, pero la forma de acceder a dichos elementos viene dada por una función especial llamada función Hash[6].
 - Disjoint-Set o Union Find: Es una estructura de datos que busca simular el comportamiento de los conjuntos disjuntos, se utilizan para determinar de manera eficiente si dos elementos pertenecen al mismo conjunto ó a conjuntos distintos. Esta tiene 2 operaciones básicas, Union, la cual une 2 elementos en el mismo conjunto, y Find que busca el conjunto al cual pertenece un elemento.[1].
 - Sparse Table: Es una estructura de datos que permite la resolución de consultas del tipo: “mínimo elemento en un rango” de manera muy eficiente, se crea en base a un pre-procesamiento previo de los datos[1].
 - Binary Indexed Tree (BIT): o también conocido como árbol de Fenwick, es una estructura de datos muy útil para la implementación de tablas de frecuencias acumuladas que pueden cambiar sus valores en ejecución[1].
 - Árbol de segmentos (ST): Es una estructura de datos que se basa en un esquema de Divide y Vencerás, es un árbol binario completo en el que cada nodo representa un rango, dicho rango se divide entre sus dos hijos hasta que quede un único elemento[1].
 - Suffix Array: Es aquella estructura que almacena todos los sufijos de una cadena de caracteres, para posteriormente usarlos de distintas maneras para obtener respuestas eficientes a ciertas consultas[1].
- Estructuras dinámicas:

CAPÍTULO 2. MARCO TEÓRICO

Una estructura de datos dinámica es aquella que no tiene una memoria pre-asignada por lo que puede añadir o eliminar elementos cuando se necesite, es decir la cantidad de memoria que usa puede cambiar en tiempo de ejecución.

- Arreglos dinámicos: Similares a los arreglos, pero pueden expandir su capacidad de almacenamiento en tiempo de ejecución [5].
- Cola: Una cola es una estructura que está basada en el esquema FIFO (First In First Out, en español Primero en Entrar Primero en Salir)[6].
- Pila: Una estructura que está basada en el esquema LIFO (Last in First Out, en español: Ultimo en Entrar Primero en Salir)[6].
- Árbol binario de búsqueda: es una estructura de tipo árbol que guarda elementos de manera ordenada, la estructura cuenta con dos apuntadores, hijo izquierdo y derecho, decimos que un elemento está en el sub-árbol izquierdo si la clave del nodo a insertar es menor que la clave del nodo actual, y a la derecha si es mayor, si ninguna de estas condiciones se cumple entonces el elemento ya existe en el árbol[5].
- Árbol binario de búsqueda balanceado: Similar a un árbol binario de búsqueda, no obstante en esta estructura se aplican algoritmos conocidos para mantener la altura del árbol lo más baja posible[5].
- Quadtree: Similar a un árbol binario pero cada nodo posee cuatro hijos, se utiliza para representar planos y cuadrantes ya que utiliza claves compuestas por dos elementos en lugar de uno[7].
- Min-Heap: Es una estructura de tipo árbol binario que mantiene la condición de que ambos hijos sean menores a su padre, por tanto el elemento mayor está situado en la raíz, con diversas operaciones podemos extraer y agregar nuevos elementos[5].
- Árbol k-dimensional (KD): Es una versión generalizada del Quadtree, uti-

lizada para almacenar puntos representados en cualquier dimensión[8].

- **Árbol Trie:** es una estructura de datos que permite almacenar palabras de manera que cada nodo representa un carácter de una palabra, cabe destacar que debido a su implementación si se insertan palabras con prefijos en común no se crean nuevos nodos, ayudando así a ahorrar mas de memoria y permitiendo obtener ciertas propiedades bastante útiles[1].

2.3.4. Algoritmos Greedy

Un algoritmo greedy, también conocido como algoritmo voraz, es una técnica que se basa en elegir la mejor solución de manera local, y así eventualmente llegar a una solución global[1].

2.3.5. Programación Dinámica

La Programación Dinámica (DP), es una técnica de programación que se basa en un esquema bottom-up, lo que significa que construimos una solución partiendo de soluciones óptimas más pequeñas, y que combinándolas podrán crear una solución final óptima, cabe destacar que estas soluciones óptimas previas son calculadas una sola vez[1].

El DP tiene una característica extra que es el pre-computo, esto se basa en que el DP solo tiene que ser ejecutado una vez para obtener todas las soluciones posibles de ciertos problemas, esto se logra utilizando arreglos ó estructuras de datos para “recordar” cosas ya calculadas con anterioridad[1].

2.3.6. Divide y Conquista

Divide y Conquista es una técnica utilizada para convertir un problema complejo en varios problemas simples, es decir lo “dividimos”, para luego resolver estas instancias más pequeñas que unificadas crearán la solución al problema original y así “conquistamos”. [1]

2.3.7. Algoritmos sobre grafos

- Algoritmo de BFS: El algoritmo de búsqueda en anchura (BFS) es un algoritmo que consiste en recorrer un grafo desde un nodo inicial, utilizando una cola como medio para saber cual es el siguiente nodo a visitar, de esta forma recorreremos por niveles de adyacencia, es decir si todas las aristas tienen la misma ponderación siempre visitaremos el nodo más cercano posible en ese paso del BFS y también con el menor número de nodos intermedios posible [1].
- Algoritmo de DFS: El algoritmo de búsqueda en profundidad (DFS) es un algoritmo que consiste en recorrer un grafo desde un nodo inicial, utilizando una pila como medio para saber cual es el siguiente nodo a visitar, entonces siempre llegaremos a un punto donde no podamos apilar nuevos nodos para empezar a recorrer otras adyacencias válidas de nodos anteriores [1].
- Algoritmo de Dijkstra: O algoritmo del camino más corto, fue propuesto por Edsger Dijkstra, se utiliza una lógica similar al BFS pero utilizando un heap de elemento mínimo, con esto se garantiza que cada paso para llegar de un nodo a otro iniciando en un nodo en particular es óptimo [1].
- Algoritmo de Kruskal: Es un algoritmo que calcula el minimum spanning tree ó árbol de expansión mínima, se basa en ordenar las aristas del grafo de tal manera que siempre obtengamos la arista de peso mínimo que no forme ciclo

CAPÍTULO 2. MARCO TEÓRICO

en el grafo formado por las aristas extraídas[1].

- Algoritmo de Prim: Es un algoritmo que calcula el minimum spanning tree ó árbol de expansión mínima, se trata de un BFS que opera extrayendo siempre la arista de peso mínimo que no crea un ciclo al combinarla con otras aristas que fueron extraídas[1].
- Algoritmo de Tarjan: Utilizado para calcular las componentes fuertemente conexas en un grafo dirigido , una componente fuertemente conexa es aquella en la que sin importar en que nodo iniciemos un recorrido siempre podremos recorrer todos los demás nodos de la componente en cualquier secuencia[1].
- Puntos de articulación: Son aquellos nodos que si no existieran en un grafo no dirigido causarían que dicho grafo fuese desconexo, este algoritmo utiliza un DFS para detectar los puntos de articulación[1].
- Algoritmo de Edmons Karp: Es un algoritmo de BFS modificado que encuentra el máximo flujo que puede pasar por un grafo dirigido dado que este posee un límite máximo de elementos (flujo) que pueden transitar por cada arista[1].
- Algoritmo de Kuhn: Es un algoritmo de DFS modificado que encuentra el máximo cardinality bipartite matching ó el máximo número de parejas que se pueden formar dado dos grupos y ciertas conexiones representados por un grafo que es en la mayoría de los casos dirigido[1].
- Algoritmo de Kahn: Un BFS modificado para encontrar un ordenamiento topológico de mínimo costo. [1]

2.3.8. Algoritmos matemáticos

Todos aquellos algoritmos que tienen alguna relación con la aritmética, probabilidad, funciones, números primos, combinatorias, y optimizaciones; Son conocidos como algoritmos matemáticos, y tienen una enorme importancia en las competen-

CAPÍTULO 2. MARCO TEÓRICO

cias de programación, ya que su dificultad no radica en el diseño del algoritmo, sino en el análisis y la comprensión de la teoría detrás de cada una de las ramas de la matemática. Por lo general estos problemas se resuelven mediante un análisis previo, resultando así en un algoritmo que puede ser bastante corto y sencillo de implementar[1].

2.3.9. Geometría computacional

La geometría computacional se diferencia de los algoritmos matemáticos, ya que aquí, vemos algoritmos mucho más extensos y todos trabajan sobre esta rama de la matemática, existen muchos algoritmos utilizados en esta clasificación que son representaciones de lo que se puede hacer en el plano o el espacio, lo que se traduce en poder representar figuras, puntos, líneas, segmentos, áreas entre otras cosas en el computador, los algoritmos geométricos son en ocasiones bastante complejos[1].

Entre ellos destaca la Convex Hull ó Cápsula Convexa , la cual genera un polígono convexo con la menor cantidad de puntos posibles de tal manera que encierre a todos los puntos en el plano/espacio.

2.3.10. Algoritmos sobre strings

Los algoritmos sobre strings son aquellos algoritmos específicamente diseñados para el tratamiento de strings ó cadenas de caracteres, su funcionalidad principal es resolver los problemas de una manera eficiente ó que de otra manera no podrían resolverse.

En esta categoría destacan los autómatas, los cuales nombraremos aquí:

- Autómata finito determinista: Aquel que verifica si una palabra o frase es aceptada por una gramática tipo 3, muy frecuente en competencias regionales eu-

ropeas y asiáticas[9].

- Autómata de pila: Aquel que verifica si una palabra o frase es aceptada por una gramática de tipo 2 en adelante, rara vez aparece en la programación competitiva, pero han habido problemas de este tipo[9].
- Máquina de Turing: Aquel que verifica si una palabra o frase es aceptada por una gramática de tipo 0 en adelante, muy raro en la programación competitiva[9].
- Algoritmo de Knuth Morris Pratt (KMP): Encuentra las sub cadenas comunes entre dos cadenas, puede realizar este procedimiento con mayor eficiencia a un algoritmo fuerza bruta.[1]
- Algoritmo de Aho Corasick: Es una versión optimizada de KMP cuya funcionalidad es sumamente compleja y bastante raro de ver en la programación competitiva.[1]

2.4. Competencias

En la programación competitiva existen varios tipos de competencias que varían tanto en formato como en el tipo de problemas que podemos encontrar en ellas, las más conocidas son las competencias presenciales como ACM International Collegiate Programming Contest (ICPC) y la International Olympiad in Informatics (IOI), sin embargo también existen muchas competencias por internet como TopCoder Single Round Match (SRM), Codeforces, Google Code Jam, Facebook Hacker Cup, entre otras [4].

2.4.1. ACM International Collegiate Programming Contest (ACM-ICPC)

ACM International Collegiate Programming Contest (ICPC) es una competencia de programación de varios niveles, basado en equipos que opera bajo los auspicios de la ACM y con sede en la Universidad de Baylor. El concurso consiste en una red global de universidades que organizan competiciones regionales para seleccionar a los equipos que participarán en la final mundial ACM-ICPC. La participación ha crecido a varias decenas de miles de los mejores estudiantes y profesores la disciplina, lo que se traduce en casi dos mil quinientas treinta y cuatro universidades de más de ciento un países en seis continentes. El concurso fomenta la creatividad, el trabajo en equipo y la innovación en la creación de nuevos programas de software, y permite a los estudiantes poner a prueba su capacidad para actuar bajo presión. Es el concurso más antiguo, más grande y de mayor prestigio en el mundo de la programación [10].

Es una de las competencias más importantes en el tema de la programación competitiva, su misión es ofrecer a los estudiantes universitarios la oportunidad de interactuar con los estudiantes de otras universidades para afilar y demostrar sus habilidades de programación y de trabajo en equipo. Este certamen ofrece una base para la ACM, la industria y el mundo académico para fomentar y centrar la atención pública en la próxima generación de profesionales en ciencias de la computación que persiguen la excelencia[11].

Este evento se realiza anualmente y consta de dos fases entre equipos formados por tres estudiantes que representan a una institución de educación superior y un entrenador de la misma institución (que no puede ser un competidor). Durante la primera fase los equipos compiten en concursos regionales que se celebran en todo

CAPÍTULO 2. MARCO TEÓRICO

el mundo entre septiembre y diciembre de cada año. Los ganadores de cada región asisten a la segunda fase que corresponde a la final mundial de ACM-ICPC que se celebra normalmente entre marzo y junio[12].

2.4.2. International Olympiad in Informatics (IOI)

La Olimpiada Internacional de Informática (IOI) es uno de los concursos de informática más reconocidos en el mundo. Las tareas de la competencia son de naturaleza algorítmica; sin embargo, los concursantes tienen que mostrar habilidades básicas como el análisis de problemas, diseño de algoritmos y estructuras de datos, programación y pruebas. Los participantes son siempre jóvenes promesas para el mundo de la computación.

El objetivo principal del IOI es estimular el interés por la informática (ciencias de la computación) y la tecnología de la información. Otro objetivo importante es reunir, desafiar, y dar reconocimiento a los jóvenes estudiantes que son los mejores en el mundo de la informática (ciencias de la computación), y para fomentar la amistad entre estos estudiantes de diversas culturas.

La primera IOI se celebró en Bulgaria en 1989. Fue propuesto por la Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO) y es uno de las cinco Olimpiadas Internacionales de Ciencia para estudiantes de escuelas secundarias de todo el mundo.

La IOI se organiza anualmente en y por uno de los países participantes. Cada país participante normalmente envía una delegación de cuatro estudiantes y dos adultos. Muchos países organizan una olimpiada nacional en informática para seleccionar los estudiantes que envían a la IOI.

Los estudiantes compiten de forma individual, con un máximo de cuatro estudiantes que compiten por cada país participante (con más de 80 países reciente-

mente). Los estudiantes compiten individualmente y tratan de maximizar su puntuación mediante la resolución de un conjunto de tareas de informática durante dos días de competencia. Eventos culturales y recreativos se organizan en los días restantes[13].

Es importante señalar que según el reglamento un concursante es un estudiante que se matriculó en una escuela de educación secundaria, en el país que representan, durante el período de septiembre a diciembre en el año anterior de la IOI correspondiente, y no es mayor a veinte años en el primero de julio del año que se celebra la IOI correspondiente. [14]

2.4.3. Top Coder Single Round Match

En esta competencia de programación, llamada "Single Round Match"(SRM) es una competencia online con límite de tiempo donde todos los concursantes se miden entre ellos intentando resolver los mismos problemas con las mismas limitaciones. Las competencias están disponibles en cinco lenguajes de programación: Java, C#, C++, VB.NET y Python. Los SRM se llevan a cabo de dos a cuatro veces al mes y duran dos horas cada uno. El día y hora de los SRM varía concurso a concurso [15].

Los SRM se suelen realizar en dos divisiones, cada una con su propio conjunto de problemas. Para fines prácticos, cada división se puede considerar como una competencia separada.

Los problemas son escritos por miembros de alto rango de TopCoder. TopCoder es conocida por tener problemas de alta calidad que son interesantes y difíciles de resolver. TopCoder ha estado realizando SRM desde 2001, y se puede encontrar un archivo que contiene un más de un millón de problemas. Después de cada SRM, se crea un foro para que los miembros puedan discutir el conjunto de problemas

CAPÍTULO 2. MARCO TEÓRICO

y un editorial que está escrito y publicado con las soluciones a los problemas. Los problemas y las soluciones de la competencia del concurso también se añaden al archivo de problemas, donde los miembros pueden ver los problemas y ver cómo otros los resuelven (o intentaron resolver).

Cada SRM se compone de las siguientes 4 fases:

- **Registration:** Los competidores deben registrarse antes de cada competencia, esto comienza tres horas antes de la competencia real, y termina cinco minutos antes de la misma.
- **Coding phase:** Es la fase principal de la competencia en donde todos los competidores hacen frente a tres problemas con distintos niveles de complejidad con una puntuación para cada uno (que varían según su dificultad), los puntos son otorgados para cualquier solución enviada que compile perfectamente y se basa en el tiempo transcurrido desde que se abre el problema hasta que se entrega el último envío de la solución. Esta fase tiene una duración de setenta y cinco minutos.
- **Challenge Phase:** Esta fase es en donde cada competidor tiene la oportunidad de desafiar la funcionalidad del código de otros competidores proporcionando una entrada que haga que el código del competidor produzca una salida errónea, genere algún error, exceda el límite de tiempo, ó falle de alguna otra manera. Se realiza con el objetivo de reducir a cero los puntos obtenidos por el acusado en el problema y un aumento de cincuenta puntos para el retador, sin embargo si el desafío falla, el retador pierde veinticinco puntos. Esta fase tiene una duración de quince minutos.
- **System Testing Phase:** Es la fase final de la competencia en donde se prueban todos los códigos presentados que no hayan sido impugnados en la Challenge Phase, si el código enviado en algún problema es incorrecto entonces la pun-

tuación obtenida en el problema se reduce a cero. Para probar el código este se ejecuta sobre un conjunto de datos de entrada, y se considerará correcto si la salida generada por la ejecución del programa es igual a la salida esperada.

2.4.4. Codeforces

Según la sección de preguntas frecuentes de Codeforces se establece que Codeforces es un proyecto para unir a las personas interesadas en los concursos de programación y su participación. Por un lado, Codeforces es una red social dedicada a la programación y las competencias. Por otro lado, es una plataforma donde los concursos se celebran con regularidad, las habilidades de los participantes se reflejan en su calificación y los antiguos concursos se pueden utilizar con el fin de prepararse. Codeforces esta en constante desarrollo y tiene la intención de mejorar la plataforma para dar a los participantes la oportunidad de organizar sus propios concursos, llenando el proyecto con contenido de aprendizaje, desarrollando así Codeforces como una plataforma de formación y aprendizaje [16].

Los concursos de Codeforces están basados en sus propias reglas, los concursos constan generalmente de cinco problemas a resolver en dos horas de competencia y están ordenados por dificultad (según los autores del concurso), además estos son visibles desde el inicio del concurso, cada problema tiene una puntuación base que puede disminuir según el tiempo que se tardó en resolver el problema y en los envíos erróneos hechos por el competidor. Además durante la competencia es posible bloquear un problema (si ya está resuelto) para así intentar “Hackear” las soluciones de los demás competidores. “Hackear” un problema consiste en introducir algún caso de prueba que no pueda ser resuelto por el programa enviado por el competidor al que se le hace el “Hackeo”. Es importante mencionar que los envíos realizados durante toda la competencia solo se evalúan con casos preliminares y al finalizar se

CAPÍTULO 2. MARCO TEÓRICO

evalúan con el resto de los casos para así establecer las puntuaciones y determinar los ganadores del concurso [17].

2.4.5. Google Code Jam

Google Code Jam es una competencia organizada por Google diseñada para ser una de las competencias más desafiantes a nivel mundial sobre programación. El concurso consiste en varias rondas por medio de internet donde se presentan desafíos de naturaleza algorítmica cada vez más difíciles, para así culminar el evento con la última ronda que se organiza de manera presencial en las oficinas de Google [18].

Cada concurso Code Jam consiste en una o más rondas como se puede describir con más detalle en las normas publicadas en el sitio web del Concurso [19]. Cada ronda consta de uno o más problemas. En cada ronda, el competidor recibirá una puntuación basada en las respuestas que proporcione a los problemas. Si el concurso tiene más de una ronda y la puntuación del competidor supera el umbral especificado o es uno de un número determinado de las puntuaciones más altas en esa ronda, el competidor avanza a la siguiente ronda.

En cada ronda a los competidores se le pedirá resolver uno o varios problemas algorítmicos, que pueden incluir una o más entradas pequeñas (casos de pruebas sencillos), entradas grandes (casos de pruebas más complicados) u otras entradas descritas en el enunciado del problema. Además un problema puede describir reglas especiales para las entradas que no sean pequeñas o grandes. El competidor es capaz de acceder a los problemas en la web del Concurso y descargar los archivos de entrada pertinentes una vez que comienza la ronda.

2.4.6. Facebook Hacker cup

La Facebook Hacker Cup es una competición internacional proporcionada por Facebook, Inc. donde los competidores (llamados "Hackers") compiten entre sí por la gloria y una oportunidad por obtener la "Hacker Cup" otorgada por Facebook[20].

Al igual que el Code Jam de Google, este concurso se divide en varias rondas. En cada ronda, los competidores recibirán calificaciones basadas en las respuestas que aportan a una serie de problemas. Después de cada ronda, un determinado número de competidores con las puntuaciones más altas en esa ronda avanzarán a la siguiente ronda. Los concursantes deberán resolver con éxito al menos 1 problema para avanzar más allá de cualquier ronda de la competición. La Ronda de Clasificación y Rondas 1, 2 y 3 se lleva a cabo en línea y la ronda final de manera presencial en la oficina de Facebook en Londres, Inglaterra.

Durante cada ronda de la competición, se presentará con un conjunto de problemas algorítmicos que consiste y un conjunto de entrada para cada problema. Una vez que comienza una ronda, cada competidor será capaz de acceder a los problemas de esa Ronda y descargar los archivos de entrada pertinentes. Una Presentación válida para un problema consiste en tener el código fuente del competidor, que se cree, resuelve el problema y el archivo de salida que se genera cuando se ejecuta el código fuente de la competencia en el conjunto de entrada correspondiente. Las presentaciones deben hacerse dentro del plazo fijado y determinado para cada problema, para que el envío pueda ser considerado. Además, los archivos de salida se deben proporcionar en el formato especificado [21].

2.5. Jueces y veredictos

Un programa es la solución a un problema si este pasa todos los casos de prueba, instancias del problema creados por el autor del mismo (casos que son desconocidos por el competidor). En los concursos tanto presenciales como a distancia, disponen de jueces que se encargan de evaluar si el programa presentado por el competidor es correcto o no, dando un veredicto según sea el caso. Existen distintos veredictos que se pueden asignar a un programa luego de ser ejecutado por el juez, mas allá de que indique si la solución es correcta o no, hay veredictos que permiten obtener más información sobre el programa enviado, estos términos usados para juzgar un programa pueden variar dependiendo del juez, sin embargo estos suelen ser los más generales que se pueden observar en la mayoría de los jueces [22]:

- In Queue (QU): El juez está ocupado y no puede atender su presentación, será juzgado tan pronto como sea posible.
- Accepted (AC): El programa es correcto. Produjo la respuesta correcta en un tiempo establecido y dentro del límite de uso de la memoria.
- Presentation Error (PE): La salida del programa es correcta pero no se presentan en la forma correcta. Generalmente suelen ser problemas con espacios en blanco o saltos de línea.
- Wrong Answer (WA): El programa presentado no resuelve todos los casos de prueba, por tanto es incorrecto.
- Compile Error (CE): El compilador del juez no logró compilar el código enviado. Algunos jueces indican el lugar donde ocurre el error de compilación.
- Runtime Error (RE): El programa falla durante su ejecución, generalmente por fallo de segmentación, divisiones entre 0, excepciones de punto flotante, entre otras cosas.

CAPÍTULO 2. MARCO TEÓRICO

- Time Limit Exceeded (TLE): El programa excedió el límite de tiempo establecido para su ejecución. Este error no permite saber si el programa resuelve correctamente todos los casos.
- Memory Limit Exceeded (ML): El programa usa más memoria de la permitida por el juez.
- Output Limit Exceeded (OL): El programa intentó escribir más información de la permitida, esto ocurre generalmente cuando el programa ejecuta un ciclo infinito.
- Submission Error (SE): El envío del código ha fallado, quizás por un error al momento de enviar, corrupción de los datos o algún error interno del juez.
- Restricted Function (RF): El programa intenta ejecutar alguna instrucción que esta prohibida por el juez.

También Existen variedad de jueces usados en competencias presenciales y por internet, los mas conocidos son los siguientes:

2.5.1. BOCA

Boca es un sistema de administración para realizar competencias de programación desarrollado en Brasil en la Universidad Catolica de Sao Paulo y en la Universidad de Sao Paulo. Está escrito en Php y utiliza Postgresql como motor de bases de datos. Soporta competencias de tipo ACM-ICPC y se ha utilizado en un gran número de competencias Regionales de este evento [23].

