Explicaciones visuales para la gestión y la recomendación en jueces en línea

Trabajo de Fin de Grado
Curso 2018–2019

Autor
Ederson Aldair Funes Castillo
José Luis Gómez Alonso

Director
Guillermo Jiménez Díaz
Pedro Pablo Gómez Martín

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Explicaciones visuales para la gestión y la recomendación en jueces en línea

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Ederson Aldair Funes Castillo
José Luis Gómez Alonso

Director
Guillermo Jiménez Díaz
Pedro Pablo Gómez Martín

Convocatoria: *Septiembre 2019*
Calificación: *Nota*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

20 de septiembre de 2019

Dedicatoria

Dedico este trabajo a Alexia, Lili, Karmezina, Alisson, Nila y Mark por convertirme en el hombre que soy hoy. A mi familia y amigos, por haber confiado en mí hasta el final. Y a ti, el lector.

Ederson Funes Castillo

Dedico este trabajo a mi amigo y compañero Luis, que estuvo en los momentos más difíciles brindando siempre su apoyo y que hoy no esta aquí para verlo.

José Luis Gómez Alonso

Agradecimientos

Queremos mostrar nuestros agradecimientos a nuestros tutores Guillermo y Pedro por su paciencia, su cercana ayuda y su profesional dedicación hacia nosotros durante todo el proyecto. Muchas gracias por darnos la oportunidad de trabajar sobre algo real y útil como lo es *¡Acepta el reto!*.

Dar las gracias a la UCM y más concretamente a la FDI por permitirnos experimentar una gran y valiosa etapa en nuestra vida donde hemos crecido como personas, nos hemos formado como profesionales, hemos descubierto un gran mundo de hobbies y curiosidades y he hecho amistad con amigos y compañeros que nos llevamos fuera de este ciclo, un abrazo para todos ellos, sin vosotros no hubiera sido posible.

Y por último dar las gracias a las personas de nuestro entorno, amigos y familia, por de una manera u otra, contribuir a que nosotros podamos haber experimentado esta etapa.

Resumen

¡Acepta el reto! es un almacén y juez en línea de problemas de programación en español que acepta soluciones en C, C++ y Java. Esta herramienta crea un entorno académico en el que los usuarios pueden poner a prueba sus conocimientos de programación sobre las distintas temáticas que se ofrecen, y recibir feedback de las soluciones que proporcionen. Los problemas almacenados en *¡Acepta el reto!* están organizados según categorías y volúmenes, pero esta división puede no ser suficiente para conectar con las necesidades del usuario a la hora de decidir el siguiente problema al cual enfrentarse.

Este trabajo surge ante la necesidad de introducir un sistema de recomendación de problemas para los usuarios de jueces en línea para facilitar la selección de problemas que los usuarios consideren relevantes o asequibles. Esto servirá para amenizar la experiencia con los jueces en línea y disminuir la frustración de los usuarios a la hora de intentar resolver problemas que sobrepasan o subestiman su nivel de habilidad. El proyecto diferencia tres partes claras:

- Planteamiento de un sistema de puntuación para aproximar el nivel de habilidad de los usuarios y el nivel de dificultad de los problemas disponibles en un juez en línea.
- Diseño y desarrollo de un sistema de recomendación capaz de recomendar problemas a usuarios basándose en su habilidad y dificultad de los problemas.
- Implementación de un prototipo en el que se pueda comprender y gestionar mediante explicaciones visuales el comportamiento del sistema de puntuación y recomendación.

Palabras clave

Sistemas de recomendación, Jueces en línea, Aplicación web, Resolución de Problemas, Elo, *¡Acepta el reto!*

Abstract

¡Acepta el reto! is an online automated judge for programming problems in Spanish that accepts solutions in C, C ++ and Java. This site creates an academic environment in which users can test their programming skills under the different kinds of problems offered, and receive feedback for the solutions that they provide. These problems are divided into multiple categories, but this division may not be enough to satisfy the user's needs when choosing the next problem.

This work arises from the need to introduce a recommendation system capable of offering problems to each user that uses online judge programs, so that in this way, he can choose to solve problems that are included in his skill level and knowledge. This will make using online judges a more enjoyable experience and will reduce users' frustration when trying to solve problems that exceed or underestimate the level of the user. The project differentiates three clear parts:

- Build a ranking system capable of measuring skill level and difficulty for both users and problems from any online judge.
- Development of a recommender system capable of recommending problems to users based on the user's skill level and the problem difficulty.
- Implementation of a prototype in which the behavior of both the rating system and recommender system can be understood and managed through visual explanations.