2.5.2. Kattis

Tal como se describe en una reciente investigación [4], kattis fue desarrollado por KTH Royal Institute of Technology para corregir tareas de varios cursos de esta uni-

CAPÍTULO 2. MARCO TEÓRICO

versidad. Luego sería usado en la final mundial de las competencias ACM-ICPC. Kattis está escrito en una combinación de Python, PHP y SQL corriendo sobre Solaris. Además de ser utilizado para corregir las tareas de variados cursos en la universidad KTH, actualmente es utilizado en la final mundial ACM-ICPC.

2.5.3. DOMjudge

DOMjudge se inició en 2004 en el estudio de la Asociación A-Eskwadraat en la Universidad de Utrecht, Países Bajos, sin embargo, en la actualidad es un proyecto independiente. Fue escrito en lenguaje PHP y utiliza como base de datos MySQL y está destinado para su uso en los concursos de programación como los ACM-ICPC y se ha utilizado en numerosos concursos regionales (principalmente de la región europea)[24].

2.5.4. UVA online judge

Es un juez online en donde se encuentran gran cantidad de problemas como los que se utilizan en las competencias de programación, disponibles en formato HTML y PDF, permitiendo resolverlos en una variedad de lenguajes.

En su página web[25] se permite la creación de un usuario, para el cual registra todas las estadísticas según los problemas que se resuelvan o se intenten resolver, mostrando así, por ejemplo, la cantidad de problemas aceptados, número de respuestas incorrectas, entre otras cosas.

2.5.5. SPOJ

La plataforma SPOJ es un sistema de juez online, que sirve para la evaluación automática de los programas presentados por el usuario. Según se expresa en la

CAPÍTULO 2. MARCO TEÓRICO

pagina web [26] sus características mas importantes son:

- SPOJ soporta más de 45 lenguajes de programación y compiladores, incluyendo C, C ++, Pascal, Java, C#, Perl, Python, Ruby, Haskell, ocaml y lenguajes esotéricas.
- Tiene a su disposición un creciente conjunto de problemas (cerca de trece mil) disponibles las 24 horas del día en distintos idiomas (ingles,polaco,vietnamita, portugués, entre otros), además de muchos problemas originales elaborados por la comunidad de “problemsetters”(aquel que crea un problema, su solución y los casos de prueba).
- Tiene un sistema de pruebas flexible, soportando incluso interacción dinámica con los problemas enviados, además de que la salida usada para la evaluación de resultados es altamente personalizable.
- Además esto posee una interfaz para la gestión de contenido basada en navegador que permite a los usuarios la creación de sus propios concursos en cuestión de minutos, que además pueden hacer uso de los problemas ya alojados en el sistema.

2.5.6. Moose

Moose es un juez realizado por Emilio Tirado y Ricardo Tovar como proyecto de tesis en la Universidad Central de Venezuela, que permite realizar distintos formatos de competencias de programación, además de ofrecer una visualización agradable y moderna, atractiva para este tipo de actividad [4].

2.6. Minería de datos

La minería de datos es una rama interdisciplinaria de la computación, es el proceso computacional de descubrir patrones en grandes conjuntos de datos, utilizando métodos que se pueden ver como el punto de intersección entre inteligencia artificial, machine learning, la estadística y los sistemas de bases de datos.

El objetivo de la minería de datos es extraer información de un conjunto y transformarla en estructuras de alguna forma entendibles para uso futuro, Los siguientes aspectos son aquellos que componen a la minería de datos:

- **Inteligencia artificial (IA):** Es un campo de la computación que estudia como crear computadores y software que sean capaces de tener un comportamiento inteligente, la meta propuesta de este campo es investigar sobre como generar razonamiento, conocimiento, aprendizaje, percepción y procesamiento del lenguaje natural en una máquina.
- **Machine learning:** Es un campo de la computación que ha evolucionado del estudio del reconocimiento de patrones y de la teoría del aprendizaje iniciados en la inteligencia artificial, es un campo de estudio que le da a las computadoras la habilidad de aprender sin ser explícitamente programadas.
Machine learning explora el estudio y la construcción de algoritmos que puedan aprender y hacer predicciones sobre datos, estos algoritmos operan creando un modelo mediante ejemplos de entrada para poder tomar decisiones o hacer predicciones en base a ese conjunto de prueba, en lugar de hacer programas estrictamente estáticos.
- **La Estadística:** Es el estudio de la recolección, análisis, interpretación, presentación y organización de los datos, con el objetivo de encontrar una determinada característica común para poder llegar a una conclusión a partir de los datos

CAPÍTULO 2. MARCO TEÓRICO

procesados, así mismo la manipulación de los mismos para realizar inferencias en el cálculo de probabilidades.

- sistemas de bases de datos (DB): una DB es una colección organizada de datos, manejados por medio de esquemas, tablas, consultas, reportes, vistas y otros objetos. Los datos son organizados para modelar aspectos de la realidad de una manera que cumplan los requisitos de información de los procesos, similar a modelar la disponibilidad de cuartos en un hotel.

La minería de datos posee además una serie de pasos que da a entender el proceso por el cual pasan los datos antes de convertirse en información útil:

- Recolección de datos: Es el proceso de obtener y medir información en forma de variables asignadas de una manera sistemática, lo cual permite responder preguntas relevantes y evaluar resultados.

En minería de datos, es el primer paso el cual es obtener los datos en un estado no estructurado y que no aporta mucho en su estado actual, este estado se le conoce como “crudo”.

- Extracción de información (IE): Es el proceso automatizado mediante el cual se extrae información estructurada de un conjunto de datos no estructurados ó semi estructurados los cuales pueden ser leídos por la máquina, comúnmente esta actividad concierne el procesamiento de palabras de un documento escrito utilizando un lenguaje natural.

En minería de datos, este es el segundo paso mediante el cual podremos extraer información, al convertir los datos no estructurados en información estructurada que el programa pueda entender.

- Data warehousing: Es un sistema usado para reportes y análisis de datos, son repositorios centrales de datos integrados de una o más fuentes.

En minería de datos, este es el tercer paso mediante el cual podemos almacenar

la información que va siendo recolectada de manera eficiente y coherente.

- **Análisis de los datos:** Es el proceso de inspeccionar, limpiar, transformar y modelar los datos para encontrar información útil, sugerir conclusiones y ayudar a la toma de decisiones.

En minería de datos, este es el cuarto paso mediante el cual podemos filtrar los datos almacenados y realizar pruebas con el algoritmo para llegar a algún resultado.

2.7. Algoritmos de aprendizaje

Son algoritmos relacionados directamente al machine learning, los cuales tienen el objetivo de crear un sistema que pueda decidir un camino ó predecir un resultado dado un conjunto de datos, sin necesidad de programar dichas decisiones de manera explícita.

A lo largo de los años se han desarrollado diversos algoritmos para el aprendizaje, ya que es un concepto relativamente nuevo muchos de estos algoritmos son mejorables y siguen en estudio constante por parte de los científicos.

Algunos tipos de aprendizaje son los siguientes:

2.7.1. Aprendizaje supervisado

Es una técnica de aprendizaje que elabora una función matemática (hipótesis) a partir de datos de entrenamiento previamente etiquetados, entonces se colocan dos tipos de entradas, entradas de entrenamiento, que son entradas cuyas respuestas ya son conocidas por el sistema y lo ayudarán a poder predecir el resultado correcto de el otro tipo entrada que son las entradas de prueba.

2.7.2. Aprendizaje no supervisado

Es aquella técnica en la cual no se dispone de un conjunto de entrenamiento si no que se intentan establecer los resultados utilizando técnicas de clustering o similares en tiempo de ejecución.

2.7.3. Aprendizaje semi-supervisado

En los datos de entrenamiento existen elementos ya etiquetados pero a su vez elementos cuyo resultado se desconoce, esto permite una mayor versatilidad mezclando los aprendizajes supervisados y no supervisados.

2.7.4. Aprendizaje por refuerzo

Es un tipo de aprendizaje donde un agente toma acciones en un ambiente para maximizar la noción de una recompensa a largo plazo. El aprendizaje por refuerzo busca relacionar estados con acciones que el agente puede realizar para cumplir su meta final, este tipo de aprendizaje difiere del aprendizaje supervisado ya que la entrada y salida no está presente, y las acciones sub-óptimas que llevaron al estado final no son explícitamente las correctas.

2.8. Técnicas de Minería de datos

Las técnicas de la minería de datos provienen de la inteligencia artificial y de la estadística, dichas técnicas, no son más que algoritmos, más o menos sofisticados que se aplican a un conjunto de datos para obtener unos resultados. Algunas de estas Técnicas son las siguientes [27]:

2.8.1. Clustering

El Clustering, es un procedimiento de agrupación de un conjunto de datos de acuerdo con un criterio. Esos criterios son por lo general son distancia o similitud. La cercanía se define en términos de una determinada función de distancia, como la euclídea, aunque existen otras más robustas o que permiten extenderla a variables discretas. La medida más utilizada para medir la similitud entre los casos es la matriz de correlación. Sin embargo, también existen muchos algoritmos que se basan en maximizar una propiedad estadística llamada verosimilitud [28].

Estas técnicas y algoritmos de clustering se aplican cuando no existen clases específicas para predecir a cual pertenecen los datos pero sus instancias deben ser divididos en grupos naturales de acuerdo a algún criterio específico. Estos grupos se forman con instancias que tienen una semejanza fuerte entre si y que con otras instancias no tiene, para así crear una diferencia entre el conjunto de datos [27].

Hay diferentes maneras en que el resultado de un agrupamiento se puede expresar. Los grupos que se identifican pueden ser [27]:

- Exclusivos: Cualquier instancia pertenece a un solo grupo específico.
- Superpuestas: Una instancia puede caer en varios grupos distintos.
- Probabilística: Una instancia pertenece a cada grupo con una cierta probabilidad.
- Jerárquica: Se realiza una división aproximada de casos en grupos de nivel superior y cada grupo es refinado aun mas. Una división aproximada de casos en grupos en el nivel superior y cada grupo refinado aún más, así hasta llegar a grupos mas individuales y específicos.

El uso de una u otra manera de agrupar el conjunto de datos dependerá del problema y como se quiera determinar el agrupamiento de los datos, sin embargo muchas veces la elección del tipo de representación es dictado por las herramientas de

CAPÍTULO 2. MARCO TEÓRICO

agrupación que están disponibles.

Esta técnica se lo considera una técnica de aprendizaje no supervisado puesto que busca encontrar relaciones entre los datos que a priori no se sabe con exactitud sus similitudes y no se tienen clases predefinidas para comparar con dichos datos.

2.8.2. Clasificación

La clasificación es una de las tareas que se llevan a cabo en minería de datos. Consiste en determinar la clase a la que pertenece un nuevo item (objeto o datos que se quiere clasificar), en una estructura conocida donde ya están definidas dichas clases.

2.8.3. Reglas de asociación

Se trata de encontrar nuevas relaciones en conjuntos de datos muy grandes, esto se hace para poder hacer búsquedas más rápidas en el futuro. Las reglas de asociación relacionan una determinada conclusión con un conjunto de condiciones.

2.8.4. Regresión

El análisis de la regresión es un proceso estadístico para estimar las relaciones entre variables. Incluye muchas técnicas para el modelado y análisis de diversas variables, cuando la atención se centra en la relación entre una variable dependiente y una o más variables independientes (o predictoras). Más comúnmente, el análisis de regresión estima la esperanza condicional de la variable dependiente dadas las variables independientes, es decir, el valor promedio de la variable dependiente cuando se fijan las variables independientes. El objetivo de la regresión es la estimación de una función de las variables independientes llamada la función de regresión.

CAPÍTULO 2. MARCO TEÓRICO

En el análisis de regresión, también es de interés para caracterizar la variación de la variable dependiente en torno a la función de regresión que puede ser descrito por una distribución de probabilidad.

2.8.5. Algoritmo Apriori

El algoritmo Apriori es un algoritmo diseñado para agrupar elementos comunes en conjuntos, es un algoritmo de fuerza bruta que se centra en encontrar conjuntos combinándolos con cierto nivel de similitud entre varios elementos y agruparlos, los cuales sirven de base para generar reglas de asociación. Se basa en ir incluyendo de manera progresiva elementos en conjuntos de tamaño k para crear un nuevo conjunto de tamaño $k+1$. Este algoritmo posee dos parámetros importantes que son el soporte, el cual es un índice para la generación de combinaciones del algoritmo y la confianza, que es el índice para la generación de reglas. Ambos índices funcionan para reducir el número de combinaciones y de reglas que se generan, debido a la alta complejidad del algoritmo [29].

2.9. Sistemas de recomendación

Un sistema de recomendación es un software o sistema que posee herramientas y técnicas para proporcionar sugerencias personalizadas sobre determinado tipo de elementos (también llamados Items o artículos) que son de utilidad para el usuario que usa el sistema [30]. Estas recomendaciones se hacen a partir de las preferencias y opiniones del usuario, para esto se hace un estudio sobre las características del usuario y mediante un procesamiento de datos, se encuentra un subconjunto de items que puede ser de utilidad e interés para el usuario [31].

Los sistemas de recomendación principalmente van dirigidos a personas que no

CAPÍTULO 2. MARCO TEÓRICO

tienen suficiente experiencia personal o competencia para evaluar una gran cantidad de ítems de interés en un sitio web, para así ayudar a los usuarios a encontrar el objeto de interés lo más rápido posible.

La implementación de técnicas para el desarrollo de los SR está directamente relacionada con el tipo de información que se vaya a utilizar. Una primera fuente de información a tener en cuenta es el tipo de ítems con los que se va a trabajar. Habrá situaciones en las que únicamente se conozca un identificador de cada ítem. En otras situaciones, se dispondrá de más información sobre los ítems, a través de una serie de atributos. En general, cuanto más sofisticada es la representación de los ítems mejor se puede desarrollar la actividad de los sistemas de recomendación [32].

Independientemente de la representación de los ítems hay diversos aspectos a tener en cuenta al realizar un sistema de recomendación [32, 33]:

- Representación de las recomendaciones: Los contenidos de una evaluación pueden venir dados por un único bit (recomendado o no) o por comentarios de texto sin estructurar.
- Expresión de las recomendaciones: Las recomendaciones pueden ser introducidas de forma explícita o bien de forma implícita.
- Aspectos de identificación de la fuente: Las recomendaciones pueden realizarse de forma anónima, identificando la fuente, o bien usando un seudónimo.
- Forma de agregar las evaluaciones: Se refiere a cómo se va a ir agregando las evaluaciones disponibles sobre los ítems de cara a generar las recomendaciones.
- Uso de las recomendaciones: Las recomendaciones se pueden usar de distintas formas. Por ejemplo, se podrían mostrar los ítems en forma de lista ordenada según las recomendaciones de cada usuario, o a la hora de visualizar los ítems que se muestre también su recomendación.

A continuación se describirá los elementos fundamentales que forman un sistema de recomendación que además sirven como criterios para clasificar los sistemas de recomendación [32]

2.9.1. Entradas y Salidas

Los sistemas de recomendación para generar las recomendaciones, utilizan información no solo del usuario activo (usuario al que se le quiere hacer una recomendación) sino también de los items e incluso de otros usuarios que son llamados colaboradores. Para poder realizar una recomendación es necesario conocer de alguna manera las preferencias de el usuario y dependiendo del tipo de sistema se requiere información y características de los items, además de la información de otros usuarios para buscar similitudes con el usuario activo[32].

La información se puede obtener del usuario puede venir de dos formas [34] por extensión e intencionalmente.

- Por extensión: Se refiere a las acciones pasadas realizadas por el usuario sobre algunos items encontrados. También se refiere a la navegación implícita ya que el usuario no es consciente de este seguimiento.
- Intencionalmente: El usuario provee información que se le solicita para recibir información de los items de su preferencia. También se conoce como navegación explícita.

Al generarse una recomendación el sistema genera una salida que variarán dependiendo del tipo, cantidad y formato de la información proporcionada al usuario. Algunas de las formas más comunes de representar la salida son las siguientes [32]:

- Sugerencia o lista de sugerencias al usuario de una serie de ítems.
- Presentar al usuario predicciones del grado de satisfacción que se asignará al

ítem concreto. Estas estimaciones pueden ser presentadas personalizadas en base a las preferencias del usuario o basado en estimaciones generales de un conjunto de usuarios colaboradores.

- Cuando la comunidad de usuarios es pequeña o se conocen bien los miembros de dicha comunidad, podría ser útil visualizar las valoraciones individuales de los miembros que permitiría al usuario obtener sus propias conclusiones sobre la efectividad de una recomendación.

Suele ser recomendable incluir alguna breve descripción o explicación sobre los ítems recomendados que sirva como justificación de la recomendación.

2.9.2. Métodos de Generación de Recomendaciones

Existen distintos métodos para generar recomendaciones, el uso de cada uno dependerá del objetivo del sistema, además determina que algoritmo utilizar para obtener las recomendación. Sin embargo hay que tener en cuenta que estos no son necesariamente excluyentes entre si, sino que podrían ser incluso complementarios, pudiendo usar uno o varios métodos en un mismo sistema de recomendación. Algunos de los mas conocidos métodos son los siguientes [30]:

- Basados en Contenido: Las recomendaciones se basan en el conocimiento que se tiene sobre los ítems que el usuario ha valorado (ya sea de forma implícita o explícita), y se le recomendarán ítems similares que le puedan gustar o interesar. El perfil de usuario y la información que se ha obtenido de el, es la clave para el uso de este método. Los ítems se suelen denotar por palabras claves para facilitar su búsqueda.
- Filtro Colaborativo: El filtrado colaborativo consiste en ver que usuarios son similares al usuario activo (o usuario al que hay que realizarle las recomendaciones) y a continuación, recomendar aquellos ítems que no han sido votados

por el usuario activo y que han resultado bien valorados por los usuarios similares. Este es uno de los métodos mas populares y usados en sistemas de recomendación ya que se suelen obtener mejores resultados debido al uso de los usuarios colaboradores.

2.9.3. Grado de Personalización

Además de las clasificaciones anteriores, los sistemas de recomendación también se pueden clasificar según el grado de personalización en las recomendaciones que proporcionan al usuario[32].

- Cuando los sistemas de recomendación proporcionan las mismas recomendaciones a todos los usuarios, son clasificados en este ámbito como no personalizados. Dichas recomendaciones estarán basadas en selecciones manuales, resúmenes estadísticos u otras técnicas similares.
- Los sistemas de recomendación que tienen en cuenta la información actual del usuario objeto de las recomendaciones, proporcionan personalización efímera, puesto que las recomendaciones son repuesta al comportamiento y acciones del usuario en su sesión actual de navegación.
- Los Sistemas de recomendación que ofrecen el mayor grado de personalización son los que usan personalización persistente ofreciendo recomendaciones distintas para distintos usuarios, incluso cuando estén buscando el mismo ítem. Estos sistemas están basados en el perfil de los usuarios, por lo que hacen uso de métodos de filtrado colaborativo, filtrado basado en contenidos o correlaciones entre ítems.

2.9.4. Modelo de los sistemas de recomendación

2.9.4.1. Matriz de Utilidad

Según [35], en un sistema de recomendación se pueden encontrar dos clases de entidades, estas se refieren a los usuarios y los artículos (o items). Los usuarios tienen preferencias por ciertos artículos, y estas preferencias deben ser obtenidas de los datos. Estos datos se representan con una Matriz de utilidad, donde para cada par de elemento Usuario-Artículo se tiene un valor que representa el grado de preferencia que el usuario tiene por ese elemento o artículo. Estos valores provienen de un conjunto ordenado que indica la calificación del usuario a un elemento específico. Por lo general se asume que esta matriz tiene muy pocos datos, por lo que la mayoría de las entradas son desconocidas y no se tiene información explícita sobre muchas de las preferencias del usuario. El objetivo de un sistema de recomendación consiste en predecir estas calificaciones que el usuario daría a un artículo, para proceder a recomendarlo en caso de ser adecuado.

Se tiene que tener en cuenta que un sistema de recomendación no necesariamente tiene que predecir todas las entradas de la matriz, sino solamente aquellas que puedan resultar más atractivas para el usuario al que se va a recomendar.

2.9.4.2. Aplicaciones de los sistemas de recomendación

Actualmente los sistemas de recomendación están presentes en multitud de aplicaciones, sin embargo las más comunes según [35] son:

- **Recomendación de Productos:** es una de las aplicaciones más importantes de los sistemas de recomendación usado por vendedores vía internet. Muchas empresas de este tipo, se esfuerzan por presentar a cada usuario, sugerencias de productos que le gustaría comprar. Estas sugerencias no son al azar, sino que

CAPÍTULO 2. MARCO TEÓRICO

están basadas en las decisiones de compra realizadas por clientes similares, o por las compras realizadas con anterioridad del mismo usuario.

- **Recomendación de Películas:** Estos sistemas se encargan de recomendar películas o cualquier otro tipo de artículos de interés para el usuario. Este tipo de recomendación se basa en las calificaciones que proporcionan los usuarios.
- **Recomendación de noticias:** Estos sistemas se encargan de recomendar noticias a los usuarios basados en las noticias que han leído con anterioridad. Se encargan de buscar similitudes entre dichas noticias basados en palabras importantes de los documentos o en noticias leídas por personas con gustos similares. Estos mismos principios se aplican no solo a noticias, sino cualquier otro tipo de artículos que brinden información al usuario.

2.9.4.3. Manejo de la matriz de utilidad

Sin el uso de una matriz de utilidad, es bastante difícil recomendar artículos. Sin embargo, la adquisición de datos para construir dicha matriz suele ser difícil. Por tanto existen dos métodos generales para determinar los valores que pondrían los usuarios a un artículo específico [35]:

- **Preguntar al usuario su calificación a los artículos.** El usuario da una calificación a un artículo específico y este es usado para generar recomendaciones. Este enfoque es bastante limitado puesto que generalmente hay usuarios que no están dispuestos a proporcionar este tipo de información.
- **Hacer inferencia de la calificación de los artículos.** Este enfoque supone que si un usuario tiene cierta interacción con un artículo, se puede decir que está interesado en dicho artículo.

2.9.5. Filtrado basado en contenido

Tal y como se menciona en la sección 2.9.2, los sistemas de recomendación basados en contenido se centran en propiedades y características de los usuarios. Para comparar los artículos se realiza una medición de la similitud de sus propiedades. A continuación se darán detalles sobre este tipo de sistemas de recomendación [35].

2.9.5.1. Perfil de artículos

En un sistema basado en contenido, se debe construir para cada elemento un perfil, que es un registro o una colección de registros que representan características importantes de ese artículo. En casos sencillos, el perfil se compone de algunas características del artículo que se pueden descubrir con facilidad, sin embargo también existen cierta clase de elementos o artículo que nos permiten obtener características en disposición de los datos, incluso sin la necesidad de introducir datos manualmente.

2.9.5.2. Obtención de características de los artículos a través de etiquetas

Las etiquetas son marcas que se le dan a los artículos para identificarlos, clasificarlos e incluso valorarlos. En los sistemas de recomendación, se le invita a algún tipo de usuario a etiquetar los artículos mediante la introducción de palabras o frases que puedan describir el artículo y estas son usadas para para recomendar a los usuarios, puesto que si se observa que un usuario visita artículos con una misma etiqueta, se puede suponer que este tiene cierto interés por artículos con una misma etiqueta y se proceder a realizar la recomendación. El problema con este enfoque es que este proceso solo funciona si los usuarios están dispuestos a crear dichas etiquetas, y en algunos casos se pueden introducir etiquetas erróneas que pueden provocar que el sistema se deteriore.

CAPÍTULO 2. MARCO TEÓRICO

2.9.5.3. Representación de perfil de artículos

Para que el sistema pueda realizar recomendaciones, es necesario comparar los distintos perfiles de los artículos y para esto se debe tener una forma de representarlos. Un perfil de los artículos consiste en pares característica-valor y el perfil de usuario resume todas las preferencias del mismo correspondiendo a una fila de la matriz de utilidad. Una manera de representar los perfiles de artículos es usando un vector basado en sus características, es decir cada elemento del vector corresponde a el valor de alguna característica del artículo, esta representación permite establecer mediciones de similitud entre los artículos usando algún tipo de función para medirla. Es importante destacar que no todas las características describen potencialmente a un artículo, algunas lo hacen en menor medida, por tanto muchas veces es necesario manipular de alguna manera los elementos del vector para así poder hacer una mejor comparación entre los artículos.

2.9.5.4. Perfil de Usuarios

No solo se debe representar los artículos, también se debe representar a los usuarios mediante vectores. Para esto generalmente se crea un vector basado en las calificaciones obtenidas de la matriz de utilidad en el cual se almacenan el grado de preferencia que tiene el usuario por las características de los artículos (que se tienen en cuenta en el perfil de artículos).

Generalmente también se suele generar una calificación promedio que el usuario da a los artículos y así normalizar al momento de encontrar las similitudes, restando el valor promedio, esto con la finalidad de obtener valores negativos para artículos que estén por debajo de la media y positivos para aquellos que estén por encima

Una vez obtenidos los vectores tanto del perfil de usuario, como de los artículos,

CAPÍTULO 2. MARCO TEÓRICO

se puede estimar el grado o calificación que un usuario daría a cierto artículo. En base a estas estimaciones se procede a recomendar los artículos al usuario.

2.9.5.5. Algoritmos de clasificación

Un enfoque completamente diferente a un sistema de recomendación basado en el perfil de artículos y la matriz de utilidad, es tratar al problema como uno de aprendizaje automático (o machine learning). Para este enfoque hay que considerar los datos dados como un conjunto de entrenamiento, y para cada usuario se construye un clasificador que predice la calificación del usuario para todos los artículos restantes. El clasificador se encarga básicamente de tomar la decisión de si un artículo específico le gusta o no al usuario.

Sin embargo usar algún algoritmo de clasificación para realizar recomendaciones puede llegar a ser un problema, puesto que, la mayoría de los clasificadores suelen ser bastante complejos y toman mucho tiempo su construcción, por tanto usar clasificadores para recomendaciones se suele usar cuando el tamaño del problema es pequeño.

2.9.6. Filtrado Colaborativo

En un sistema de recomendación con filtrado colaborativo, en vez de utilizar las características de los artículos para determinar su similitud, este sistema se basa en encontrar usuarios que sean similares y en base a eso hacer recomendaciones asumiendo que si dichos usuarios hicieron valoraciones parecidas, entonces sus gustos son similares.

Por tanto este enfoque de sistema de recomendación, se encarga de realizar los siguientes pasos:

- Establecer una métrica para la similitud entre usuario. El sistema requiere en-

CAPÍTULO 2. MARCO TEÓRICO

contrar, dado un usuario, aquellos que son similares a este, para esto se tiene que requerir de alguna métrica de similitud entre los usuarios.

- Calcular los usuarios con mayor similitud. Haciendo uso de la métrica de similitud seleccionada, se obtienen los usuarios que tienen mayor similitud con el usuario al que se le realizara la recomendación.
- Calcular las predicciones de los artículos. Una vez que se encuentren los usuarios que tienen mayor similitud, se procede a estimar las valoraciones que otorgaría el usuario a los artículos, basado en los usuarios similares obtenidos
- Realizar las recomendaciones. Tras el calculo de las predicciones, se eligen los los artículos más adecuados para ser recomendados al usuario, es decir, las predicciones más altas. De forma opcional, puede incluirse un umbral para evitar que los artículos con una predicción inferior a dicho umbral sean recomendados.

2.9.6.1. Algoritmo de los K-Vecinos

El algoritmo de los K-Vecinos mas cercanos es un método no paramétrico usado para clasificación y regresión (proceso estadístico para estimar relaciones entre variables) y su salida depende de su uso:

- En clasificación: El algoritmo recibe un objeto y como salida proporciona la clase a la que pertenece dicho objeto. Un objeto se clasifica por un voto de la mayoría de sus vecinos, es decir al objeto se le asigna la clase más común entre sus k vecinos más cercanos (k es un número entero positivo, por lo general pequeño). Si k es igual a uno, entonces el objeto simplemente se le asigna la misma clase que su vecino mas cercano.
- En regresión: la salida es el valor de una propiedad del objeto. Este valor es el promedio de los valores de sus k vecinos más cercanos.

CAPÍTULO 2. MARCO TEÓRICO

El algoritmo de los K-Vecinos más cercanos es un tipo de aprendizaje perezoso, donde la función sólo es aproximada a nivel local y todos los cálculos se aplazan hasta la clasificación. El algoritmo de los K-Vecinos está entre los más simples de todos como algoritmo de aprendizaje automático.

Los vecinos se toman de un conjunto de objetos conocidos. Esto puede ser pensado como el conjunto de entrenamiento para el algoritmo, aunque no es necesario realizar esta etapa de forma explícita.

El algoritmo de los K-Vecinos suele ser usado en los sistemas de recomendación como método de regresión para estimar el valor de las calificaciones que un usuario daría a ciertos artículos mediante la media de las calificaciones sus K-Vecinos más cercanos.

2.9.7. Métricas para recomendaciones

La medición es un proceso mediante el cual obtenemos la similitud entre dos conjuntos, a continuación se presentan distintos métodos para obtener la distancia y similitud entre conjuntos [36]

- Similaridad del coseno: Un índice con un resultado entre cero y uno, que funciona para medir qué tan similares son dos vectores numéricos dado un grado de similitud máxima y una mínima se pueden encontrar usuarios similares sin que sean demasiado similares ó demasiado diferentes.

$$1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2}. \quad (2.1)$$

- Distancia euclidiana: Podemos obtener la distancia entre dos puntos n-dimensionales, es decir se toma los datos como si fuese un punto en un espacio n-dimensional y se verifica cual es la distancia entre dichos puntos, dado una distancia máxi-

CAPÍTULO 2. MARCO TEÓRICO

ma para decir que son similares y una mínima para decir que no son demasiado similares podemos utilizar este método para determinar la similitud.

$$\|u - v\|_2 \quad (2.2)$$

- **Desviación estándar y promedio:** La desviación estándar dice qué tan separados están los datos, para llevar estos datos a una métrica de distancia congruente, es decir que por ejemplo el conjunto [1,1,1] tenga desviación "1" el conjunto [9,9,9] tenga desviación "1" dividimos entre el promedio para obtener una función de distancia.[37]

$$\text{avg} = \frac{A + B + C}{3} \quad (2.3)$$

$$\sqrt{\frac{\sum_{i=1}^n (x_i - \text{avg})^2}{n - 1}} \quad (2.4)$$

- **Distancia de Hamming:** Se denomina distancia de Hamming a la efectividad de los códigos de bloque y depende de la diferencia entre una palabra de código válida y otra. Cuanto mayor sea esta diferencia, menor es la posibilidad de que un código válido se transforme en otro código válido por una serie de errores. El concepto anteriormente descrito puede llevarse a una forma de verificar la similitud entre dos usuarios si se representa que un problema fue resuelto con "1" si no fue resuelto con "0", colocando una distancia mínima y máxima para ser considerados usuarios similares pero no demasiado.
- **Disimilaridad de Bray–Curtis:** Es un método estadístico muy usado en la biología para encontrar la disimilaridad entre dos conjuntos, esta distancia se refiere a qué tan no similares son dos conjuntos, este índice va de cero a uno, cero siendo que los dos conjuntos son idénticos y uno siendo que los dos conjuntos son

completamente distintos.

$$\sum |u_i - v_i| / \sum |u_i + v_i| \quad (2.5)$$

2.10. Aplicaciones Web

Como se expresa en [38] una aplicación web es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador o visualizador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (HyperText Transfer Protocol (HTTP)) están estandarizados y no han de ser creados por el programador de aplicaciones.

El protocolo HTTP forma parte de la familia de protocolos de comunicaciones Transmission Control Protocol/Internet Protocol (TCP/IP), que son los empleados en Internet. Estos protocolos permiten la conexión de sistemas heterogéneos, lo que facilita el intercambio de información entre distintos ordenadores.

2.10.1. Arquitectura de las Aplicaciones Web

Una aplicación Web usa una arquitectura Cliente/servidor y esta se refiere a una arquitectura de red en la que cada computador o proceso en la red es cliente o servidor [38]. Normalmente, los servidores son computadores potentes dedicados a gestionar unidades de disco (servidor de ficheros), impresoras (servidor de impresoras), tráfico de red (servidor de red), datos (servidor de bases de datos) o incluso aplicaciones (servidor de aplicaciones), mientras que los clientes son máquinas menos potentes y usan los recursos que ofrecen los servidores. Esta arquitectura implica la existencia de una relación entre procesos que solicitan servicios (clientes) y procesos que responden a estos servicios (servidores). Estos dos tipos de procesos pueden

CAPÍTULO 2. MARCO TEÓRICO

ejecutarse en el mismo procesador o en distintos.

La arquitectura cliente/servidor nos permite la separación de funciones en tres niveles o capas[38]:

- **Lógica de presentación.** La presentación de los datos es una función independiente del resto.
- **Lógica de negocio (o aplicación).** Define los procesos que involucran a la aplicación.
- **Lógica de datos.** La gestión de los datos debe ser independiente para poder ser distribuida según las necesidades de la aplicación en cada momento.

Inicialmente las arquitecturas cliente/servidor, tenían solo 2 niveles. Una arquitectura de dos niveles está basada en un sistema gestor de bases de datos donde el cliente mantiene la lógica de la presentación, negocio, y de acceso a los datos, y el servidor únicamente gestiona los datos. Suelen ser aplicaciones cerradas que supeditan la lógica de los procesos cliente al gestor de base de datos que se está usando.

En las arquitecturas de tres niveles, la lógica de presentación, la lógica de negocio y la lógica de datos están separadas, de tal forma que mientras la lógica de presentación se ejecutará normalmente en la estación cliente, la lógica de negocio y la de datos pueden estar repartidas entre distintos procesadores.

El objetivo de aumentar el número de niveles en una aplicación es lograr una mayor independencia entre un nivel y otro, lo que facilita la portabilidad en entornos heterogéneos.

2.10.1.1. Modelo-Vista-Controlador

Según [39], el Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

CAPÍTULO 2. MARCO TEÓRICO

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

De manera genérica, los componentes de MVC se pueden definir como sigue:

- **El Modelo:** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'. Este nivel también se encarga del manejo, acceso y control de los datos, generalmente usando algún sistema de gestión de bases de datos.
- **El Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.
- **La Vista:** Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requie-

re de dicho 'modelo' la información que debe representar como salida.

Aunque originalmente el patrón MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón; estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

2.11. Capas de la Arquitectura

2.11.1. Servidor de bases de datos

Un sistema manejador de bases de datos, comúnmente usados en arquitecturas cliente/servidor suele ser MySQL, en [40] podemos encontrar lo siguiente:

Es un sistema manejador de bases de datos relacionales, es uno de los manejadores más utilizados en la actualidad, y es el más usado en términos de modelo cliente-servidor, MySQL significa My Structured Query Language (Mi lenguaje de consultas estructurado).

Es muy utilizado para crear bases de datos que serán usadas en aplicaciones web, y es un componente central muy usado en aplicaciones de todo el mundo.

Puede ser usado desde una consola, o mediante interfaces gráficas, así mismo como puede ser utilizado utilizando otras herramientas y lenguajes de por medio, como por ejemplo Python.

Permite el almacenamiento de volúmenes de datos de tamaño a la escala de por ejemplo almacenar todos los problemas de programación que existen en la web.

Es redundante, por tanto se necesita tener extremo cuidado a la hora de agregar nuevos elementos, ya que maneja una clave implícita cuando se agrega un nuevo

elemento cuyas claves son iguales a un elemento ya existente.

2.11.2. Servidor Web

Un servidor web o servidor HTTP, según [41], es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI.

Existen variedad de servidores web, uno de los mas conocidos es Apache. Tal y como se describe en [42], [43] y [4] Apache un servidor web HTTP de código abierto, para plataformas Unix, Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Formo parte fundamental del crecimiento de la World Wide Web. A pesar de no ser un servidor web diseñado para ser el mas rápido, tiene un rendimiento similar a otros servidores de alto rendimiento. Apache provee una variedad de módulos de multi-procesamiento que le permite a Apache correr en modo basado en procesos, modo híbrido o basado en eventos-híbrido, lo que permite satisfacer las demandas particulares de cada infraestructura.

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python y

recientemente Ruby.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

2.11.3. Lenguajes de Programación

2.11.3.1. Python

Tal y como se describe en [44], Python es un lenguaje de alto nivel, de propósito general e interpretado, su filosofía enfatiza en la legibilidad del código y su sintaxis permite a los programadores expresar conceptos en pocas líneas de código comparadas a lenguajes como C++ y Java. El lenguaje posee herramientas que mantienen los programas a gran y pequeña escala, limpios, es decir que no toma más recursos de los que necesita.

Python soporta múltiples paradigmas, incluyendo la programación orientada a objetos, la programación imperativa y la programación funcional. Permite un sistema de manejo de memoria dinámico y posee una enorme y comprensible librería estándar.

Los interpretes de Python están disponibles para ser instalados en múltiples sistemas operativos, permitiendo al código escrito en este lenguaje ejecutarse en una gran variedad de sistemas. Usando herramientas de terceros, el código escrito en Python puede ser empaquetado en un único ejecutable para los sistemas operativos más populares.

CAPÍTULO 2. MARCO TEÓRICO

2.11.3.2. JavaScript

Según [45], JavaScript es un lenguaje de programación de alto nivel, dinámico, débilmente tipado, e interpretado. Es una de las tres tecnologías principales en la creación de contenido de la red, la mayoría de los sitios web lo utilizan y es soportado por muchos navegadores sin necesidad de plug-ins.

JavaScript es un lenguaje además basado en prototipos, haciéndolo un lenguaje multi-paradigma permitiendo el uso de la programación orientada a objetos, la programación imperativa y la programación funcional. Posee un API para trabajar con textos, arreglos, fechas y expresiones regulares, pero no incluye ningún sistema de Entrada/Salida, para esta parte JS depende del ambiente en el que está trabajando el host.

La sintaxis de JavaScript poco o nada tiene que ver con Java, ya que esta está derivada casi en su totalidad del lenguaje C, mientras que las semánticas y el diseño han sido influenciadas por los lenguajes Self y Scheme. JavaScript es también usado en ambientes que no están basados en Web, como documentos PDF y desktop widgets.

Es utilizado también en la creación de juegos, motores gráficos, aplicaciones web, aplicaciones de escritorio, y programación del lado del cliente y del lado del servidor.

2.11.4. Lenguajes de Mercado

2.11.4.1. HTML

Según [46] HyperText Markup Language (HTML) es el lenguaje de marcados estándar usado para crear páginas web, junto a CSS y JavaScript, HTML es una de las tres tecnologías principales para la creación de sitios web, así mismo para crear

CAPÍTULO 2. MARCO TEÓRICO

interfaces móviles y aplicaciones web.

Los navegadores web pueden leer archivos HTML y traducirlos en páginas web visibles y audibles. HTML describe la estructura de una página web de manera semántica mediante etiquetas, esto lo hace un lenguaje de marcados en lugar de un lenguaje de programación, debido a que solo crea una estructura que realmente no ejecuta instrucciones solo sirve como guía para crear una especie de documento.

HTML permite que imágenes y otros objetos puedan ser usados para crear formularios interactivos. Esto provee formas de crear documentos estructurados utilizando semánticas de texto como cabeceras, párrafos, listas, enlaces, citas y otros items.

Los elementos HTML son puestos como etiquetas utilizando los símbolos “<”y “>”a manera de abrir y cerrar dicha etiqueta, en dichas etiquetas puede haber una gran cantidad de instrucciones que se aplicaran a aquella parte del documento que esté entre la apertura y la clausura de la etiqueta.

HTML puede llamar a scripts hechos en lenguajes de programación como JavaScript lo que afecta el comportamiento de las páginas web. HTML también puede hacer referencias a CSS para definir un estilo y una forma distinta de mostrar el contenido del documento.

2.11.4.2. CSS

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser definida en un documento separado o en el mismo documento HTML, todo estos segun [47].

2.12. Frameworks

2.12.1. Django

Según la página oficial de Django [48], Django es un web framework de código abierto y gratuito que sigue el patrón arquitectural Modelo Vista Controlador, es mantenido por la Django Software Foundation, una organización independiente sin fines de lucro.

El principal objetivo de Django es facilitar la creación de sitios web complejos especialmente aquellos que utilizan bases de datos. Django enfatiza en la reusabilidad de componentes y parte del principio “no te repitas”.

Python es usado para todo lo relacionado con el framework, ajustes, archivos e inclusive los modelos de la base de datos. Django provee un administrador opcional para crear, eliminar, actualizar y leer datos de los distintos modelos de la base de datos. Algunos sitios que usan Django incluyen Pinterest, Instagram, Mozilla y Bitbucket.

Algunas características que posee Django son:

- Un servidor propio, ligero y diseñado para desarrollo y pruebas.
- Un sistema de serialización y validación que puede traducir forms HTML a algo que pueda ser incluido en una base de datos.
- Un sistema de ORM (Object-Relational mapping, Mapeo objeto-relacional en español) el cual crea una capa por encima de la base de datos y facilita su uso, sin importar el manejador de bases de datos que se use.
- Una plantilla que utiliza el concepto de herencia visto en la programación orientada a objetos.
- Un framework de guardado en caché, que permite usar muchos métodos relacionados a la caché del sitio.

CAPÍTULO 2. MARCO TEÓRICO

- Soporte de clases que pueden intervenir en varias etapas del proceso de petición y producir funciones propias.
- Un despachador interno que permite la comunicación entre los componentes de la aplicación via señales predefinidas.
- Un sistema de internacionalización que permite varias traducciones de los componentes de Django.
- Un sistema de serialización que puede producir y leer instancias de modelos XML y JSON.
- Un sistema para extender las capacidades del motor de plantillas.
- Una interfaz que permite realizar pruebas en el sitio.
- Un sistema de autenticación extensible.
- La interfaz administrativa, Django admin.
- Herramientas para generar RSS y Atom.
- Un framework de sitios que permite a una sola instalación de Django manejar varios sitios web a la vez.
- Herramientas para generar “Google Sitemaps”.
- Incluye herramientas para evitar ataques web comunes.
- Un framework para crear aplicaciones GIS.

2.12.2. Bootstrap

Como se puede observar en [49], Bootstrap es un framework o conjunto de herramientas de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales; para mantener un aspecto moderno. Además implementa un modelo de estilos adecuado tanto para las webs de escritorio como

móviles sin tener que duplicar el código base de la aplicación.

2.12.3. Scrapy

Scrapy [50], es un framework de software libre escrito en python diseñado para hacer web scraping y web crawling, puede ser usado para extraer datos utilizando APIs ó cómo un web crawler de propósito general, la estructura de este framework está construida alrededor de “spiders”, que son crawlers auto contenidos a los que se les da un conjunto de instrucciones, y estos ejecutan dichos comandos de manera independiente entre sí.

2.13. Web scraping

Web scraping (extracción de datos en la web) es una técnica computacional de extracción de información de sitios web. Dicha técnica simula la exploración humana en la web, mediante la implementación de el protocolo HTTP u algún otro esquema de simulación [51].

Web scraping está muy relacionado con indexación web, la cual indexa información de la web utilizando un bot o un web crawler y es una técnica universal adoptada por la mayoría d los motores de búsqueda, en contraste, el web scraping se enfoca más en la transformación de datos no estructurados en la web, usualmente en formato HTML, en datos estructurados que pueden ser guardados y analizados en una base de datos central.

Web scraping está también relacionado con la automatización web, lo cual simula el proceso de búsqueda que hace un humano mientras navega por la red, los usos de web scraping incluyen comparación de precios online, monitoreo de datos del clima, detección de cambios en un sitio web, búsquedas, integración de datos, entre

otros.

2.13.1. Técnicas

Web scraping como bien se ha dicho es en resumidas cuentas el proceso de automáticamente recolectar información en la web, existen diversas técnicas que, dependiendo del caso, pueden ser usadas de una manera eficaz.

- Human copy-and-paste: A veces la mejor tecnología para realizar web scraping no puede reemplazar la capacidad que tiene un humano para analizar y extraer información de manera manual, este es el único método utilizable si el sitio web en cuestión bloquea el resto de técnicas de extracción que si están automatizados.
- Matching de expresiones regulares: Un método simple pero a la vez muy poderoso para extraer información sobre páginas web, se basa en simplemente verificar que un texto o grupos de textos coincidan con una expresión regular.
- Programación HTTP: Utilizando programación de sockets, se puede obtener páginas completas con tan solo usar peticiones HTTP desde un servidor web remoto.
- Crawler: o arañas web es un programa que inspecciona las páginas del World Wide Web de forma metódica y automatizada. Las arañas web comienzan visitando una lista de URLs, identifica los hiperenlaces en dichas páginas y los añade a la lista de URLs a visitar de manera recurrente de acuerdo a determinado conjunto de reglas. La operación normal es que se le da al programa un grupo de direcciones iniciales, la araña descarga estas direcciones, analiza las páginas y busca enlaces a páginas nuevas. Luego descarga estas páginas nuevas, analiza sus enlaces, y así sucesivamente. Este método suele usarse para extraer información de los sitios web.

CAPÍTULO 2. MARCO TEÓRICO

- **Parsers HTML:** Muchas páginas web tienen colecciones grandes de páginas generadas dinámicamente desde una fuente estructurada, como una base de datos. Los datos de categoría similar son típicamente encontrados en páginas similares con un script ó plantilla en común. En minería de datos, un programa que detecta dichas plantillas en una fuente particular de información, el cual extrae su contenido y lo traduce a una forma relacional se le conoce como wrapper.

Un wrapper entonces puede ser utilizado para comparar varias páginas web, varios usuarios, entre otras cosas, en busca de elementos en común.

- **Parsing de DOM:** Consiste en que un navegador puede dar el contenido dinámico generado por los scripts del lado del cliente, se pueden crear programas que recorran y verifiquen estos datos para posteriormente extraerlos.
- **Software de web scraping:** Hay muchas herramientas disponibles que pueden ser utilizadas para personalizar el algoritmo de web scraping. Estas herramientas pueden intentar automáticamente reconocer las estructuras de datos de una página o proveer de una interfaz que elimine la necesidad de escribir código web scraping de manera manual.
- **Reconocimiento de información semántica:** Las páginas que son analizadas podrían incluir meta-datos o cierta información semántica como anotaciones ó comentarios, los cuales pueden ser usados comúnmente. Si estas anotaciones están en las mismas páginas, como sucede con los micro-formatos, estas podrían ser de utilidad cuando se recorra el DOM del documento. En otro caso, las anotaciones, organizadas en una capa semántica, son almacenadas y manejadas de forma separada desde otras páginas, por lo que el web scraper puede recuperar estos esquemas y las instrucciones desde esta capa antes de analizar los documentos.

- Analizador de páginas web: Existen esfuerzos utilizando machine learning que intentan identificar y extraer información de páginas web simulando la forma de verlas de un humano.

2.14. Librerías y paquetes

2.14.1. jQuery

jQuery, según [52], es una librería de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la licencia MIT y la licencia pública general de GNU, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

jQuery consiste en único archivo JavaScript que contiene funcionalidades comunes de DOM, eventos, efectos y AJAX.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX.

La forma de interactuar con la página es mediante la función `jQuery()`, que recibe como parámetro una expresión CSS o el nombre de una etiqueta HTML y devuelve todos los nodos (elementos) que concuerden con la expresión. Esta expresión es denominado selector en la terminología de esta biblioteca.