Keywords

Recommender systems, Online Judges, Web Application, Problem resolution, Elo, Acepta el reto

Índice

Índice de figuras

Índice de tablas

Chapter 1

Introduction

1.1. Motivation

Recommendation systems are invisible, discrete and opaque elements, that most people don't know how they work, but all people have used them sometime in his life. With the rise of the Internet and processing capacity, the amount and variety of data that a user has access to is almost endless. Proof of this information overload (Reaney, 2012) are the new disciplines created specifically to process and analyze large amounts of data, such as Big Data and Data Mining. With millions of options just a click away, users must now spend more time looking for whatever item meets their needs.

Although it is true that some small e-commerce sites are capable of making custom recommendations to their small number of users, those sites that have a much larger amount of both users and items need to find new ways to process and analyze all their data without sacrificing too much resources.

Recommendation systems emerged to help solve the problem of information overload. These systems are used in most sites to process and filter large amounts of data in order to obtain relevant and customized information for their users.

Sadly, there is no recommendation system that fits all e-commerce sites' needs, so most of the time it must be implemented from scratch or adapted from a different older version.

Even though these recommender systems have been used mostly in e-commerce sites to recommend products to users, they can be used for different purposes. A good example would be those apps that try to teach you

a new language, since most of them try to measure your proficiency in said language before recommending you a range of exercises.

This project will be focused on designing a recommender system capable of recommending coding problems to users of an online judge. These recommendations must take into account a user's skill and a problem's difficulty, which will be measured using a rating system. Once finished, these recommender systems will be included in an "app" which will let an admin monitor and manage both the rating system and recommendation system.

This "app" is nothing more than a testbed that will be used to test the multiple rating systems (and recommender system) developed in this project.

1.2. Background

¡Acepta el reto![1] is an online judge whose authors made available for everyone and offers multiple coding exercises aimed for students of the Computer Science degree from the UCM. It has more than 400 problems, divided into multiple categories. The solution to most of these coding exercises have to run with a limited amount of memory and under a time limit.

Even though this online judge doesn't have an exorbitant amount of problems, it suffers from information overload. This, added to the fact that the only way for a user to know how hard a problem can be to solve are the percentages of users who managed to solve it or the percentages of correct solutions submitted, makes it hard for new users to choose new problems.

One of the papers that talked about using recommender systems to solve the problem of information overload was *"Using Recommendation System for E-learning Environments at degree level"* (Martínez et al., 2015) in which they concluded: *"Our investigation shows the problem of the information overload is also present in distance educational environments. The obtained results show most of the users are not willing or can't do all of the practises the system puts at their disposal, that is why they would find help useful in order to decide which practises they should do."*

The article "*Similarity Metrics from Social Network Analysis for Content Recommender Systems*" (Jiménez Díaz et al., 2016), written by two teachers from the Facultad de Informática (UCM) and published in the International Conference on Case-Based Reasoning (ICCBR) of 2016, tried to solve this problem. The article suggests *"the use of similarity-based link*

[1] https://www.aceptaelreto.com/

prediction techniques over the implicit user-problem interaction network to predict new links that may be seen as problem recommendation to users". Marta Caro Martinez used this article to work on her TFM *"Sistemas de Recomendación basado en técnicas de predicción de enlaces para jueces en línea"* (Caro Martínez, 2017), in which she added new ways of generating recommendations using user-problem interaction graphs.

In this project we pursue the same goal as the ones before us, which is to develop a recommender system for *¡Acepta el reto!*, but using a different approach.

1.3. Objetives

The objectives pursued in this project are:

- Build a rating system capable of measuring skill level and difficulty for both users and problems from any online judge.
- Development of a recommender system capable of recommending problems to users based on the user's skill level and the problem difficulty.
- Design of a testbed in which the behavior of both the rating system and recommender system can be monitored and managed in a visual way.

The steps to follow to achieve them are:

1. Study of recommendation systems, online judges, rating systems and statistical methods.
2. Planning, design and development of a rating system capable of measuring a user's skill level and the difficulty of a problem.
3. Design and build a recommender system adapted to online judges.
4. Development, testing and analysis of the outcome obtained after using our recommender system on data from an online judge (*¡Acepta el reto!*).