2.14.2. NumPy y SciPy

NumPy [53], es el paquete fundamental para la computación científica en Python. NumPy es una extensión de el lenguaje de programación Python, agregando soporte para arreglos multidimensionales de gran tamaño, así mismo posee una librería con funciones matemáticas de alto nivel para operar en estos arreglos. Junto con este paquete hay una librería que hace que juntos desarrollen todo su potencial, la librería SciPy [54], es una librería diseñada para correr complejas operaciones matemáticas de manera simple y eficiente.

3

Trabajos previos

En esta sección se presentan algunos sitios o paginas web sobre programación competitiva usados para entrenar y practicar para las competencias. Además de esto también se presentan sitios que poseen implementaciones de sistemas de recomendación para cumplir sus objetivos.

3.1. Paginas de Programación competitiva

Existen muchos sitios dedicados al entrenamiento de programación competitiva, desde simples blogs donde se discute sobre dicho tema y se encuentran guías para aprender las técnicas necesarias, hasta sitios mas complejos que tienen a su disposición diversos recursos que no solo incluyen guías sino también jueces online y gran cantidad de problemas alojados disponibles para resolver en cualquier momento. A continuación se presentan los sitios web mas usados para entrenar programación competitiva junto con sus características mas destacadas que ayudan a cumplir con

el objetivo del entrenamiento.

3.1.1. Top Coder

Top coder es una comunidad bastante amplia que no solo abarca la programación competitiva, sino que abarca a las ciencias de la computación en general. En el área de la programación competitiva top coder no solo presenta las competencias tipo SRM descritos anteriormente en la sección 2.4.3 sino también un conjunto de características adicionales que están detalladas a continuación [55]:

- Top coder presenta una gran cantidad de guías sobre las técnicas mas usadas en programación competitiva, tales como programación dinámica, teoría de grafos, algoritmos de búsqueda, recursividad, geometría, matemática, entre otros; además también presenta numerosos consejos y recomendaciones sobre como afrontar los problemas y los enfoques adecuados para resolverlos.
- Top coder tiene un sistema de clasificación que consta de dos divisiones según el nivel que ha alcanzado el usuario, siendo la división 1 para los usuarios mas experimentados y la división 2 para los usuarios novatos. Además también se clasifica a los usuarios con un numero de “rating” que va desde 0 clasificando por colores según la puntuación, siendo gris en el rango 0-899, verde 900-1199, azul 1200-1499, amarillo 1500-2199 y rojo 2200 o mayor.
- Los problemas usados en las competencias SRM quedan archivados para poder visualizarlos e intentar resolverlos posteriormente, estos además están clasificados según la división, el nivel y la categoría del problema.

3.1.2. Spoj

Spoj es una página creada con el objetivo de enseñar a las personas a programar a un nivel más alto, Spoj posee varios módulos que han ido incrementando en número en el tiempo: [56]

- Posee una enorme cantidad de problemas a resolver, así mismo permite a usuarios comunes agregar nuevos problemas a alguna de sus categorías.
- Posee una herramienta para realizar maratones de practica con problemas del sitio, así como un foro para la competencia.
- Posee una pequeña guía de problemas recomendados para principiantes.
- Permite una modalidad para evaluar estudiantes de una manera automatizada con un mínimo esfuerzo.
- Posee rankings, basados en el puntaje de los usuarios que han resuelto problemas según su categoría y el número de personas que han resuelto dichos problemas, las categorías son:
 - Classical: Son problemas con un juez binario. Valen para el puntaje.
 - Tutorial: Son problemas con un juez binario cuya dificultad es inferior a los problemas “Classical”, están diseñados para el aprendizaje, no valen para el puntaje.
 - Challenge: Son problemas con un juez especial que se basa en dar un puntaje según qué tan buena en ciertos términos fue tu solución, usualmente son problemas clásicos que requieren modificaciones como, por ejemplo, disminuir las líneas de código ó el número de librerías usadas. Valen para el puntaje

CAPÍTULO 3. TRABAJOS PREVIOS

- **Partial:** Son problemas de la misma indole que los “Challenge” pero son con el objetivo de aprender no de competir, son muy usados como evaluaciones en universidades. No valen para el puntaje.
- **Riddle:** Un problema sin enunciado ó con uno que no explica el problema como tal (a propósito) , el competidor debe adivinar la tarea a resolver viendo solamente la entrada y la salida del programa esperado.

3.1.3. A2 Online Judge

Es un sitio dedicado a la creación de maratones de práctica, creado por Ahmed Aly, tiene un enorme reparto de opciones para la creación de competencias, y registro de progreso de sus usuarios [57].

- A2 permite crear y participar en maratones de práctica los cuales se muestran en el frontpage de la página, sean privados ó no, las personas pueden ver cuando comienzan y cuando terminan, y , en ocasiones, los resultados y los problemas usados en la práctica, además A2 permite agregar problemas de cualquier página que sea soportada por el sitio.
- Los usuarios tienen un registro de todos los problemas resueltos por ellos, todos estos problemas son problemas directamente en A2 ó problemas de cualquier página soportada por el sitio.
- Permite poder llevar las estadísticas de los competidores a lo largo de varias competencias en el sitio y otros agregados menores.

3.1.4. USACO

USACO (USA Computing Olimpiad) fue creado en 1992 por el Dr. Don Piele, profesor de matemáticas de la Universidad de Wisconsin Parkside, con el fin de entrenar y llevar equipos de USA a la IOI. Actualmente esta plataforma esta disponible de manera gratuita y abierta a todo el publico para entrenar y participar en los concursos que se organizan. USACO cuenta con las siguientes características [58].

- USACO realiza concursos mensuales desde el mes de diciembre hasta abril, los concursos duran de 3 a 5 horas para resolver de 3 a 4 problemas, aunque para mayor flexibilidad se puede elegir cualquier momento entre el viernes y el lunes (de las fechas establecidas) para participar. El concurso se hace vía online y se envían las soluciones a través de una simple interfaz web.
- Los concursos están divididos en 4 categorías según su dificultad las cuales se describen a continuación:
 - Bronce: esta enfocado en estudiantes que recientemente han aprendido a programar y que por lo tanto no tienen conocimientos mas allá de los conceptos básicos como la clasificación y la búsqueda binaria.
 - Plata: enfocado a los estudiantes que están empezando a aprender las técnicas fundamentales para resolver problemas (por ejemplo, búsqueda recursiva, algoritmos greedy) y estructuras de datos fundamentales.
 - Oro: Para los estudiantes con conocimientos de algoritmos de naturaleza más compleja (por ejemplo, los caminos más cortos, programación dinámica) y estructuras de datos más avanzadas.
 - Platino: para estudiantes avanzados que poseen conocimientos mas desarrollados en técnicas de resolución de problemas algorítmicos y que desean

enfrentar retos con problemas mas complicados y abiertos.

- USACO posee el modulo USACO Training creado para aprender las distintas técnicas y otras consideraciones a tener en cuenta en las competencias de programación (enfocado en competencias tipo IOI) dicho modulo presenta una guía con varias secciones donde se va explicando, sección por sección, una técnica o enfoque necesario para las competencias, además de dispones de problemas de ejemplo para resolver una vez aprendido el tema de la sección. Es importante destacar que las secciones se tienen que desbloquear habiendo resuelto los problemas de la sección anterior.

3.1.5. Uhunt

Es una herramienta creada por Félix Halim financiada por él mismo, diseñada para ayudar a las personas a encontrar problemas interesantes en Uva online judge, así como llevar estadísticas de problemas resueltos por los usuarios, uhunt posee una web API para que otros desarrolladores puedan usarla para acceder a problemas de Uva de manera sencilla. [59], a continuación se presentan algunas características.

- Uhunt ha categorizado los problemas de su lista de la misma manera en la que está organizado el libro “Competitive Programming III” para su fácil acceso y seguimiento en el sitio.
- Uhunt posee una visualización de los envíos en UVA de cada usuario, el cual está mucho mejor trabajado que el de UVA.
- Uhunt posee una visualización con todos los problemas de UVA ordenados por volúmenes, en el cual también hay un seguimiento de tu progreso.
- Uhunt tiene la opción de crear maratones de práctica usando problemas reco-

mendados por otros usuarios ó directamente problemas que se quieran colocar.

3.1.6. COJ

COJ fue creado por un grupo de estudiantes y profesores de de la Universidad de las Ciencias Informáticas en La Habana, Cuba, con el objetivo de entrenar para participar en las competencias de programación, mejorar y compartir conocimientos en la resolución de problemas, la programación y el trabajo en equipo e intercambiar todas estas experiencias y conocimientos. Para esto COJ cuenta con las siguientes características [60].

- COJ Tiene archivado gran cantidad de problemas disponibles para practicar y enviar en su propio Juez online.
- En COJ se realizan competencias relacionadas con el entrenamiento o selección de los equipos a participar en la ACM-ICPC en Cuba, algunos de estos están públicos con libre acceso para que personas de otros países participen.
- COJ tiene un sistema para comunicarse con otros usuarios a través de correos electrónicos presente en la pagina web. Esto permite una mayor interacción con los demás usuarios a la hora de compartir información o experiencias.

3.1.7. Codeforces

Tal y como se comenta en la sección 2.4.4 Codeforces es una red social dedicada a la programación competitiva y no solo se hacen competencias sino que tiene otras funcionalidades para promover el entrenamiento en competencias de programación, la formación y el aprendizaje [61].

CAPÍTULO 3. TRABAJOS PREVIOS

- Codeforces permite practicar intentando nuevamente los problemas de competencias pasadas ya sea individualmente o creando una competencia virtual para simular la mayoría de las reglas de la competencia original.
- Los problemas pasados están categorizados según la técnica para resolverlo, de tal manera que esto sirva como pista para los usuarios que intenten resolverlo.
- Codeforces tienen una sección denominada “Gym” en la cual permite crear competencias personalizadas con problemas alojados en Codeforces, además de que estos quedan disponibles para que cualquier pueda volver a participar e intentar los problemas presentes en la competencia. Además tiene un buscador para facilitar así encontrar competencias según el tipo (basado en las reglas), duración y dificultad.
- Codeforces tiene un sistema de “Rating” bastante parecido al de TopCoder, basado en colores y 2 divisiones. La división 2 dirigida a los principiantes donde los rangos van de 0 a 1199 (Color Gris, Newbie), 1200 a 1399 (Color Verde, Pupil), 1400 a 1599 (Color Cían, Specialist), 1600 a 1899 (Color Azul, Expert); La división 1 dirigida a competidores mas experimentados donde los rangos van de 1900 a 2199 (color Violeta, Candidate Master), 2200 a 2299 (Color Naranja, Master), 2300 a 2399 (Color Naranja oscuro, International Master), 2400 a 2599 (Color Rojo, Grandmaster), 2600 a 2899 (Color Rojo, International Grandmaster) y 2900 o mas (Color Rojo oscuro, Legendary Grandmaster).
- Cada usuario tiene un blog en donde puede publicar entradas y así compartir alguna información con los demás usuarios o solicitar ayuda con algún tópico, en otras palabras es un medio de comunicación entre los usuarios para el intercambio de ideas.

3.2. Páginas web con sistemas de recomendación

3.2.1. Amazon

Amazon es un sitio web dedicado a la compra y venta de productos, es actualmente una de las más grandes de este estilo en el mundo, utiliza un sistema de recomendación para recomendar productos en base a productos que han interesado al usuario que son similares a los ofertados por la página, para ello también utiliza las compras recientes, las calificaciones que el usuario a dado a productos de ese estilo, las listas de deseados, entre otros.

3.2.2. Hackerrank

Hackerrank es un sitio web dedicado a las competencias de programación, a pesar de realizar maratones de manera frecuente, su fuerte radica en su estructura hecha para la fácil enseñanza y aprendizaje de distintas técnicas y habilidades necesarias para la programación competitiva, para ello utiliza un sistema de recomendación de problemas, el cual se basa en verificar qué problemas han sido resueltos por el usuario para dar una lista de problemas recomendados no resueltos por él, para esto se basa en revisar lo que otros usuarios han resuelto habiendo superado los mismos retos que el usuario base, es decir aquellos usuarios que han resuelto cierto problema usualmente resuelven otro problema que el usuario no ha completado entonces se le recomienda resolverlo.

Hackerrank tiene algunas desventajas claves, entre ellas el hecho de que no recomienda en base a la dificultad, que muchos problemas en sí mismos están mal clasificados ya sea en dificultad ó en la técnica necesaria para resolverlos, y Hacke-

CAPÍTULO 3. TRABAJOS PREVIOS

rank no permite a la comunidad ayudar a clasificar los problemas lo cual produce lo anteriormente estipulado.

Hackerrank por otro lado tiene mucho contacto con el mundo empresarial, así mismo posee muchas competencias patrocinadas por empresas de todo el mundo que buscan nuevos talentos.

3.2.3. Youtube

El sitio web Youtube es una red social donde los usuarios comparten contenido creado por ellos mismos en forma de vídeos y contenido multimedia, utiliza un sistema de recomendación para recomendar vídeos de un tema específico que interese al usuario basado en los vídeos que ha visto y que le han gustado, también vídeos que tengan gran cantidad de vistas ó votos positivos.

3.2.4. LinkedIn

Utilizando retroalimentación de los usuarios, LinkedIn, que es una red social que se especializa en poner en contacto profesionales y empresas para generar empleos, verifica las habilidades expuestas por los postulantes y recomienda a los que mejor se ajusten a un perfil buscado por el contratante guiándose además por el número de referencias y qué tan importantes son los usuarios con dichas referencias.

3.2.5. NetFlix

NetFlix es un servicio pago que permite mirar películas o series online , utiliza un sistema de recomendación que se basa en las series ó películas que has visto y

CAPÍTULO 3. TRABAJOS PREVIOS

te recomienda otras del mismo género ó , alguna más popular o con mejor calificación.

4

Diseño

Este capítulo tiene como objetivo mostrar el proceso de diseño de la aplicación, presentando la metodología y arquitectura empleada, el modelo de datos y los procesos o funcionalidades que se llevan a cabo en la aplicación.

4.1. Metodología

Como metodología de trabajo, se aplicó una metodología Ad-Hoc orientada a prototipos, debido a que es una metodología bastante flexible que consiste en la realización continua de diversos prototipos cada vez más refinados con el fin de incrementar la comprensión que tiene del sistema tanto el usuario como el desarrollador. Además esta metodología se empleó siguiendo el proceso de desarrollo de un sistema de minería de datos, ya que el desarrollo de la aplicación está estrechamente relacionado con dicho enfoque.

4.2. Arquitectura

En esta sección se describen los principales componentes de la aplicación para en futuras secciones describirlas a fondo.

Se optó por una arquitectura Cliente-Servidor estándar para la creación de una aplicación web.

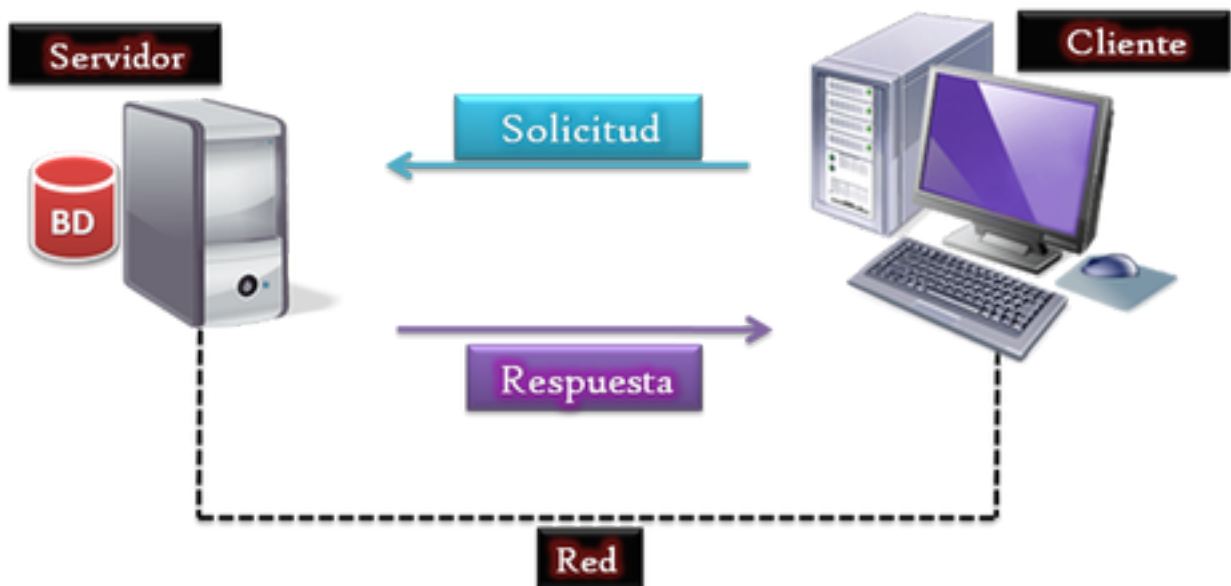


Figura 4.1: Cliente-Servidor

Cada parte contiene los siguientes elementos:

- Cliente

Del lado del cliente se tiene toda la interfaz de la aplicación, en la cual se presentan todas las funciones que puede ejecutar el usuario y todo aquello que se le muestra tales como el perfil del usuario, las distintas secciones, los tutoriales entre otras funciones.

■ Servidor

Del lado del servidor se ejecuta toda la lógica de la aplicación además de almacenar y proveer la información solicitada por el cliente, en el servidor se tienen los crawlers, recomendadores, la base de datos y todo el manejo de la aplicación realizado por el framework Django.

La aplicación fue realizada utilizando Python como lenguaje de programación principal y usando el framework Django, que facilita la creación de aplicaciones web de manera rápida y segura, ya que con tan solo unas configuraciones iniciales se puede comenzar directamente a desarrollar la aplicación. Además Django provee su propio ORM que facilita el manejo con la base de datos, en cual se puede utilizar algún motor de base de datos como base del cual seleccionamos el motor de MySQL. También se hizo uso de la herramienta Scrapy para realizar los crawler que se encargan de extraer los datos de los problemas y los usuarios en jueces externos.

4.3. Modelo de datos

A continuación se expone el modelo de datos utilizado para representar los diferentes objetos o entidades que componen la información manejada por el sistema en su totalidad.

Se seleccionó un diagrama Entidad-Relación como la mejor manera para exponer el modelo de datos, debido a que es la manera más sencilla de representar gráficamente el mismo, a pesar de que este modelo Entidad-Relación no esté totalmente ligado a la aplicación debido al uso del ORM que proporciona Django para el manejo de datos.

Con la intención de mejorar la legibilidad, el diagrama esta dividido en "módulos" que representan las funcionalidades del sistema.

4.3.1. Cuentas de Usuario

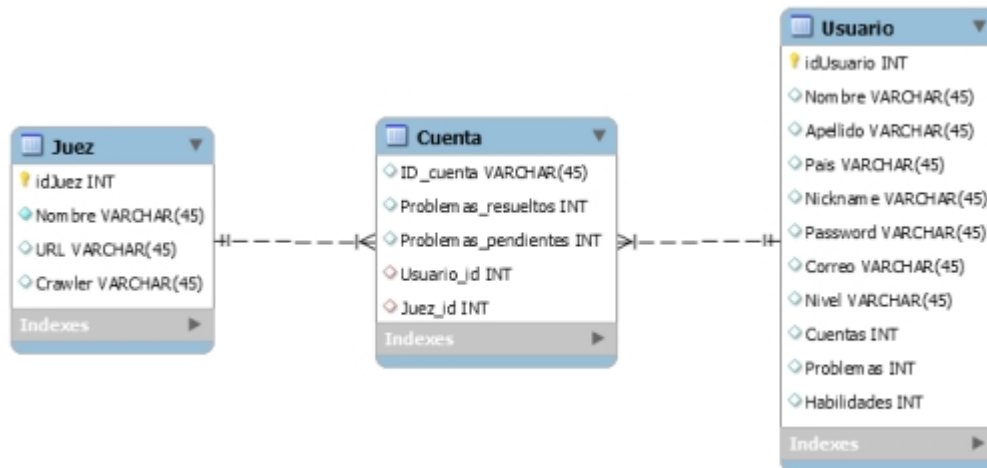


Figura 4.2: Relacion Usuario-Juez.

Una de las principales funciones de la aplicación es permitir al usuario enlazar con cuentas de jueces online de programación competitiva, para cumplir con este objetivo se requieren las entidades mostradas en la figura 4.2. La entidad "Usuario" se encarga de almacenar todos los datos personales de los usuarios, además de información necesaria para el funcionamiento de otros aspectos de la aplicación como lo es el atributo "Nivel", el cual determina el nivel general que tiene un usuario según los problemas que haya resuelto. La entidad "Juez" por otro lado se encarga de guardar la información respectiva de los jueces y posee el atributo "Crawler", este tiene el nombre del proceso que ejecuta el Crawler (encargado de extraer los problemas) del respectivo juez. Por último se tiene una relación Usuario-Juez (Relación Muchos a Muchos) llamada "Cuenta" la cual indica los id de las cuentas del

CAPÍTULO 4. DISEÑO

usuario en cada juez online, además de otros datos importantes como los problemas resueltos del usuario en un juez respectivo.

4.3.2. Problemas y Categorías

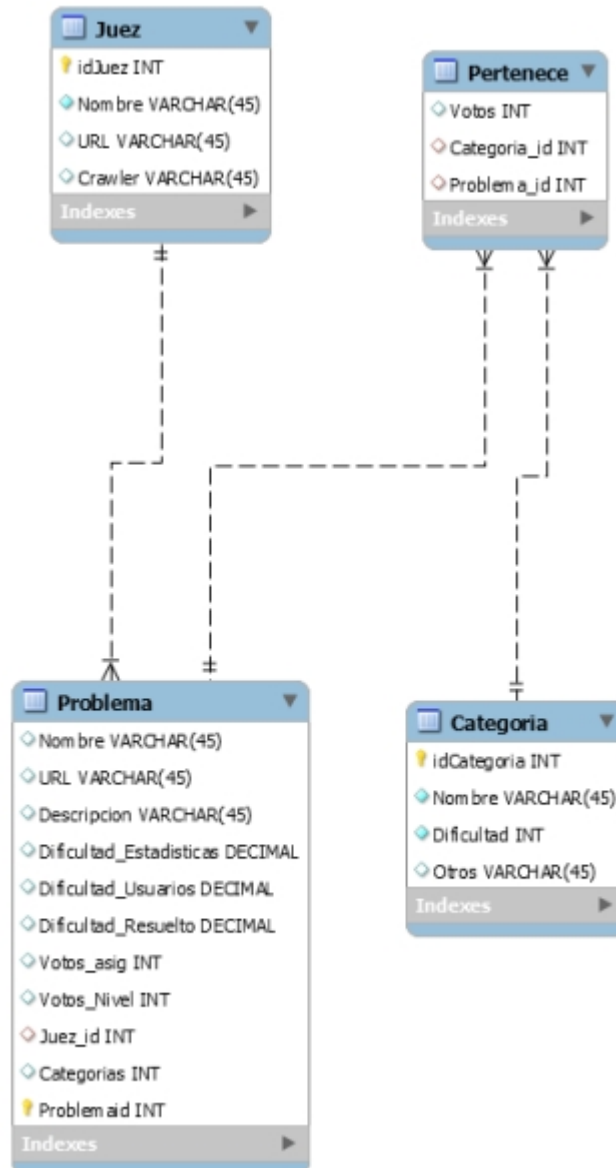


Figura 4.3: Problemas y Categorías.

CAPÍTULO 4. DISEÑO

Para representar los problemas y las categorías a los cuales podrían pertenecer se usaron las entidades mostradas en la figura anterior 4.3, donde se tiene la entidad "Categoría" la cual se describe con su respectivo nombre y un atributo "Otros" que almacena otros posibles nombres que puede tener la categoría, además se encuentra la entidad "Problema" que posee información básica del problema junto con atributos como "Dificultad_Estadísticas", "Dificultad_Usuarios" y "Dificultad_Resuelto" que son usados para determinar la dificultad del problema según distintos criterios; Atributos como "Votos_asig" y "Votos_Nivel" son usados solo para facilitar algunos cálculos en la aplicación.

Con respecto a las relaciones, se tiene una relación muchos a muchos (llamada "Pertenece") entre la entidad "Categoría" y "Problema" puesto que un problema podría pertenecer a distintas categorías y por ultimo la relación entre "Problema" y "Juez" la cual es de uno a muchos puesto que cada problema solo esta alojado en un juez.

4.3.3. Resolución de Problemas

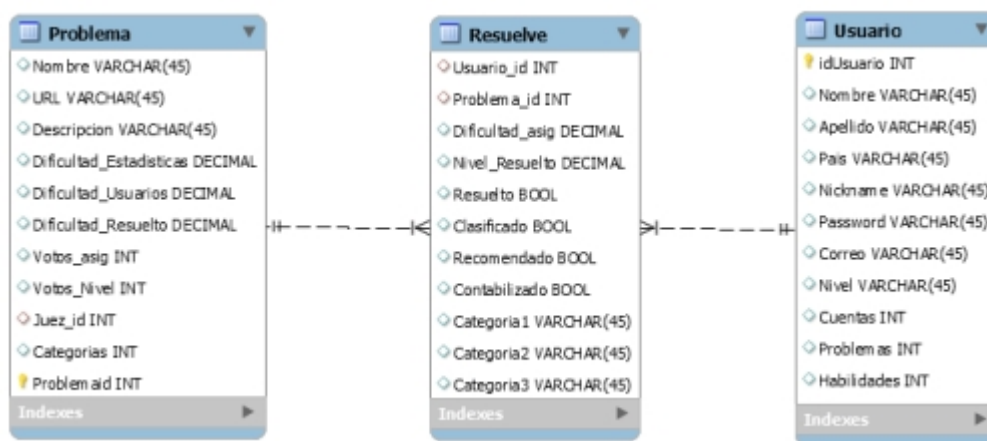


Figura 4.4: Relacion Problema-Usuario.

CAPÍTULO 4. DISEÑO

Como se puede observar en la figura 4.4, se tiene una relación entre "Usuario" y "Problema", esta entidad llamada "Resuelve" tiene distintas funciones:

- Registrar todos los problemas que hayan sido resueltos por el usuario haciendo uso de del atributo "Resuelto", además de guardar el nivel que tenía el usuario cuando resolvió el problema.
- Almacenar la información que el usuario proporcione sobre los problemas ya resueltos por el mismo. Para esto se hace uso de los atributos: "Dificultad_asig" que almacena la dificultad que el usuario cree que debería tener el problema, "Categoría1", "Categoría2" y "Categoría3" las cuales permiten al usuario asignar las categorías a las cual el cree que pertenece un problema ya resuelto y por ultimo el atributo "Clasificado" que indica si ya el usuario ha proporcionado estos datos o no.
- Indicar cuales problemas no resueltos por el usuario serán recomendados usando el atributo "Recomendado".

Por ultimo el atributo "Contabilizado" solo sirve para agilizar algunos cálculos al momento de estimar la dificultad de los problemas según distintos criterios.

4.3.4. Habilidades

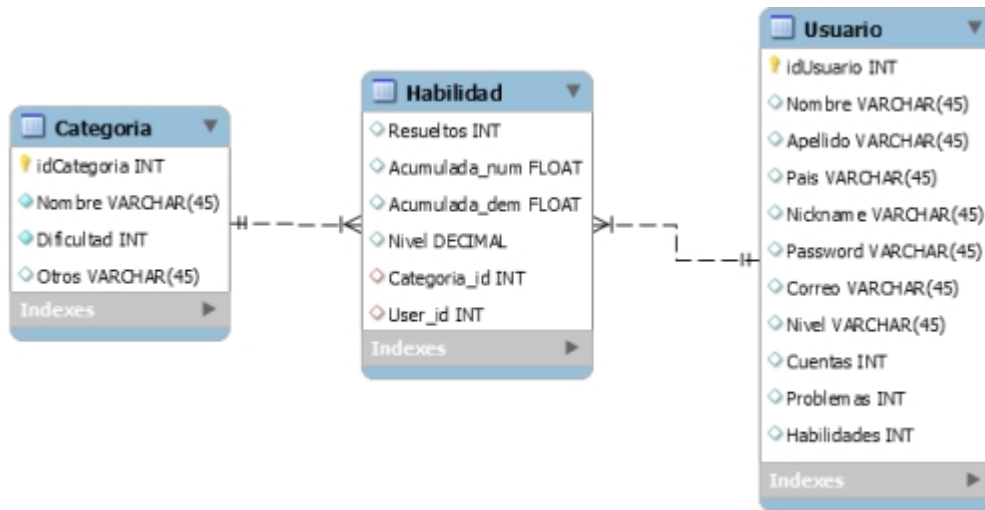


Figura 4.5: Relación Categoría-Usuario.

Para esta entidad "Habilidad", la cual es una relación muchos a muchos entre "Categoría" y "Usuario" que se encarga de almacenar información que es usada para la recomendación de problemas, ya que se estima el nivel del usuario en una respectiva categoría, para esto se usan la cantidad de problemas que ha resuelto el usuario en una categoría específica y el nivel de dificultad de cada que problema resuelto, calculando un promedio ponderado que dará como resultado el nivel del usuario en la categoría específica.

4.3.5. Tutoriales

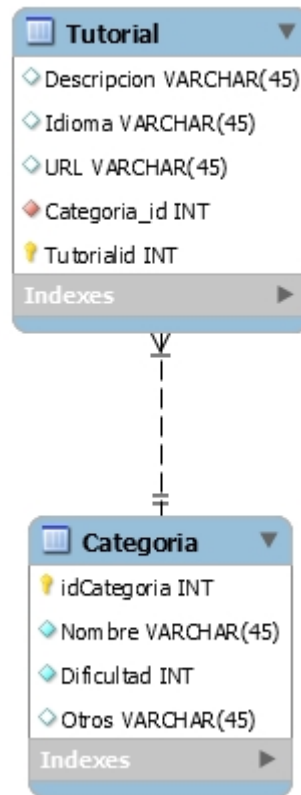


Figura 4.6: Relacion Tutorial-Categoría

Por ultimo se tiene la entidad "Tutorial", encargada de almacenar la información de los enlaces a tutoriales o guías externas de los cuales dispone la aplicación. Esta entidad mantiene una relación unos muchos junto con la entidad "Categoría" ya que los tutoriales pertenecen a alguna categoría específica.

4.4. Procesos y Vistas

Esta sección tiene como objetivo mostrar los diversos procesos de la aplicación a la hora de ser usada por los usuarios y administradores, además de presentar una

CAPÍTULO 4. DISEÑO

vista de la interfaz donde se ejecuta dicha funcionalidad o proceso.

4.4.1. Registro

El usuario crea una cuenta para poder utilizar la aplicación, en esta sección de registro el usuario proporciona su nombre, apellido, correo electrónico, nombre de usuario, contraseña y país al que representa. Posteriormente debe leer y aceptar las condiciones de uso de la aplicación y hacer click en el botón registrar.



The image shows a user registration form with the following fields and labels:

- Nombre:** Enter First Name
- Apellido:** Enter Last Name
- Correo:** Enter email address
- Username:** Enter Username (with an example: yourname@domain.com)
- Password:** Enter Password
- Confirma el Password:** Enter Password
- Pais:** Afganistán

At the bottom, there is a checkbox labeled "Acepto los termino y condiciones" and a vertical column of five lock icons on the right side of the form.

Figura 4.7: Registro de usuario.

4.4.2. Inicio de sesión

El usuario inicia sesión en la aplicación introduciendo su nombre de usuario y contraseña y posteriormente da click en el botón "Log in".



Figura 4.8: Inicio de sesión.

4.4.3. Cerrar sesión

El usuario con una sesión iniciada puede hacer click en el botón "cerrar sesión" para cerrar la sesión.

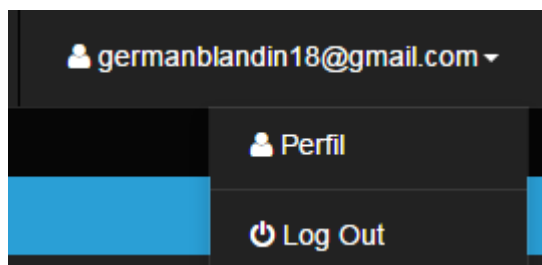


Figura 4.9: Cerrar sesión.

4.4.4. Perfil de Usuario

En esta sub-sección el usuario puede ver los datos introducidos por él en el registro, así mismo puede cambiar sus cuentas en los jueces online en caso de que al momento del registro no tuviese alguna de esas cuentas ó se haya equivocado. También se pueden apreciar distintos elementos en los cuales se incluyen:

CAPÍTULO 4. DISEÑO

- Nivel del usuario: refleja el nivel del usuario en distintas categorías una vez que da click en el botón "Calcular nivel" , cada vez que quiera actualizar su nivel deberá dar click en este botón.



→ Habilidad del Usuario

Nivel general: 2.9

Ad-Hoc:	2.4	Algoritmos_constructivos:	3.0	Arboles:	4.2
Busqueda_Binaria:	3.1	Busqueda_ternaria:	0.0	Camino_mas_corto:	4.4
Combinatoria:	3.5	DFS:	3.8	Divide_y_venceras:	3.0
DP:	3.9	DSU:	4.3	Estructuras_con_Strings:	4.0
Estructuras_de_Datos:	3.8	Expresiones_parseo:	2.6	FFT:	0.0
Flujo:	0.0	Fuerza_Bruta:	2.8	Geometria:	4.4
Greedy:	2.9	Hashing:	3.3	Mascara_de_bits:	3.8
Matching:	3.8	Matematica:	3.5	Matrices:	4.7
Meet_in_the_middlle:	2.0	Ordenamiento:	3.3	Probabilidades:	4.0
Simulacion:	0.0	Strings:	2.6	Teorema_chino_del_resto:	0.0

Figura 4.10: Habilidades del usuario.

- Lista de problemas resueltos: El usuario posee una lista de sus problemas resueltos, la cual puede ser actualizada dando click en "Actualizar problemas".

CAPÍTULO 4. DISEÑO

Problemas Resueltos 493					
Uva-44	Uva-47	Uva-52	Uva-57	Uva-83	Uva-88
Uva-102	Uva-126	Uva-127	Uva-132	Uva-152	Uva-167
Uva-183	Uva-195	Uva-293	Uva-314	Uva-378	Uva-379
Uva-422	Uva-438	Uva-448	Uva-503	Uva-528	Uva-615
Uva-728	Uva-774	Uva-777	Uva-778	Uva-861	Uva-880
Uva-919	Uva-924	Uva-931	Uva-944	Uva-977	Uva-996
Uva-1012	Uva-1015	Uva-1019	Uva-1071	Uva-1072	Uva-1136
Uva-1183	Uva-1191	Uva-1202	Uva-1204	Uva-1239	Uva-1247
Uva-1254	Uva-1278	Uva-1288	Uva-1298	Uva-1341	Uva-1346
Uva-1387	Uva-1406	Uva-1407	Uva-1410	Uva-1418	Uva-1437
Uva-1461	Uva-1475	Uva-1526	Uva-1557	Uva-1605	Uva-1608
Uva-1625	Uva-1629	Uva-1662	Uva-1696	Uva-1714	Uva-1760
Uva-1768	Uva-1773	Uva-1806	Uva-1843	Uva-1851	Uva-1853
Uva-1868	Uva-1884	Uva-1921	Uva-1932	Uva-1944	Uva-1967

Figura 4.11: Problemas Resueltos.

- Lista de problemas no clasificados: Se puede observar una lista de los problemas resueltos por el usuario que aun no han sido clasificados por él, al darle click en cualquiera de ellos se le llevará a una nueva pestaña donde podrá clasificar el problema.
- Problemas recomendados: sub-sección en donde al hacer click en el botón "Recomendar" el usuario recibirá una lista de problemas recomendados según su nivel.

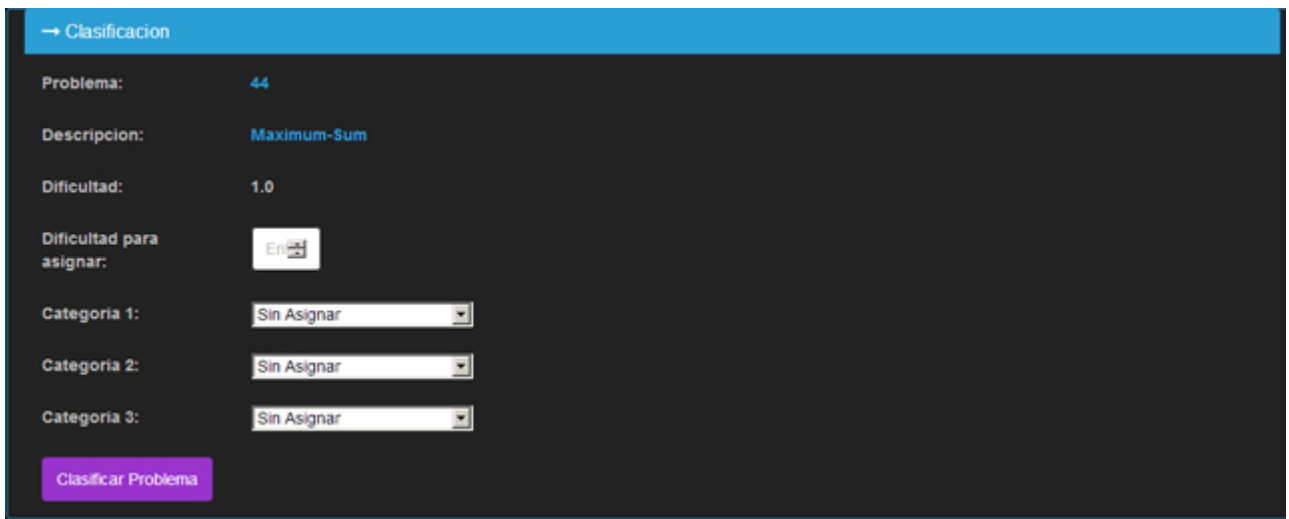


Figura 4.12: Problemas sin clasificar y recomendaciones.

Todos estos datos son visibles por los administradores y otros usuarios, exceptuando las recomendaciones y problemas no clasificados.

4.4.5. Clasificación

El usuario introduce la dificultad que cree conveniente para el problema, un número entre uno y diez, posteriormente coloca las categorías a las que cree que pertenece el problema, luego da click en aceptar para almacenar estos datos sobre el problema. En el diagrama los datos requeridos tienen un asterisco.



The image shows a web interface for classifying a problem. At the top, there is a blue header with a right-pointing arrow and the text "Clasificacion". Below this, the form is set against a dark background. It contains several fields: "Problema:" with the value "44"; "Descripcion:" with the value "Maximum-Sum"; "Dificultad:" with the value "1.0"; "Dificultad para asignar:" with a small white button labeled "En"; and three dropdown menus for "Categoria 1:", "Categoria 2:", and "Categoria 3:", all of which currently show "Sin Asignar". At the bottom left of the form is a purple button labeled "Clasificar Problema".

Figura 4.13: Clasificación de problemas.

4.4.6. Buscador de problemas

El usuario puede colocar la categoría que desea buscar, la dificultad mínima y máxima de los problemas a ser mostrados y posteriormente hacer click en "Buscar problemas", se muestra una lista de problemas no resueltos por el usuario con las especificaciones dadas.

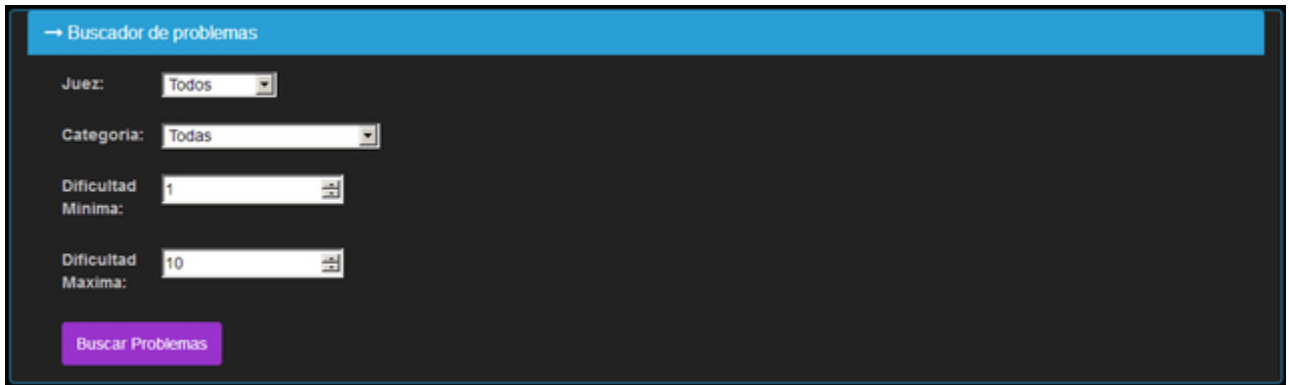


Figura 4.14: Buscador de problemas.

4.4.7. Tutoriales

En esta sección el usuario puede encontrar diversos tutoriales relacionados con la programación competitiva, los cuales están en forma de vínculos a las páginas web donde están alojados dichos tutoriales con una pequeña descripción, los tutoriales son agregados por los administradores únicamente.



Figura 4.15: Tutoriales.

4.4.8. Recomendaciones

A pesar de que en el perfil de usuario se pueden solicitar recomendaciones, el sistema también permite hacer recomendaciones mas especificas usando los distintos métodos para realizar las recomendaciones:

- Recomendaciones basadas en contenido: en esta el usuario debe seleccionar una categoría especifica de la cual quiere solicitar las recomendaciones, también permite obtener la categoría seleccionando la mejor y peor habilidad que posea el usuario en el sistema. Una vez solicitada el sistema proporciona las recomendaciones basadas en la categoría y el nivel del usuario en dicha categoría.

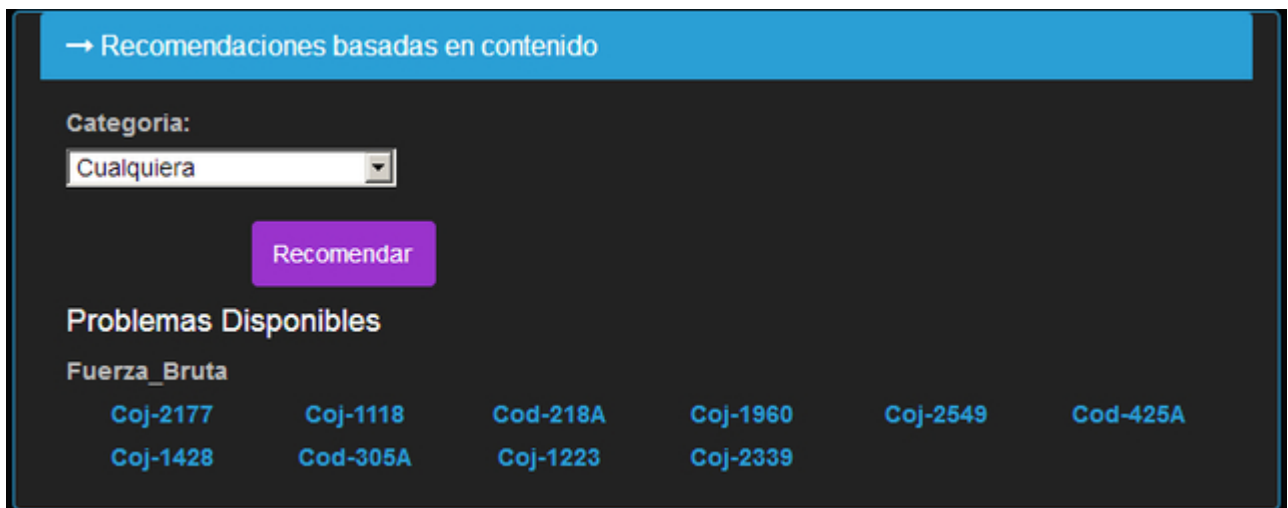


Figura 4.16: Recomendaciones basadas en contenido.

- Recomendaciones usando filtros colaborativos: el usuario selecciona la métrica a emplear en el algoritmo de los k-vecinos para ejecutar las recomendaciones basadas en usuarios similares. El sistema presenta tanto los usuarios mas simi-

lares como los problemas que recomienda en base a ellos.

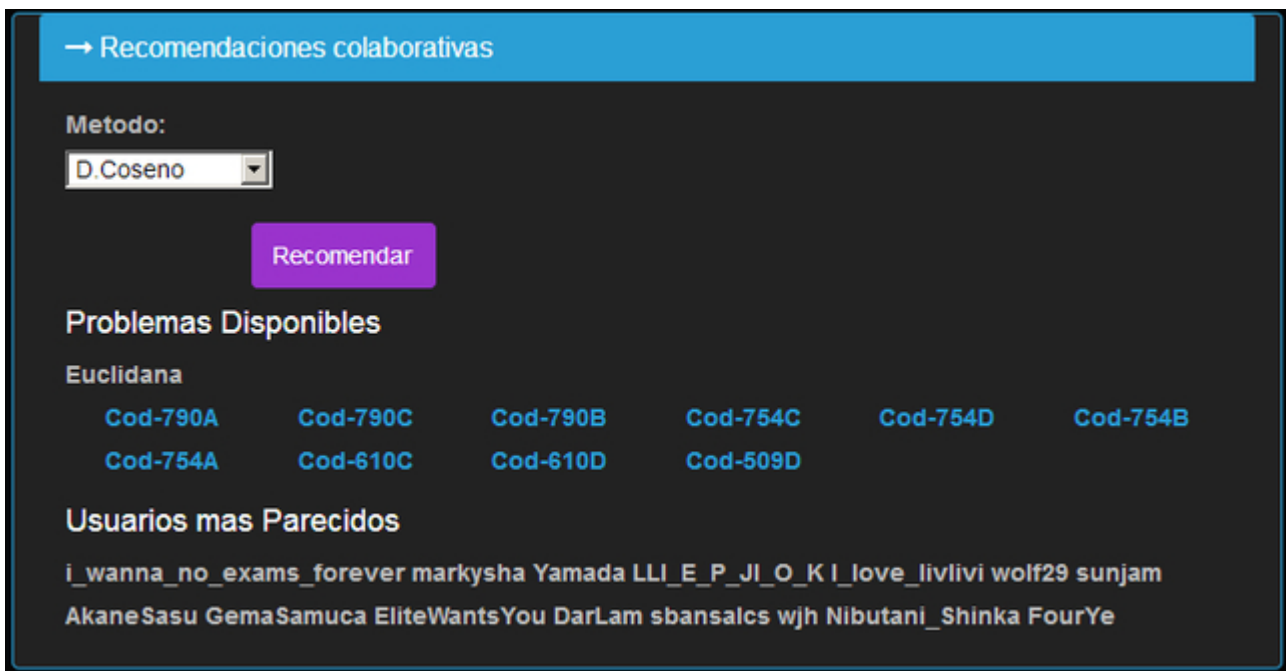


Figura 4.17: Recomendaciones usando filtro colaborativo.

- Recomendaciones usando el algoritmo apriori: el usuario proporciona los valores de soporte y confianza para ejecutar el algoritmo apriori. El sistema, una vez ejecutado el algoritmo, proporciona los problemas que están mas relacionados entre si y han sido resueltos muchos usuarios.