1.4. Document structure

The contents of this project have been divided into seven chapters:

1. **Chapter 2. State of the art**. In this chapter we study what is a recommender system and the different approaches to create one. We

also study the structure of online judges, which ones are the most popular, and how the users choose the problems they want to solve.

2. **Chapter 3. Adaptation of the ELO rating system for recommendation in online judges**. In this chapter, we adapt the ELO rating system so it can be used in online judges. We also explain how the problems will be recommended to users.

3. **Chapter 4. Testing and Analysis**. In this chapter we talk about how we modified and filtered data from *¡Acepta el reto!*'s DB in order to use it in our tests. We also talk about which rating system measures users' skills / problem difficulty better, and how accurate our recommender system is.

4. **Chapter 5. Web App Development**. This chapter describes the development of our testbed where admins can observe, modify and interact with both the recommender and rating system. It also talks about the multiple tools used to develop it.

5. **Chapter 6. Conclusions and Future Work**. In this chapter, we focus on the outcome of this project and the work that can be done in the future.

6. **Chapter 7. Personal Contributions**. In this chapter, each member mentions their contribution to this project.

Capítulo 1

Introducción

"Nunca dejes que tu sentido de la moralidad te impida hacer lo que está bien"
— Los límites de la Fundación, Isaac Asimov (1982)

1.1. Motivación

Los sistemas de recomendación son elementos invisibles, discretos y opacos que la mayoría de personas usan de manera inconsciente sin conocer los engranajes que hacen a este tipo de sistemas funcionar. Con el auge de Internet y la capacidad de procesamiento, la cantidad y variedad de los datos a los que un usuario puede tener acceso son casi infinitos. Prueba de esta sobrecarga de información son las nuevas disciplinas creadas específicamente para procesar y analizar estas grandes cantidades de datos, como son el Big Data y la minería de datos (Reaney, 2012). Con millones de opciones al alcance de un clic, los usuarios tienen que gastar más tiempo para encontrar aquello que satisface sus necesidades.

Aunque es verdad que existen tiendas online capaces de realizar recomendaciones personalizadas a grupos de clientes reducidos, aquellos que experimentan un aumento tanto del número de clientes potenciales como el de productos disponibles se ven obligados a hacer uso de sistemas capaces de captar y procesar grandes cantidades de datos de forma automática y eficaz. Ante este problema de saturación surgieron los sistemas de recomendación automatizados, encargados de filtrar y priorizar la información considerada como adecuada para sus usuarios.

No existe un sistema de recomendación "universal" que se pueda aplicar a todos los entornos, sino que en muchas ocasiones este debe ser creado desde cero, dado que la naturaleza de los datos a tratar y las técnicas que se

quieren utilizar no siempre son las mismas.

A pesar de que estos sistemas nacieron de la necesidad de las tiendas online por satisfacer a sus clientes, su uso no está limitado a estos entornos. Un ejemplo de esto son las aplicaciones que buscan enseñar nuevos idiomas, las cuales intentan determinar el nivel de conocimiento del lenguaje de un usuario, y a partir de ello recomiendan un punto de partida desde el cual empezar, o ejercicios cuya finalidad es la de reforzar conceptos que el usuario no domina como debería.

En este trabajo, realizaremos una investigación que se centrará en diseñar un sistema de recomendación de problemas para usuarios adaptado a entornos de jueces en línea que se aproxime lo máximo al nivel de habilidad, exigencia e intereses de los usuarios basándonos en la inclusión de un sistema de puntuación para medir la habilidad de los usuarios. Este sistema de recomendación se incluirá en una aplicación web en la que se pueda comprender y gestionar mediante explicaciones visuales tanto el sistema de puntuación como el sistema de recomendación.

La aplicación no es un dashboard que pueda ser utilizado por usuarios noveles, sino que servirá como banco de pruebas, el cual se usará para probar los diferentes sistemas de puntuación y recomendación y analizar el comportamiento de éstos.

1.2. Antecedentes

¡Acepta el reto![1] se presenta como un juez en línea en el que los autores ponen a disposición de los usuarios problemas en español planteados para los alumnos de las diferentes asignaturas de programación de los Grados de Informática y Ciclos Formativos. Dispone de una batería de más de 400 problemas, que pueden recorrerse por categorías, lo que simplifica encontrar aquellos que resulten más adecuados en cada momento. Muchos de los problemas están pensados para forzar soluciones con complejidades específicas no solo en tiempo, sino también en espacio, cosa que es poco habitual en otros jueces (Gómez-Martín y Gómez-Martín, 2017).