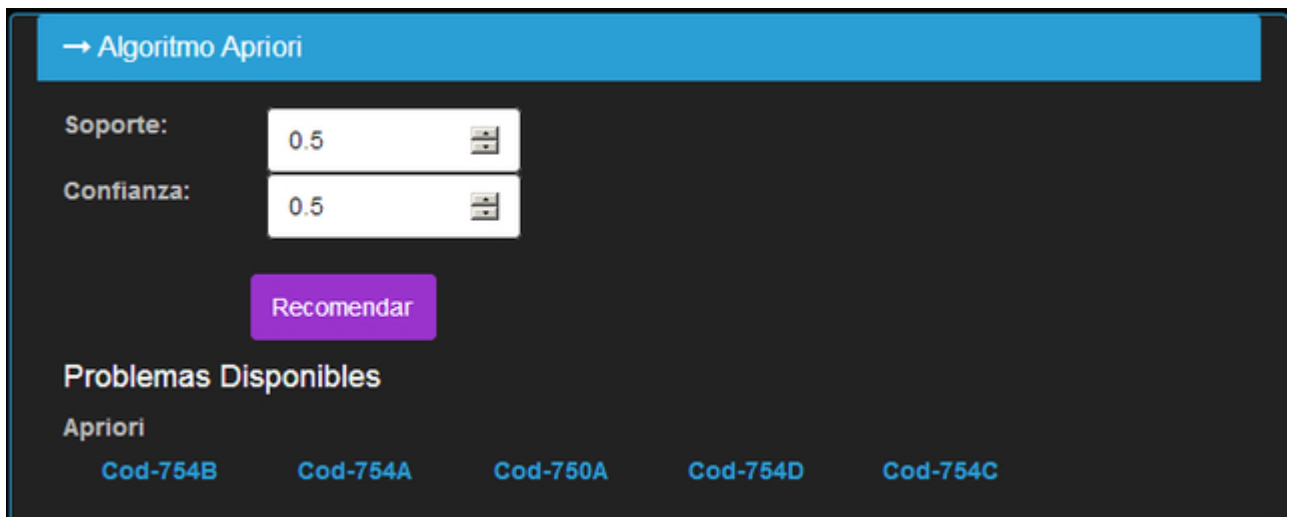


Figura 4.18: Recomendaciones usando el algoritmo apriori.

5

Implementación

Este capítulo tiene como objetivo explicar cómo se realizaron las distintas funciones de la aplicación, tanto la parte no visible (back-end) como la parte visible para el usuario (front-end).

5.1. Back-end

Esta sección tiene como objetivo mostrar cómo se realizaron las funciones de minería de datos, así como los recomendadores y otros algoritmos que funcionando en conjunto componen la parte no visible del proyecto. Esta parte de la aplicación fue realizada en su totalidad usando el lenguaje de programación Python y haciendo uso del framework Django.

5.1.1. Crawling

Para poder tener una amplia gama de problemas a la hora de recomendar o sugerir a los usuarios se realiza un proceso de web scraping haciendo uso de crawling para realizar este proceso de manera automática y así realizar la recopilación de problemas de diversas páginas como SPOJ, UVA, Codeforces, entre otras. Estos problemas son almacenados en la base de datos si no han sido almacenados ya, este proceso lo realiza el administrador una vez cada cierto tiempo para poder obtener nuevos problemas que han sido agregados a las páginas en cuestión.

Para poder hacer esto se crearon crawlers independientes para cada página, cada uno desarrollado de manera que se adapte al formato de cada página para luego ser estandarizados, es decir cambiados a la forma en la que estos problemas son almacenados en la base de datos. Este proceso lo denominamos Mapping. Este crawler, o recolector de datos, está hecho para obtener todos los problemas que se encuentran actualmente en los jueces online, a su vez obtienen todas las estadísticas de cada problema en conjunto con su categoría (si la posee), a la vez que obtiene los datos de los usuarios registrados en el sistema que posean cuentas en dichos sitios web, esto incluye principalmente los problemas que el usuario ha resuelto en las páginas. La manera en la que funcionan estos crawlers se explica a continuación.

Todos los crawlers están hechos con el lenguaje de programación Python utilizando el framework Scrapy destinado a facilitar los procesos de recolección.

5.1.1.1. Crawler SPOJ

- **Recolección de datos del usuario:** Dado el usuario, se utilizan expresiones XML para navegar por el documento HTML de la página spoj.com utilizando spi-

ders (hilos que trabajan sobre direcciones web específicas) , para facilitar esto se utilizó una extensión de chrome llamada "Xpath helper" [62] que mostraba la query aproximada que debía hacerse, con esto se obtienen los problemas resueltos por el usuario en forma de lista. A continuación se presenta un código representativo en lenguaje de programación Python que muestra como se extrajeron los problemas resueltos por el usuario en el juez Spoj:

```

class Spoj_user_spider(scrapy.Spider):
    name="spoj_user"
    allowed_domains = ["spoj.com"]
    def __init__(self, usuario=None,*args,**kwargs):
        super(Spoj_user_spider, self).__init__(*args,**kwargs)
        self.nombre_user=usuario
        self.start_urls = ['http://www.spoj.com/users/{0}'.
            format(usuario)]
    def parse(self, response):
        problemas = response.xpath('//table[@class="table_
            table-condensed"]/tr/td')
        Problemas_Resueltos=[]
        for p in problemas:
            ID_Problema = p.xpath('a/text()').extract()
                [0]
            Problemas_Resueltos.append(ID_Problema)

```

- **Recolección de datos de los problemas:** Utilizando expresiones XML se generan varias spiders que recorran y obtengan los problemas "Classical" de SPOJ, los cuales se encuentran distribuidos en varias secciones de su página web. Una vez encontrado un problema se procede a revisar su categoría nuevamente con una función XML ya que en la página la categoría aparece como una etiqueta, posteriormente se ingresa a los ranks del problema, donde se extrae

CAPÍTULO 5. IMPLEMENTACIÓN

una gran cantidad de información con el objetivo de utilizarla cómo base para asignar una dificultad al problema en el sistema. A continuación se presenta un código representativo en lenguaje de programación Python que muestra como se extrajeron datos de los problemas del juez Spoj:

```
class Spoj_problem_spider(scrapy.Spider):
    name="spoj_problem"
    allowed_domains=["spoj.com"]
    def __init__(self, *args, **kwargs):
        super(Spoj_problem_spider, self).__init__(*args, **
            kwargs)
        self.start_urls=[]
        Total = 0
        while(Total<=4000):
            link = "http://www.spoj.com/problems/
                classical/"
            if (Total!=0):
                link= link+"sort=0,start="+str(Total)
            Total= Total+50
            self.start_urls.append(link)
```

```
def parse (self ,response):
    problemas = response.xpath('//div[@class="table-
        responsive"]/table/tbody/tr/td[@align="left" ]')
    for x in problemas:
        url_temp= x.xpath('a/@href').extract()
        Descripcion=x.xpath('a/text()').extract()[0]
        ID_Problema=Extraer_ID(url_temp)
        link = "http://www.spoj.com/ranks/" +
            ID_Problema
        request=scrapy.Request(Link, callback=self.
            parse_problem)
```

```
Guardar_en_request(request ,ID_Problema ,
                    Descripcion)
yield request
```

```
def parse_problem(self , response):
    Submits = submits [1].xpath('text()').extract()[0]
    AC = submits [2].xpath('text()').extract()[0]
    URL = "http://www.spoj.com/problems/"+response.meta["
        ID_Problema"]
    req = scrapy.Request(Link2 , callback=self.parse_tags)
    Guardar_en_request(request , Submits , AC, URL)
    yield req

def parse_tags(self , response):
    tags = response.xpath('//div[@id="problem-tags"]/a')
    Dificultad=Calcular_Dificultad(AC, Submits)
    togs = 0
    Tags = []
    for tag in tags:
        T=extraer_tag(tag.select('@href'))
        Tags.append(T)
```

- Guardado de datos: Se guardan los datos crudos en documentos de texto, para luego ser llevados al formato requerido, y posteriormente almacenados en la base de datos.

5.1.1.2. Crawler UVA

- Recolección de datos del usuario: Utilizando el API de uhunt.com [63] se obtienen directamente los datos del usuario con una spider, entre ellos su id y los problemas que ha resuelto. A continuación se presenta un código representa-

tivo en lenguaje de programación Python que muestra como se extrajeron los problemas resueltos por el usuario en el juez UVA:

```

class UVA_user_spider(scrapy.Spider):
    name="uva_user"
    allowed_domains=["uhunt.felix-halim.net"]
    def __init__(self, usuario=None, Cantidad=10000,*args, **
        kwargs):
        super(UVA_user_spider, self).__init__(*args, **kwargs
        )
        r = requests.get('http://uhunt.felix-halim.net/api/
            uname2uid/{0}'.format(usuario))
        user_id = r.json()
        self.start_urls = ["http://uhunt.felix-halim.net/api/
            subs-user/{0}".format(user_id)]
    def parse (self, response):
        jsonresponse = json.loads(response.body.decode("utf
            -8","replace"))
        Problemas_Resueltos=[]
        for sub in jsonresponse["subs"]:
            p_id=sub[1]
            status=sub[2]
            if (status==90):
                ID_Problema=str(p_id)
                Problemas_Resueltos.append(
                    ID_Problema)

```

- Recolección de datos de los problemas: Utilizando el API de uhunt.com se obtienen directamente los datos de los problemas, sin embargo a través de esta API no es posible obtener las categorías de los problemas. A continuación se presenta un código representativo en lenguaje de programación Python que

muestra como se extrajeron los datos de los problemas en el juez UVA:

```
class UVA_problem_spider(scrapy.Spider):
    name="uva_problem"
    allowed_domains=["uhunt.felix-halim.net"]
    start_urls = ["http://uhunt.felix-halim.net/api/p"]
    def parse (self ,response):
        jsonresponse = json.loads(response.body.decode("utf
            -8", "replace"))
        for x in jsonresponse:
            ID_Problema=str(x[0])
            Descripcion=str(x[2])
            URL="https://uva.onlinejudge.org/index.php?
                option=com_onlinejudge&Itemid=8&page=
                show_problem&problem="+ID_Problema
            AC=x[18]
            ERR=0;
            for i in range(12,18):
                ERR=ERR+x[i]
            Total=AC+ERR
            Dificultad=Calcular_Dificultad(AC,Submits)
```

- Guardado de datos: Se guardan los datos crudos en documentos de texto, para luego ser llevados al formato requerido, y posteriormente almacenados en la base de datos.

5.1.1.3. Crawler COJ

- Recolección de datos del usuario: Se utiliza una consulta usando expresiones XML, para así obtener los datos del usuario directamente del documento HTML de la página coj.uci.cu con una spider, con esto se obtienen los proble-

mas resueltos por el usuario. A continuación se presenta un código representativo en lenguaje de programación Python que muestra como se extrajeron los problemas resueltos por el usuario en el juez COJ:

```

class Coj_user_spider(scrapy.Spider):
    name="coj_user"
    allowed_domains = ["coj.uci.cu"]
    def __init__(self, usuario=None, *args, **kwargs):
        super(Coj_user_spider, self).__init__(*args, **kwargs)
        self.start_urls = ['http://coj.uci.cu/user/
            useraccount.xhtml?username=%s' % usuario]
    def parse(self, response):
        problem_url="https://coj.uci.cu/24h/problem.xhtml?pid
            ="
        Resueltos=len(response.xpath('//div[@id="probsACC"]//
            a'))
        if tam<=0:
            return
        Problemas_Resueltos=[]
        for sel in response.xpath('//div[@id="probsACC"]//a'):
            ID_Problema = sel.xpath('span/text()').
                extract()[0]
            Problemas_Resueltos.append(ID_Problema)

```

- Recolección de datos de los problemas: Realizando otra consulta XML sobre otra dirección en el mismo dominio de COJ se obtienen todos los problemas con sus respectivas dificultades y categorías, este proceso se aplica varias veces en varias secciones de la página donde están almacenados los problemas. A continuación se presenta un código representativo en lenguaje de programa-

CAPÍTULO 5. IMPLEMENTACIÓN

ción Python que muestra como se extrajeron los datos de los problemas en el juez COJ:

```
class Coj_problem_spider(scrapy.Spider):
    name="coj_problem"
    allowed_domains = ["coj.uci.cu"]
    start_urls = ['https://coj.uci.cu/24h/problems.xhtml']
    def parse(self, response):
        problem_url="https://coj.uci.cu/tables/problems.xhtml
        ?"
        for sel in response.xpath('//select[@name="
        classification"]/option'):
            value=sel.xpath('@value').extract()[0]
            if (value=="-1"):
                continue
            Categoria=sel.xpath('text()').extract()[0].
            replace("_","-")
            link=problem_url+"classification="+value+"&
            complexity="
            for i in range(5):
                link2 = link+str(i+1)
                request=scrapy.Request(link2,
                callback=self.parse_problem)
                request.meta['categoria']=Categoria
                yield request
```

```
def parse_problem(self, response):
    tam=len(response.xpath('//table[@id="problem"]/tbody/
    tr[@class!="empty"]'))
    if tam==0:
        return
    for sel in response.xpath('//table[@id="problem"]/
```

```
tbody/tr '):
    ID_Problema = sel.xpath('td[@class="headidp
        "]/text()').extract()[0].strip()
    Descripcion = sel.xpath('td[@style="text-
        transform:none"]/a/text()').extract()[0]
    Submits= sel.xpath('td[3]/a/text()').extract
        ()[0]
    AC = sel.xpath('td[4]/a/text()').extract()
        [0]
    Categoria = response.meta['categoria']
    Dificultad = Calcular_Dificultad(AC,Submits)
    URL = "https://coj.uci.cu/24h/problem.xhtml?
        pid="+ID_Problema
```

- Guardado de datos: Se guardan los datos crudos en documentos de texto, para luego ser llevados al formato requerido, y posteriormente almacenados en la base de datos.

5.1.1.4. Crawler Codeforces

- Recolección de datos del usuario: Se utiliza el API de codeforces.com [64] para obtener los datos del usuarios con una spider, principalmente los problemas que ha resuelto. A continuación se presenta un código representativo en lenguaje de programación Python que muestra como se extrajeron los problemas resueltos por el usuario en el juez Codeforces:

```
class Codeforces_user_spider(scrapy.Spider):
    name="codeforces_user"
    allowed_domains = ["codeforces.com"]
    def __init__(self, usuario=None, Cantidad=10000,*args, **
        kwargs):
```

```

        super(Codeforces_user_spider, self).__init__(*args,
            **kwargs)
        self.start_urls = ["http://codeforces.com/api/user.
            status?handle={0}&from=1&count={1} ".format(
                usuario, str(Cantidad))]
    def parse(self, response):
        data = json.loads(response.body.decode("utf-8", "
            replace"))
        if data["status"] != "OK" or response.status != 200:
            return
        for p in data["result"]:
            Veredicto=p["verdict"]
            Problemas_Resueltos=[]
            if (Veredicto=="OK"):
                ID_Problema = str(p["problem"]["
                    contestId"])+str(p["problem"]["
                    index"])
                Problemas_Resueltos.append(
                    ID_Problema)

```

- Recolección de datos de los problemas: Utilizando el API de codeforces.com se obtienen todos los problemas de la página con su categoría, dificultad y estadísticas, cabe destacar que en Codeforces no es posible obtener los envíos totales de un problema, por tanto se utiliza la cantidad de problemas aceptados y la dificultad asignada en Codeforces (que es determinada por la letra asignada al problema, donde la "A" es el problema mas fácil hasta la "H" que suele ser el mas difícil). A continuación se presenta un código representativo en lenguaje de programación Python que muestra como se extrajeron los datos de los problemas en el juez Codeforces:


```

class Codeforces_problem_spider(scrapy.Spider):
    name="codeforces_problem"
    allowed_domains = ["codeforces.com"]
    contador=0
    start_urls=[
        "http://codeforces.com/api/problemset.problems"
    ]
    def parse(self, response):
        data = json.loads(response.body.decode("utf-8", "
            replace"))
        if data["status"]!="OK":
            return
        i=0
        for p in data["result"]["problems"]:
            ID_Problema = str(p["contestId"])+str(p["
                index"])
            Letra=str(p["index"]
            Descripcion_Problema =p["name"]
            Submits_AC=int(data["result"]["
                problemStatistics"][i]["solvedCount"])
            Dificultad=calcular_dificultad(AC, Letra)
            URL="http://codeforces.com/problemset/problem
                /{0}/{1}".format(str(p["contestId"]),
                Letra)
            Tags=[]
            for tag in p["tags"]:
                Tags.append(str(tag).replace("_", "-")
                    )
            i=i+1

```

- Guardado de datos: Se guardan los datos crudos en documentos de texto, para

luego ser llevados al formato requerido, y posteriormente almacenados en la base de datos.

5.1.2. Mapping

En este proceso se estandarizan los datos proporcionados por los crawlers, debido a que varias páginas nombran de manera diferente las categorías se decidió darles a cada una un nombre único para ser identificadas más fácilmente, en esta etapa se hace un listado que funciona de la siguiente manera:

- El primer string sin contar espacios en blanco de una línea es el nombre de la categoría a ser guardada en la base de datos.
- Cada string siguiente en la misma línea es otro nombre que se le da a la categoría, entonces si un problema posee esta categoría se le reemplaza con el nombre que se usará en la base de datos.

Posterior a este proceso, los datos deben ser almacenados en la base de datos para su futuro uso.

5.1.3. Clasificación

En este proceso se clasifican los problemas sin una categoría y además se les asigna a todos diversas dificultades, esto con el fin de tener una idea clara del tipo de problema y su dificultad para poder dar una recomendación más precisa al usuario dependiendo de su nivel en la categoría a recomendar. Para clasificar los problemas se utiliza la retroalimentación de los usuarios que ya han resuelto el problema, estos pueden votar hasta tres categorías a las que el problema puede pertenecer, el clasificador toma la categoría más votada como la categoría principal y si la segunda más

CAPÍTULO 5. IMPLEMENTACIÓN

votada es relativamente cercana a la primera se determina que el problema pertenece a ambas categorías.

Los problemas poseen tres dificultades, cada una suministrada por una fuente distinta:

- Dificultad votada por los usuarios: Es la dificultad promedio calculada en base a los votos de los usuarios que han resuelto el problema.
- Dificultad de la página: Es la dificultad calculada mediante la formula ó simplemente la dificultad ya asignada por la página en cuestión.

$$\text{ratio} = \frac{\text{Solucionesaceptadastotales}}{\text{Envíostotales}} \quad (5.1)$$

- Dificultad respecto al nivel: Es el promedio del nivel que tienen los usuarios que han resuelto el problema.

5.1.4. Calculo de habilidades

El nivel de habilidad de un usuario se calcula utilizando una media ponderada o promedio ponderado donde los problemas de mayor dificultad influyen más mientras que los de menor complejidad influyen menos, con esto se pretende saber el nivel aproximado del usuario en una o varias categorías para poder realizar recomendaciones en base a eso. Se utiliza la formula:

$$\bar{x} = \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i} = \frac{x_1 w_1 + x_2 w_2 + x_3 w_3 + \dots + x_n w_n}{w_1 + w_2 + w_3 + \dots + w_n} \quad (5.2)$$

5.1.5. Recomendaciones

Las recomendaciones vienen dadas por varios algoritmos que buscan según sus propios términos el problema no resuelto por el usuario con mayores probabilidades a ser un candidato que haga al usuario mejorar en la categoría a la que este problema pertenece.

En todos los casos se utilizó el lenguaje de programación Python, para los cálculos de distancias se utilizó SciPy y NumPy ambas librerías numéricas para el lenguaje Python utilizadas en minería de datos.

5.1.5.1. Recomendador utilizando el algoritmo apriori

El algoritmo apriori se basa en un conjunto de transacciones, en este caso la transacción (A,B) que se entiende como el usuario de nivel A resolvió el problema B, para posteriormente realizar un conjunto de pasos que nos permitan clasificar qué problemas están relacionados con otros. Primero se modeló una matriz de transacciones en la cual cada fila contiene todos los problemas resueltos por un mismo usuario y al aplicar el algoritmo este da como resultado aquellos problemas que están mas relacionados entre si debido a que muchos usuarios tienen en común estos conjuntos de problemas. Para este trabajo se utilizó la librería "apriori" [65] la cual es una de tantas posibles implementaciones del algoritmo apriori, funciona de la siguiente manera:

- Asignación de parámetros: el algoritmo pide que se le de un mínimo nivel de confianza para aceptar dos elementos en un grupo y un mínimo nivel de soporte para lo mismo. La confianza es qué tan precisa fue la clasificación, mientras el soporte es qué tan frecuentes son los elementos.

CAPÍTULO 5. IMPLEMENTACIÓN

- Procesamiento de las transacciones: una vez recibidas las transacciones el algoritmo procede a almacenarlas en un conjunto A y un conjunto B.
- Generación de candidatos: En este paso se recibe un tamaño máximo para los conjuntos finales que pueden ser demasiado grandes, posteriormente mientras existan candidatos se evalúa el soporte del candidato y si este es válido se añade al conjunto incrementando su tamaño en uno. Si el máximo tamaño de los conjuntos no se ha alcanzado se generan nuevos candidatos, estos nuevos candidatos son filtrados de manera que si todos los subconjuntos del candidato existen en la lista de candidatos previos este no sea agregado como nuevo candidato, de otra forma se agregan todas las combinaciones a la lista de candidatos.

A continuación se presenta un código representativo en lenguaje de programación Python que muestra el uso del algoritmo apyori para realizar las recomendaciones:

```
class Recomendador_Apriori:
    def main(Usuario , Soporte , Confianza , Max_iter):
        #Max_iter es el maximo numero de iteraciones permitidas
        Transacciones=self.Cargar_Transacciones()
        resultado = list(apriori(Transacciones , min_support=
            Soporte , min_confidence=Confianza))
        resultado.reverse()
        i=0
        for Regla in resultado:
            for Problema in Regla:
                if (No_Resuelto(Problema , Usuario))
                    Recomendaciones.append(Problema)
            if (i==Max_iter):
                break
```

CAPÍTULO 5. IMPLEMENTACIÓN

El proceso del algoritmo apriori es $O(N * 2^N)$ en tiempo gracias al paso de generación de conjuntos, para poder lidiar con esta complejidad tan alta se decidió siempre dar un límite pequeño al tamaño de los conjuntos con una confianza alta para asegurar que el procesamiento no tarde demasiado y siga teniendo buena probabilidad de dar una recomendación aceptable. Posterior a esto el algoritmo procede a elegir de manera aleatoria elementos del conjunto generado con el nivel del usuario hasta que obtenga un problema que el usuario no ha resuelto, gracias al alto nivel de confianza utilizado la recomendación debería estar cercana a lo que se requiere para que el usuario pueda mejorar.

5.1.5.2. Recomendador utilizando filtros colaborativos

Este recomendador utiliza un filtro colaborativo, es decir, sin tener conocimiento exacto de lo que se está recomendando, tener aún así una alta probabilidad de realizar una recomendación correcta, estos filtros funcionan de la siguiente manera:

- Lectura de datos: Se necesita un conjunto de elementos almacenados en la base de datos, entre ellos: el id del usuario, los problemas resueltos por los usuarios en forma de matriz, el nivel del mismo, la categoría a recomendar, el número de problemas a recomendar, y el número de usuarios entre los que comparar.
- Cálculo de distancia ó índice: Se hace un ciclo para comparar al usuario contra los demás del sistema, utilizando una formula de distancia, en este trabajo se utilizaron las siguientes distancias:
 - Euclidiana
 - Hamming
 - Similaridad del coseno

CAPÍTULO 5. IMPLEMENTACIÓN

- Disimilaridad de Bray-Curtis
- Ordenamiento de los resultados: Se ordenan los resultados dependiendo de lo que se necesite, en el caso de las distancias se ordenan de menor a mayor, mientras que en el caso de las similitudes de mayor a menor. Estos resultados reflejan el usuario más cercano al usuario actual.
- Selección de problemas: Se eligen problemas a recomendar, estos problemas deben cumplir que no fueron resueltos por el usuario pero si por otro usuario similar, a su vez el problema debe estar cercano al nivel del usuario que va a recibir la recomendación, cabe destacar que si la distancia o el índice da que los usuarios son demasiado similares no se tomará en cuenta dicho usuario y se pasará al siguiente.

El usuario final puede decidir qué problema resolver de los que fueron recomendados.

A continuación se presenta un código representativo en lenguaje de programación Python que muestra el enfoque del filtro colaborativo para realizar las recomendaciones:

```
class Recomendador_Colaborativo:
    def main(Usuario,K,Can,Opc):
        #K son el numero de vecinos
        #Can es la cantidad de problemas que se quiere recomendar
        #Opc es la opcion que determina que metrica de similitud
        usar
        Matriz ,id_usuario=Cargar_Matriz(Usuario) #Explicar esta
        matriz OJO borrar esto luego
        U_simil=[]
        i=0
        for User in Matriz:
```

```

        if (i==id_usuario):
            continue:
        else:
            U_simil.append((self.distancia(Opc, Matriz
                [id_usuario], User), i))
        i=i+1
    Usuarios_Similares= sorted(U_simil, key=lambda U_simil:
        U_simil[0])
    Usuarios_Similares=Usuarios_Similares[:K]
    for U in Usuarios_Similares:
        for P in Matriz[U[1]]:
            if (No_Resuelto(P, Usuario)):
                Recomendaciones.append(Pro) #
                guardamos todos los problemas
                que los K-vecinos hayan
                resuelto y el usuario no
    return Recomendaciones, Usuarios_Similares

```

5.1.5.3. Recomendador basado en contenido

Este recomendador utiliza las matrices de dificultades para generar una recomendación que a diferencia del filtro colaborativo, se basa directamente en los problemas y en qué tan bien están clasificados estos últimos. Este recomendador funciona de la siguiente manera:

- **Lectura de datos:** Se necesita un conjunto de elementos almacenados en la base de datos, entre ellos: el id del usuario, los problemas resueltos por el usuario, el nivel del mismo, la categoría a recomendar y el número de problemas a recomendar.

CAPÍTULO 5. IMPLEMENTACIÓN

- **Calculo del promedio:** Para cada problema se calcula el promedio entre la dificultad votada por los usuarios, la dificultad de la página, y la dificultad respecto al nivel, lo que se busca con esto es tener una idea del posible nivel real del problema.
- **Calculo de la desviación estándar:** Para poder tener una idea de cuan relativamente separadas están las dificultades se utiliza una formula de desviación estándar , para poder calcularla primero se obtiene el promedio entre las dificultades , posteriormente se calcula la desviación. Entre más bajo sea el valor resultado de esta formula significa que el problema está clasificado de mejor manera ya que las distintas partes coinciden, esta es una desviación relativa.

$$\text{avg} = \frac{A + B + C}{3} \quad (5.3)$$

$$\sqrt{\frac{\sum_{i=1}^3 (x_i - \text{avg})^2}{2}} \quad (5.4)$$

- **Ordenamiento de los resultados:** Se ordenan de menor a mayor desviación, de esta manera el primer problema es el problema con mejor clasificación y el último el peor.
- **Selección de problemas:** Para seleccionar los problemas se verifica que sean del mismo nivel que el usuario o un nivel por encima, y también que no haya sido resuelto por el usuario, se recomiendan tantos problemas como se solicite de una sola categoría.

A continuación se presenta un código representativo en lenguaje de programación Python que muestra el uso del enfoque basado en contenido para realizar las recomendaciones:

CAPÍTULO 5. IMPLEMENTACIÓN

```
class Recomendador_Basado_en_Contenido:
    def main(Usuario , Categoria , Can):
        #Can es la cantidad de problemas que se quiere recomendar
        Problemas=Extraer_Problemas_sin_Resolver(Usuario ,
            Categoria)
        Nivel=Nivel_por_Categoria(Usuario , Categoria)
        Mejores=[]
        for P in Problemas:
            A=P.Dificultad_estadisticas
            B=P.Dificultad_resuelto
            C=P.Dificultad_usuarios
            Promedio=(A+B+C)/3
            Desviacion = Desviacion_Estandar(A,B,C,
                Promedio)
            if (Nivel+1>=round(AVG) and Nivel<=round(
                AVG)):
                Mejores.append(Desviacion ,P) #
                    desviacion standard / promedio
                    es la similaridad relativa
        Mejores= sorted(Salida , key=lambda Mejores:
            Mejores[0])
        Mejores=Mejores[:Can]
        Recomendaciones=[]
        for M in Mejores:
            Recomendaciones.append(M[1])
    return Recomendaciones
```

5.2. Front-end

En esta sección se muestra como se realizaron todos los aspectos y funciones que son visibles al usuario.

5.2.1. Interfaz

La interfaz de la aplicación está dividida en tres secciones:

- Cabecera: Donde se muestra el nombre de la aplicación, y también donde se puede iniciar sesión.
- Menú: Ubicado a la izquierda, su utilidad yace en navegar por las distintas vistas de la aplicación.
- Contenido: Ubicado a la derecha y ocupando un espacio más grande que el menú, en ésta sección se encuentra la información y las funciones que provee la aplicación en esa vista.

Para la realización de la interfaz se hizo uso principalmente la librería Bootstrap que provee un sistema de cuadrículas que facilita la construcción de interfaces y su organización. Además también se usaron diferentes funciones HTML que provee HTML5 para lograr una interfaz lo mas fácil posible de usar.

También se hizo uso de la tecnología Javascript para realizar distintas validaciones y proporcionar mensajes de utilidad al usuario al momento de utilizar la aplicación

CAPÍTULO 5. IMPLEMENTACIÓN

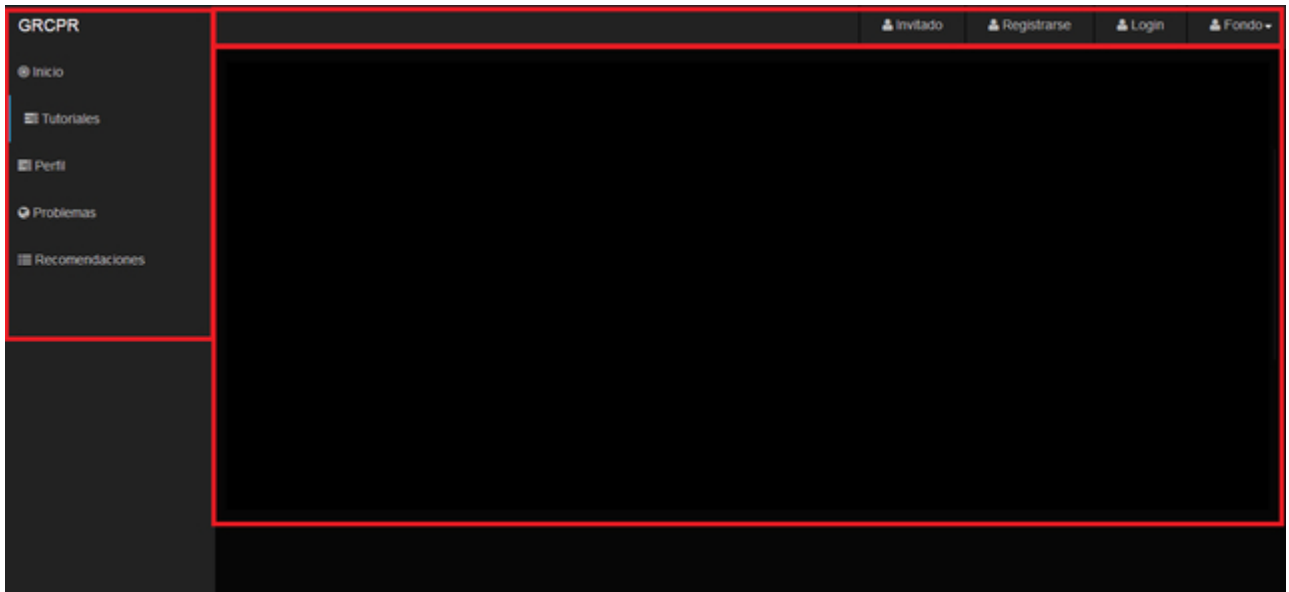


Figura 5.1: Secciones de la interfaz.

Además es posible cambiar el color de fondo de la aplicación a color blanco, para que sea del agrado de la mayoría de usuarios.

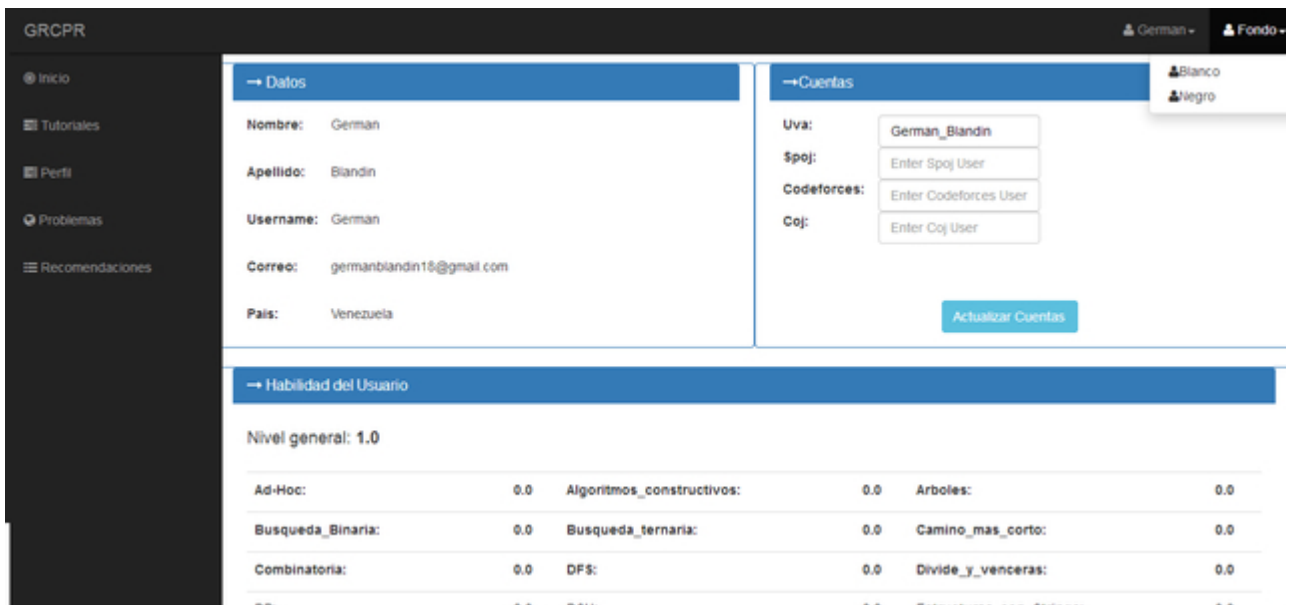


Figura 5.2: Cambio de fondo.

5.2.2. Log-in con Google y Facebook

La aplicación cuenta con la opción de registrarse y hacer Log-in utilizando cuentas externas de Google y Facebook. De estas cuentas se hace uso solo de el nombre de la cuenta del usuario y su correo electrónico. Esta opción fue implementada utilizando las tecnologías de Javascript y jQuery.

5.3. Seguridad

En el aspecto de la seguridad se utilizó lo siguiente:

- Login/Password: Una vez registrada la cuenta el usuario deberá introducir su nombre de usuario y contraseña para poder ingresar al sitio, la contraseña se

CAPÍTULO 5. IMPLEMENTACIÓN

encuentra oculta mientras el usuario la escribe.

- **MySQL Security:** La seguridad estándar de la base de datos fue la elegida ya que por el momento la página es solo para pruebas de funcionamiento.
- **Seguridad proveniente de Django:** Django como framework posee distintas funcionalidades que proveen a la aplicación una capa mas de seguridad, por ejemplo una de ellas es protección para evitar ataques de inyección SQL.

6

Pruebas y resultados

En este capítulo se busca mostrar cómo se comportó la aplicación con respecto a su eficiencia en tiempo dado un conjunto de usuarios artificiales y su efectividad, la cual fue comprobada mediante encuestas a usuarios orgánicos que probaron la aplicación.

6.1. Pruebas eficiencia en tiempo

Para estas pruebas se utilizaron dos sets de datos los cuales consisten en usuarios participantes en ciertas competencias del juez online Codeforces, estos usuarios fueron tomados utilizando un crawler en conjunto con la API de Codeforces.

6.1.1. Prueba uno

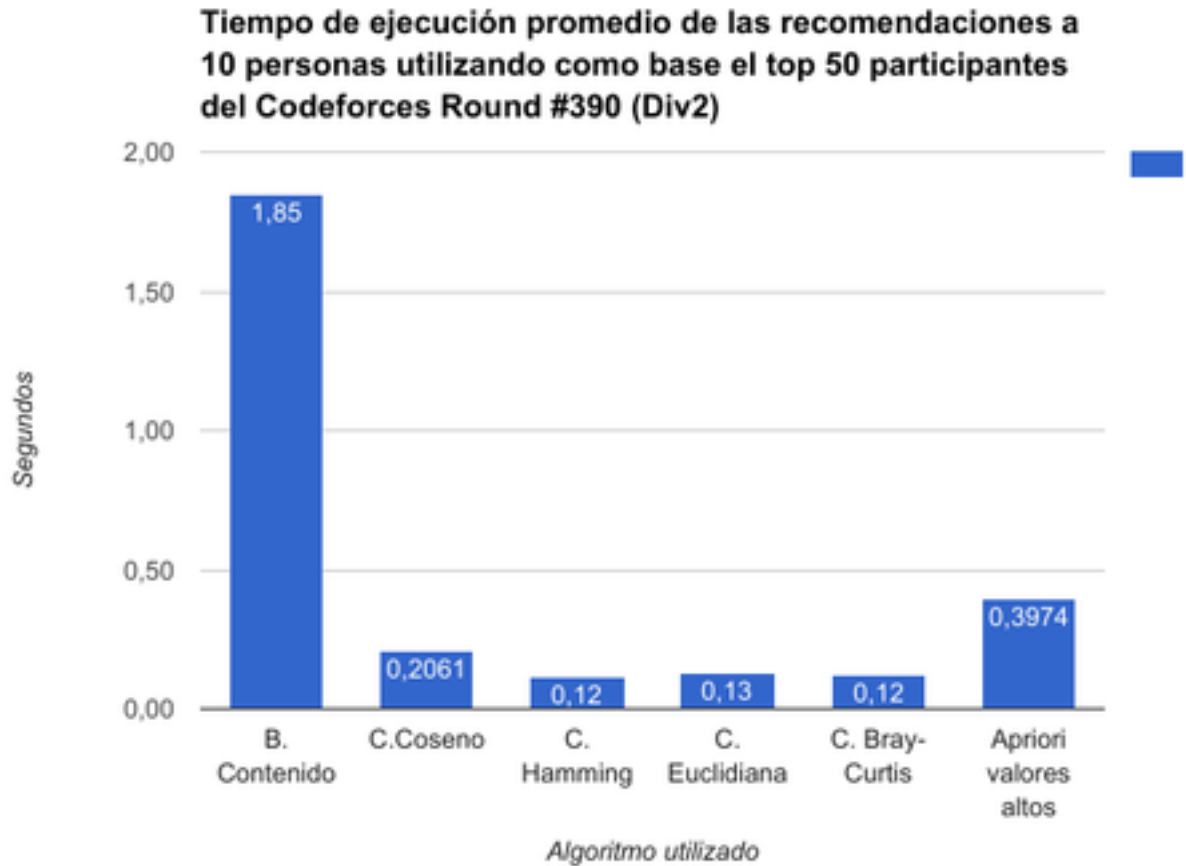


Figura 6.1: Prueba 50 personas.

La primera prueba (figura 6.1) se realizó con cincuenta usuarios del contest "Codeforces Round 390 (Div2)", se tomaron diez de estos para hacer pruebas de rendimiento promedio, como se puede apreciar los algoritmos colaborativos los cuales dependen del número de usuarios se comportan de manera eficiente respecto al colaborativo que a pesar de ser lento en comparación sigue siendo bastante bueno, el algoritmo apriori logró tener un buen rendimiento en esta prueba luego de colocar el soporte y la confianza por encima del valor "0,50", colocar cualquier valor por debajo

CAPÍTULO 6. PRUEBAS Y RESULTADOS

de este resultaba en un tiempo de ejecución superior a los diez minutos debido a la naturaleza exponencial del algoritmo.

6.1.2. Prueba dos

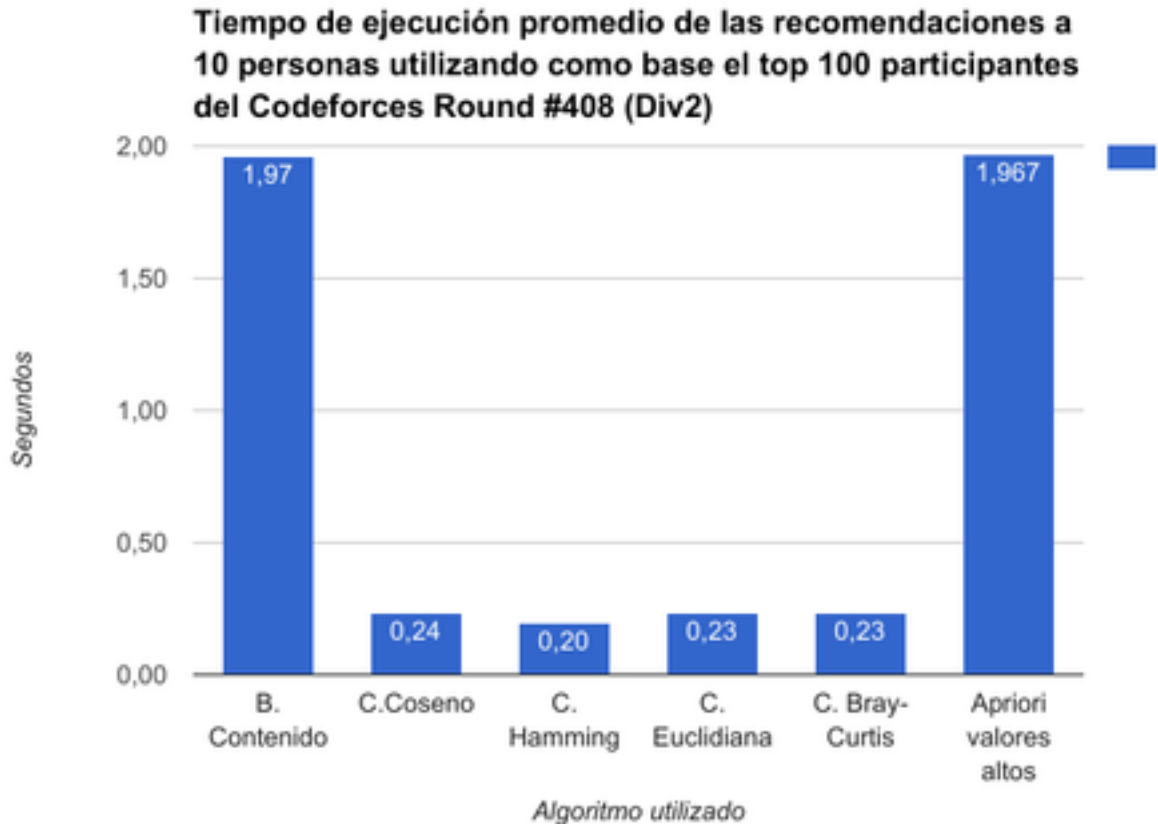


Figura 6.2: prueba 100 personas.

La segunda prueba (figura 6.2) se realizó con cien usuarios del contest "Codeforces Round 408 (Div2)", se tomaron diez de estos para hacer pruebas de rendimiento promedio, los algoritmos colaborativos nuevamente se comportaron de manera eficiente aunque más lentos que antes, el algoritmo basado en contenido se mantuvo

en casi el mismo tiempo promedio de la anterior prueba, mientras que el apriori tuvo un cambio brusco en el tiempo pasando de menos de un segundo a más de dos segundos en su tiempo de ejecución promedio.

6.1.3. Conclusiones

Se puede concluir con estas pruebas lo siguiente:

- Respecto al algoritmo basado en contenido: A pesar del incremento de usuarios al doble, el algoritmo mantiene un tiempo similar en ambas pruebas ya que no depende de la cantidad de usuarios, sino de la cantidad de problemas guardados que por lo general es constante durante el día.
- Respecto a los algoritmos colaborativos: Fueron los que mejor desempeño mostraron con respecto al tiempo de ejecución, se vio un incremento leve entre la prueba con cincuenta usuarios y la prueba con cien, se puede concluir que el algoritmo va a ser más lento entre más usuarios registrados existan en la página, no obstante su desempeño seguirá siendo bueno hasta que se alcancen cotas altas de usuarios.
- Respecto al algoritmo apriori: El brusco incremento en tiempo de ejecución hace ver que el algoritmo apriori es inviable para un volumen grande de usuarios para resolver este problema, no obstante puede utilizarse para resolver otro tipo de problemas.

6.2. Pruebas de aceptación

Se realizaron un total de doce preguntas a veinticinco usuarios que utilizaron la aplicación, todos ellos con alguna noción sobre la programación competitiva y todos ellos con conocimientos en programación convencional.

¿Le parece que la aplicación posee un diseño entendible y/o usable?
(24 respuestas)

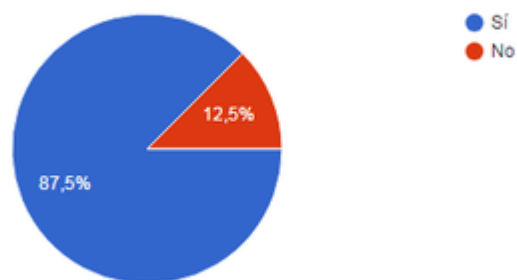


Figura 6.3: Pregunta 1

Como podemos observar en la gráfica , la mayoría de los usuarios considera que la aplicación posee un diseño entendible y usable.

CAPÍTULO 6. PRUEBAS Y RESULTADOS

¿Le parece que la aplicación posee una paleta de colores adecuada?
(24 respuestas)

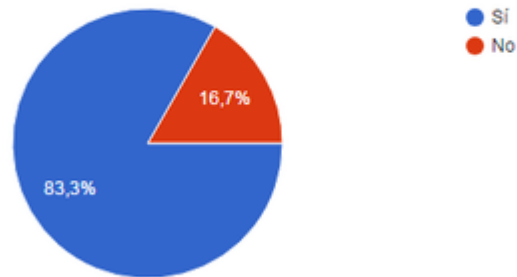


Figura 6.4: Pregunta 2

En esta pregunta la mayoría de usuarios respondió positivamente a la paleta de colores utilizada originalmente (negro), se acordó colocar una opción para poder utilizar un fondo diferente si el usuario desea (blanco).

¿Le parece que la sección de tutoriales es necesaria? (24 respuestas)

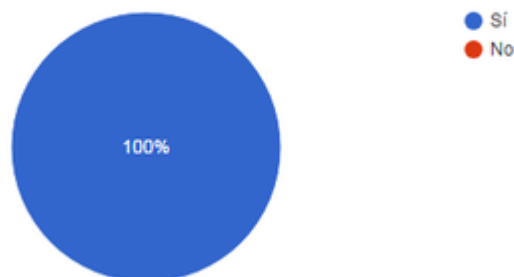


Figura 6.5: Pregunta 3

Como era de esperarse de una aplicación que busca enseñar, la respuesta unánime

CAPÍTULO 6. PRUEBAS Y RESULTADOS

fue que la aplicación requiere de una sección de tutoriales para ayudar en el aprendizaje de nuevas técnicas.