Es una plataforma que encaja perfectamente con el modelo de e-learning, pero que presenta el problema de la sobrecarga de información al poseer en su repositorio más de 400 problemas de programación. Esto supone un inconveniente al usuario (sobre todo a primerizos) a la hora de elegir cuál es el problema que más se ajusta a su nivel de conocimiento. En *¡Acepta el reto!*,

[1] https://www.aceptaelreto.com/

los únicos sistemas de guía existentes son la categorización de sus problemas según el concepto de programación necesario para resolverlos, el porcentaje de envíos aceptados con respecto al número de envíos realizados, y el porcentaje de envíos aceptados con respecto al número de usuarios que lo han intentado. Estos dos últimos se aplican tanto a problemas individuales como a categorías. Estas estadísticas pueden no ser buenos indicadores de la verdadera dificultad de los problemas.

Uno de los artículos que hablaba de los sistemas de recomendación como solución al problema de sobrecarga de información fue *"Using Recommendation System for E-learning Environments at degree level"* (Martínez et al., 2015), el cual concluyó: *"Nuestra investigación muestra que el problema de la sobrecarga de información también está presente en entornos educativos a distancia. Los resultados obtenidos muestran que la mayoría de los usuarios no están dispuestos o no puede hacer todas las prácticas que el sistema pone a su disposición, es por eso que encontrarían la ayuda útil para decidir qué prácticas deberían hacer"*

A este problema se le intentó poner solución en el trabajo realizado en 2016 por profesores de la Facultad de Informática (UCM) y miembros del grupo de investigación GAIA en el artículo "*Similarity Metrics from Social Network Analysis for Content Recommender Systems*", del congreso International Conference on Case-Based Reasoning (ICCBR) de 2016 (Jiménez Díaz et al., 2016). En él, se propone un método de recomendación para jueces en línea a partir de un grafo de interacción de problemas contenidos en el juez para guiar estos usuarios. A este trabajo le sucede el TFM de Marta Caro Martinez "*Sistemas de Recomendación basado en técnicas de predicción de enlaces para jueces en línea*" (Caro Martínez, 2017) en el que reproducía y ampliaba la investigación de Guillermo Jimenez Diaz incluyendo nuevas propuestas de recomendadores basados en el grafo de interacción de los usuarios de la plataforma, así como la inclusión de nuevos métodos que cambian la forma de recomendar problemas a los usuarios.

Nuestro trabajo persigue el mismo objetivo propuesto en trabajos anteriores en lo referido a establecer un sistema de recomendación de problemas en *¡Acepta el reto!*, pero tomando caminos distintos, planteando una solución basada en otras técnicas de recomendación, y explorando el abanico de posibilidades con el cual se puede abordar este problema.

1.3. Objetivos

Los objetivos que se persiguen en este trabajo son:

- Planteamiento de un sistema de puntuación para aproximar el nivel

de habilidad de los usuarios y el nivel de dificultad de los problemas disponibles en un juez en línea.

- Diseño y desarrollo de un sistema de recomendación capaz de recomendar problemas a usuarios basándose en su habilidad y dificultad de los problemas.
- Diseño de un banco de pruebas en el que se pueda observar y gestionar de manera visual el comportamiento del sistema de puntuación y recomendación.

Estos objetivos se alcanzarán mediante:

1. Investigación sobre sistemas de recomendación, jueces en línea, sistemas de puntuación y métodos estadísticos.
2. Planteamiento, diseño y desarrollo de un sistemas de puntuación que permita medir la habilidad de los usuarios y la dificultad de los problemas.
3. Diseño de un sistema de recomendación adaptado a las necesidades, condiciones y utilidades de los jueces en línea.
4. Desarrollo, experimentación y análisis de los resultados obtenidos, usando un conjunto de datos extraídos de *¡Acepta el reto!*, para confirmar que los sistemas utilizados son correctos.