¿Le parece que mostrar el nivel del usuario por categoría es un buen método para ver su progreso en el tiempo?
(24 respuestas)

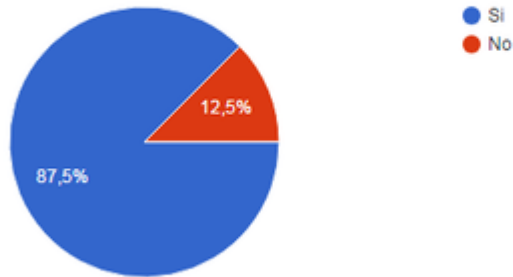


Figura 6.6: Pregunta 4

La gran mayoría de usuarios consideró que ver cómo su nivel mejora es un reflejo de qué tanto ha mejorado él en el tiempo.

¿Le parece que tener sus problemas resueltos en la página facilita su búsqueda?
(24 respuestas)

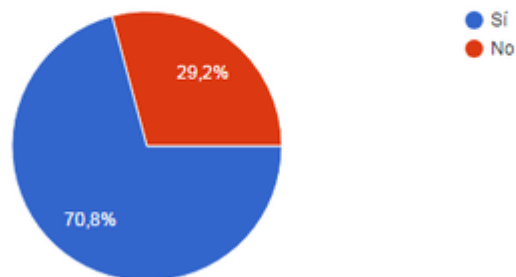


Figura 6.7: Pregunta 5

CAPÍTULO 6. PRUEBAS Y RESULTADOS

Los problemas resueltos de cada usuario se encuentran en la página, un porcentaje alto de personas consideraron esto como innecesario, pero la mayoría sigue considerándolo como algo importante para la aplicación.

¿Le parece que la comunidad sea la que clasifique los problemas en categorías ayuda a que estas clasificaciones sean precisas?
(24 respuestas)

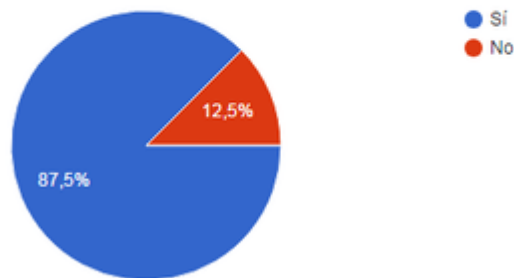


Figura 6.8: Pregunta 6

La gran mayoría de usuarios están de acuerdo con que los problemas sean clasificados por la comunidad da una clasificación más precisa que ayuda a que las recomendaciones sean mejores.

CAPÍTULO 6. PRUEBAS Y RESULTADOS

¿Se tomaría el tiempo en clasificar los problemas que ha resuelto?
(24 respuestas)

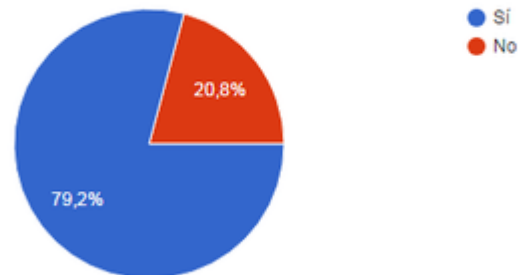


Figura 6.9: Pregunta 7

La mayoría de los usuarios estarían dispuestos a clasificar problemas que ha resuelto, no obstante un cuarto de los usuarios no estaría dispuesto a hacerlo.

¿Le parece que un sistema de recomendación es una buena manera de encontrar problemas que lo ayuden a mejorar en programación competitiva?
(24 respuestas)



Figura 6.10: Pregunta 8

Todos los usuarios están de acuerdo en que un sistema de recomendación es una

CAPÍTULO 6. PRUEBAS Y RESULTADOS

muy buena forma para mejorar en programación competitiva.

Teniendo como referente el recomendador basado en contenido seleccione las opciones que aplican

(24 respuestas)

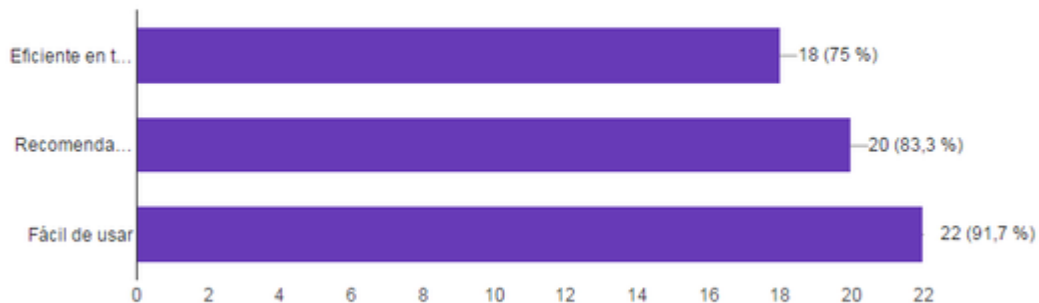


Figura 6.11: Pregunta 9

El recomendador basado en contenido probó ser eficiente en tiempo para la mayoría de usuario así como confiable a la hora de dar recomendaciones y fácil de usar.

Teniendo como referente el recomendador usando el algoritmo apriori seleccione las opciones que aplican

(24 respuestas)

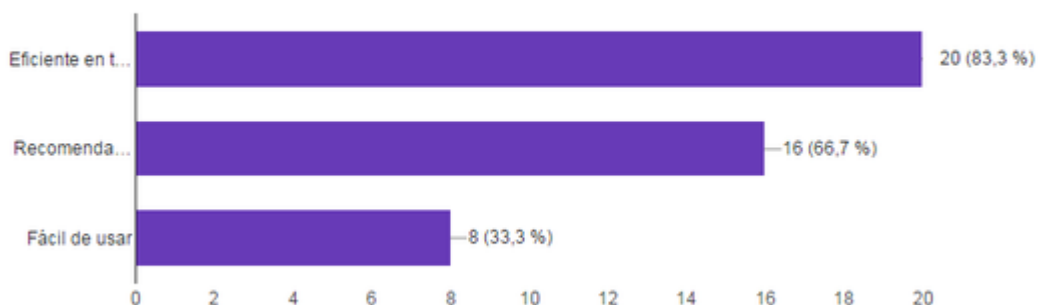


Figura 6.12: Pregunta 10

CAPÍTULO 6. PRUEBAS Y RESULTADOS

El algoritmo apriori fue eficiente en tiempo y dio recomendaciones precisas a la mayoría de usuarios, pero muchos usuarios encontraron dificultades al usarlo.

Teniendo como referente el recomendador usando filtros colaborativos seleccione las opciones que aplican

(24 respuestas)

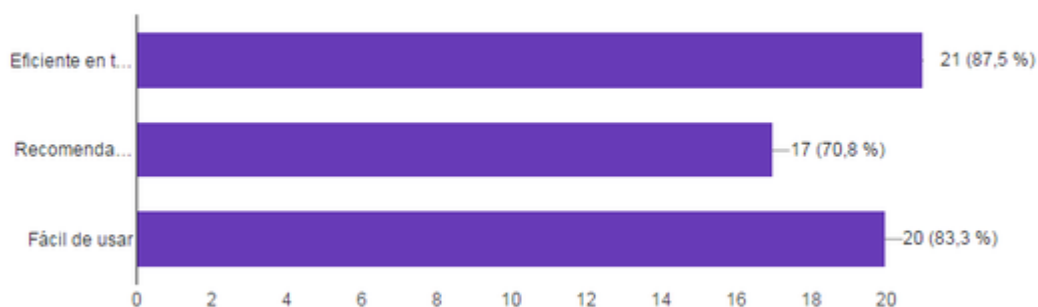


Figura 6.13: Pregunta 11

El recomendador usando filtros colaborativos probó ser eficiente en tiempo, preciso y fácil de usar para la gran mayoría de usuarios.

¿A su parecer cual o cuales fueron los mejores filtros colaborativos?

(24 respuestas)



Figura 6.14: Pregunta 12

CAPÍTULO 6. PRUEBAS Y RESULTADOS

Los filtros colaborativos con mejor desempeño según los usuarios fueron la "similaridad de cosenos" y la "distancia euclidiana", no obstante todos los algoritmos tuvieron un desempeño similar.

6.2.1. Conclusiones

Se puede concluir que la aplicación en su estado actual fue bien aceptada por los usuarios, no obstante existen ciertos puntos sujetos a mejoras. Muchos usuarios dieron diversas sugerencias que serán tomadas para mejorar la aplicación en el futuro.

7

Conclusiones y trabajos a futuro

7.1. Conclusiones

Se logró utilizar el método para Web Scraping: Crawling para la extracción de información de los distintos sitios web. Extrayendo los datos de los problemas como su ID, descripción, URL, estadísticas de envío y categorías de los problemas en caso de que las poseyera, esta información sirve como base del sistema, para que el usuario pueda obtener recomendaciones y medir su nivel de habilidad. Luego de esto se logró implementar un sistema de recomendación basado en contenido centrado en las características y necesidades del usuario en las técnicas y algoritmos que conforman la programación competitiva, a su vez se implemento un sistema de recomendación haciendo uso de filtros colaborativos para mantener la competitividad entre usuarios que posean un nivel similar o que hayan resuelto problemas parecidos, ambos sistemas presentaron buenos resultados ya que cumplen el objetivo de ir centrados en lo que el usuario necesita para su desarrollo como competidor.

CAPÍTULO 7. CONCLUSIONES Y TRABAJOS A FUTURO

Para la correcta implementación de estos sistemas de recomendación también se asignó dificultad a los problemas usando las estadísticas de envíos y problemas aceptados, además se dio la opción para que los usuarios puedan votar para modificarla. También es posible que los usuarios asignen las categorías a los problemas que ya hayan resuelto con anterioridad. Estas acciones permiten que el sistema mejore a largo plazo debido a que los problemas tendrán sus características mas definidas.

Como complemento del aprendizaje y desarrollo del competidor se creó la sección de tutoriales, preparada para agregar diversas guías y tutoriales de diversas técnicas que permitan el desarrollo de las habilidades del competidor.

Además se realizaron distintas pruebas, que, como se pudo ver anteriormente, muestran que la aplicación cumple con su propósito y fue aceptada por los usuarios, certifican que tanto un sistema de recomendación de problemas, como una sección de tutoriales de programación competitiva es necesario para un mejor desarrollo de las habilidades y técnicas necesarias para competir. También se pudo comprobar el rendimiento de los distintos métodos usados para las recomendaciones, obteniendo un buen desempeño en cada uno de ellos teniendo como base un máximo de 100 usuarios, sin embargo tanto el sistema basado en contenido, como el de filtros colaborativos podrían mantener su buen desempeño aun con mas usuarios.

Para lograr que la aplicación llegue al mayor publico posible, la aplicación posee compatibilidad multi-plataforma debido a la utilización de tecnologías compatibles con los navegadores mas populares en la actualidad y la interfaz de la aplicación es usable y aceptada por la mayoría de usuarios.

Para finalizar, como ya se ha mencionado, las competencias de programación son eventos importantes en el mundo de las ciencias de la computación, que muestran un crecimiento cada vez más rápido en términos de cantidad de participantes y di-

CAPÍTULO 7. CONCLUSIONES Y TRABAJOS A FUTURO

ficultad en los problemas, muchas universidades cuentan con entrenadores de alto nivel lo cual les da una clara ventaja por sobre otros participantes. Solucionar esto no es sencillo, no obstante con el trabajo presentado se muestra el inicio de un sistema que pueda ayudar a personas en general a mejorar en programación competitiva sin depender enteramente de algún profesor ó entrenador que lleve su progreso o le diga qué problemas resolver.

Presumiblemente el trabajo presentado mejorará el rendimiento de muchísimos participantes y abrirá las puertas a nuevos competidores que buscan aprender de una manera rápida y efectiva a resolver problemas, no obstante la aplicación también puede ayudar a personas que buscan simplemente aprender a resolver problemas ó a programar a un nivel mayor. Sin embargo, a pesar de tener muy buenas impresiones en cuanto al desempeño de la aplicación durante las pruebas de rendimiento y de aceptación, es necesario que la aplicación continúe con un proceso de maduración para ganar la confianza de mas usuarios y que vean en la aplicación una excelente oportunidad para entrenar y ser mejores competidores en lo que respecta a este tipo de eventos.

7.2. Trabajos a futuro

Con respecto a los trabajos a futuro se podrían determinar algunas nuevas formas de determinar la dificultad de los problemas en base a sus estadísticas.

Se podría establecer una manera de categorizar automáticamente los problemas sin necesidad de que el usuario tenga que clasificarlos.

Con respecto a la interfaz de usuario, encontrar mejores maneras de presentar las funciones que provee la aplicación de manera de que sea mas agradable de usar y abarque un publico mucho mas amplio.

Debido a la naturaleza del algoritmo apriori y los resultados que proporciona, este se podría utilizar para mostrar los problemas mas comunes que han sido resueltos por los usuarios en periodos de tiempo determinados, como por ejemplo los problemas que mas han sido resueltos en la ultima semana y el ultimo mes.

Bibliografía

- [1] S. Halim y F. Halim, *Competitive Programming 3*, 3ra. ed. Steven Halim and Felix Halim, Mayo 2013.
- [2] F. R. Cepeda y D. E. Ruiz, "Programación competitiva," *www.holamundo.mx*, 2014, [Accessed 25-January-2016]. [En línea]. Disponible en: <http://www.holamundo.mx/programacion-competitiva/>
- [3] ACM-ICPC, "The problems," <https://icpc.baylor.edu/compete/problems>, 2015, [Online; accessed 25-January-2016].
- [4] E. Tirado y R. Tovar, "Construcción de un juez para competencias de programación," Master's thesis, Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, 2014.
- [5] L. Joyanes y I. Zahonero, *Estructura de datos en C++*, 1ra. ed. S.A. MCGRAW-HILL, 2007.
- [6] R. Sedgewick y K. Wayne, *Algorithms*, 4th ed. Addison-Wesley Professional., 2011.
- [7] R. A. Finkel y J. L. Bentley, "Quad trees a data structure for retrieval on

BIBLIOGRAFÍA

- composite keys,” *Acta Inf.*, vol. 4, núm. 1, pp. 1–9, Mar. 1974. [En línea]. Disponible en: <http://dx.doi.org/10.1007/BF00288933>
- [8] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Commun. ACM*, vol. 18, núm. 9, pp. 509–517, Sep. 1975. [En línea]. Disponible en: <http://doi.acm.org/10.1145/361002.361007>
- [9] A. V. Aho *et al.*, *Compiladores principios, técnicas y herramientas*, 2da. ed. Pearson Education, 2008.
- [10] ACM-ICPC, “About acm-icpc,” <https://icpc.baylor.edu/>, 2015, [Online; accessed 25-January-2016].
- [11] A. ICPC, “World finals rules,” <https://icpc.baylor.edu/worldfinals/rules>, 2015, [Online; accessed 25-January-2016].
- [12] A. ICPC, “Regional rules,” <https://icpc.baylor.edu/regionals/rules>, 2015, [Online; accessed 27-January-2016].
- [13] IOI, “About ioi,” http://wiki.ioinformatics.org/wiki/About_IOI, 2011, [Online; accessed 28-January-2016].
- [14] IOI, “Ioi rules,” <http://ioinformatics.org/rules/reg10.pdf>, 2010, [Online; accessed 28-January-2016].
- [15] TopCoder, “Top coder srm,” <https://www.topcoder.com/member-onboarding/competing-in-an-algorithm-match-srm/>, 2015, [Online; accessed 01-February-2016].
- [16] Codeforces, “Codeforces help,” <http://codeforces.com/help>, 2011, [Online; accessed 28-January-2016].
- [17] Codeforces, “Codeforces contest,” <http://codeforces.com/blog/entry/456>, 2010, [Online; accessed 28-January-2016].

BIBLIOGRAFÍA

- [18] Google, "Google code jam," <https://code.google.com/codejam>, 2016, [Online; accessed 30-January-2016].
- [19] Google, "Google code jam, term and conditions," <https://code.google.com/codejam/terms.html>, 2016, [Online; accessed 30-January-2016].
- [20] Facebook, "Facebook hacker cup, page info," https://www.facebook.com/hackercup/info/?tab=page_info, 2011, [Online; accessed 30-January-2016].
- [21] Facebook, "Facebook hacker cup, term and conditions," <https://www.facebook.com/hackercup/terms>, 2016, [Online; accessed 30-January-2016].
- [22] A.-I. L. Archive, "Verdict information," https://icpcarchive.ecs.baylor.edu/index.php?option=com_content&task=view&id=14&Itemid=30, 2016, [Online; accessed 26-January-2016].
- [23] C. P. DE CAMPOS y C. E. FERREIRA, "Boca online contest administrator," <http://www.ime.usp.br/~cassio/boca/>, 2004, [Online; accessed 03-February-2016].
- [24] J. Eldering y other, "Domjudge - introduction," <https://www.domjudge.org/intro>, 2015, [Online; accessed 03-February-2016].
- [25] M. Ángel Revilla-Universidad de Valladolid, "Uva online judge - home," <https://uva.onlinejudge.org>, 1995, [Online; accessed 03-February-2016].
- [26] Spoj.com, "About the spoj system," <http://www.spoj.com/info/>, [Online; accessed 03-February-2016].
- [27] I. H. Witten, E. Frank, y M. A. Hall, *Data Mining Practical Machine Learning Tools and Techniques*, 3ra. ed. Morgan Kaufmann, 2011.
- [28] L. Kaufman y P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

BIBLIOGRAFÍA

- [29] R. J. Bayardo, Jr., "Efficiently mining long patterns from databases," *SIGMOD Rec.*, vol. 27, núm. 2, pp. 85–93, Jun. 1998. [En línea]. Disponible en: <http://doi.acm.org/10.1145/276305.276313>
- [30] F. Ricci, L. Rokach, y B. Shapira, "Introduction to recommender systems," <http://www.inf.unibz.it>, 2011, [Accessed 25-February-2016]. [En línea]. Disponible en: <http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf>
- [31] R. Moya, "¿que son los sistemas de recomendación?" <http://jarroba.com>, 2013, [Accessed 25-February-2016]. [En línea]. Disponible en: <http://jarroba.com/que-son-los-sistemas-de-recomendacion/>
- [32] E. Herrera-Viedma, C. Porcel, y L. Hidalgo, "Sistemas de recomendaciones: herramientas para el filtrado de información en internet," <https://www.upf.edu>, 2004, [Accessed 25-February-2016]. [En línea]. Disponible en: <https://www.upf.edu/hipertextnet/numero-2/recomendacion.html>
- [33] P. Resnick y H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, núm. 3, pp. 56–58, Mar. 1997. [En línea]. Disponible en: <http://doi.acm.org/10.1145/245108.245121>
- [34] R. R. Yager, "Fuzzy logic methods in recommender systems," *Fuzzy Sets Syst.*, vol. 136, núm. 2, pp. 133–149, Jun. 2003. [En línea]. Disponible en: [http://dx.doi.org/10.1016/S0165-0114\(02\)00223-3](http://dx.doi.org/10.1016/S0165-0114(02)00223-3)
- [35] J. D. Ullman, *Mining of Massive Datasets*, 2da. ed. Cambridge University Press, 2014.
- [36] T. S. community, "'scipy'," <https://docs.scipy.org/doc/scipy/reference/spatial.distance.html>, [Online; accessed 29-Abril-2017].
- [37] Wikipedia, "Desviación estándar," https://en.wikipedia.org/wiki/Standard_

BIBLIOGRAFÍA

- deviation, [Online; accessed 29-Abril-2017].
- [38] S. L. Mora, *Programacion en Internet*, 1ra. ed. Editorial Club Universitario, 2001.
- [39] W. Español, ""modelo vista controlador"," <https://es.wikipedia.org/wiki/Modelo-vista-controlador>, [Online; accessed 19-Abril-2016].
- [40] O. C. and/or its affiliates, "Mysql," <https://www.mysql.com/>, [Online; accessed 19-Abril-2016].
- [41] W. Español, "Servidor web," https://es.wikipedia.org/wiki/Servidor_web, [Online; accessed 19-Abril-2016].
- [42] T. A. S. Foundation., "Apache," http://httpd.apache.org/ABOUT_APACHE.html, [Online; accessed 19-Abril-2016].
- [43] W. Español", "Servidor http apache," https://es.wikipedia.org/wiki/Servidor_HTTP_Apache, [Online; accessed 19-Abril-2016].
- [44] P. S. Foundation, "Python," <https://www.python.org/>, [Online; accessed 19-Abril-2016].
- [45] W. Español, "Javascript," <https://es.wikipedia.org/wiki/JavaScript>, [Online; accessed 19-Abril-2016].
- [46] W. Español, "Html," <https://es.wikipedia.org/wiki/HTML>, [Online; accessed 19-Abril-2016].
- [47] w3schools, "Css," <http://www.w3schools.com/css/>, [Online; accessed 19-Abril-2016].
- [48] D. S. Foundation, "Django," <https://www.djangoproject.com/L>, [Online; accessed 19-Abril-2016].
- [49] M. Otto *et al.*, "Bootstrap," <http://getbootstrap.com/>, [Online; accessed 19-

BIBLIOGRAFÍA

Abril-2016].

[50] Scrapinghub, "'scrapy'," <http://scrapy.org/>, [Online; accessed 19-Abril-2016].

[51] W. Ingles, "'web scraping'," https://en.wikipedia.org/wiki/Web_scraping, [Online; accessed 19-Abril-2016].

[52] T. jQuery Foundation, "jquery," <https://jquery.com/>, [Online; accessed 19-Abril-2016].

[53] N. developers, "'numpy'," <http://www.numpy.org/>, [Online; accessed 19-Abril-2016].

[54] S. developers, "'scipy'," <https://www.scipy.org/>, [Online; accessed 19-Abril-2016].

[55] TopCoder, "Competitive programming at topcoder," <https://www.topcoder.com/community/competitive%20programming/>, [Online; accessed 16-February-2016].

[56] Spoj.com, "'spoj pagina principal'," <http://www.spoj.com>, [Online; accessed 20-February-2016].

[57] A. Aly, "A2 online judge," <http://www.ahmed-aly.com/>, [Online; accessed 20-February-2016].

[58] Usaco, "Usa computing olimpiad," <http://www.usaco.org/>, [Online; accessed 18-February-2016].

[59] Uhunt, "uhunt pagina principal," <http://www.uhunt.com>, [Online; accessed 20-February-2016].

[60] COJ, "Coj pagina principal," <http://coj.uci.cu/>, [Online; accessed 20-February-2016].

BIBLIOGRAFÍA

- [61] Codeforces, "codeforces pagina principal," <http://codeforces.com>, [Online; accessed 20-February-2016].
- [62] A. Sadosky, "Xpath helper," <https://chrome.google.com/webstore/detail/xpath-helper/hgimnogjllphhkhlmbebbmlgjoejdpl>, [Online; accessed 21-Mayo-2017].
- [63] F. Halim, "Api uhunt," <http://uhunt.felix-halim.net/api>, [Online; accessed 21-Mayo-2017].
- [64] Codeforces.com, "Api codeforces," <http://codeforces.com/api/help>, [Online; accessed 21-Mayo-2017].
- [65] Y. Mochizuki, "apryori," <https://pypi.python.org/pypi/apryori/1.0.0>, [Online; accessed 29-Abril-2017].
- [66] A. S. Arefin, *Art of Programming Contest*, 2da. ed. Gyankosh Prokashoni, 2006.
- [67] H. Navarro, *Notas de Docencia Técnicas Avanzadas de Programación*, Agosto 2011. [En línea]. Disponible en: <http://ccg.ciens.ucv.ve/~hector/tap/TAP.pdf>
- [68] D. Cournapeau, "'scikit learn.," <http://scikit-learn.org/>, [Online; accessed 19-Abril-2016].
- [69] W. McKinney, "'pandas'," <http://pandas.pydata.org/>, [Online; accessed 19-Abril-2016].
- [70] N. Project, "'nltk'," <http://www.nltk.org/>, [Online; accessed 19-Abril-2016].
- [71] D. Albanese, R. Visintainer, S. Merler, S. Riccadonna, G. Jurman, y C. Furlanello, "mlpy: Machine learning python," <http://mlpy.sourceforge.net/>, 2012, [Online; accessed 19-Abril-2016].