1.4. Organización de la memoria

La secuencia de capítulos que conforman esta memoria es la siguiente:

- **Capitulo 2. Estado del arte**. En este capítulo investigamos qué son los sistemas de recomendación y cuáles son las técnicas más utilizadas a la hora de implementarlos. También nos introducimos en qué son los jueces en línea y cómo funcionan, además de qué problemas presentan a la hora de seleccionar qué ejercicios le interesan más al usuario resolver en la gran colección de la que disponen estos jueces. Realizamos un breve estudio de cuáles son los más utilizados y qué características tienen. Estudiamos también los sistemas de puntuación que utilizan alguno de los jueces en línea.
- **Capitulo 3. Adaptación de ELO para recomendación en jueces en línea**. En este capítulo adaptamos y personalizamos el sistema de puntuación ELO, de modo que pueda usarse con usuarios y problemas de la mayoría de jueces en línea, y explicamos distintos tipos de recomendaciones de problemas.

- **Capitulo 4. Análisis y Evaluación**. Este capítulo incluye los pasos que seguimos para realizar las pruebas y evaluaciones necesarias para conocer la precisión de nuestros sistemas de recomendación. Establecemos un filtrado de la base de datos para escoger una muestra lo más representativa y útil como para sacar resultados claros. Además, comentamos las modificaciones que hemos realizado en la base de datos para ajustarla al sistema de recomendación creado y realizamos un estudio de cuál de los sistemas de ELO creados refleja el verdadero nivel de habilidad de los usuarios y la dificultad de los problemas.

- **Capitulo 5. Desarrollo de Aplicación Web**. Este capítulo describe la arquitectura y desarrollo de una Aplicación Web desde donde un usuario podrá observar, de manera visual, la información del sistema de puntuación y los datos relacionados con el sistema de recomendación y poder interactuar de modo que pueda alterar valores, añadir unos nuevos o simular un envió de manera intuitiva. Se explica la parte funcional de la aplicación y qué herramientas se han usado para su construcción.

- **Capitulo 6. Conclusiones y Trabajo Futuro**. Este capítulo resume las principales aportaciones realizadas y algunas de las posibles líneas de trabajo futuro.

- **Capitulo 7. Trabajo Personal**. En este capítulo mencionamos las contribuciones realizadas por ambos miembros.

Resaltar antes de comenzar, que todo el código fuente que es usado en la plenitud del desarrollo de la memoria y por extensión del proyecto en si, ha sido colgado a un repositorio de GitHub y puede ser consultado a través de este enlace[2].

[2]https://github.com/jlsomeg/TFG-ELORecommender

Capítulo 2

Estado del Arte

"Sin el continuo crecimiento y progreso, la palabras como mejora, logro y éxito no tienen ningún significado"
— Benjamin Franklin

Como mencionamos en la introducción, uno de los objetivos de este proyecto es el desarrollo de un sistema de puntuación capaz de determinar la habilidad de los usuarios y la dificultad de los problemas que nos permitirá crear un recomendador para los jueces en línea. En este capítulo haremos un estudio del estado del arte de estos tres elementos principales: jueces en línea, sistemas de recomendación y sistemas de puntuación.

2.1. Jueces en linea

En términos generales, los jueces en línea son sistemas web que contienen una gran colección de problemas y desafíos en sus repositorios, generalmente de programación, que aceptan envíos de código fuente. Este código es compilado, ejecutado y su salida es capturada y comparada con diferentes casos de prueba predefinidos para cada problema, con el objetivo de que los estudiantes practiquen con estos ejercicios y mejoren sus habilidades. Por cada solución enviada, el juez devuelve un veredicto, anunciando si la solución propuesta es correcta o no. También pueden generar respuestas sobre restricciones propias en cuanto a límites en memoria, tiempo de ejecución o errores y excepciones surgidos durante la compilación o ejecución del código. Estos sistemas se suelen usar para el aprendizaje de la programación, aunque su origen son los concursos de programación.

A la estela del funcionamiento de los concursos de programación como el ACM/ICPC *(International Collegiate Programming Contest)*, los problemas propuestos en los jueces en línea suelen exigir que los programas lean de la

entrada estándar un conjunto de casos de prueba, y que para cada caso generen, por la salida estándar, un resultado. La validación de una determinada solución se realiza a través de análisis dinámico, de modo que el código es compilado y ejecutado en el servidor y, utilizando las capacidades de redirección del sistema operativo, se le proporciona por la entrada una batería de casos de prueba creados de antemano. La salida obtenida a partir del código suministrado es comparada con la esperada, creada con la solución oficial implementada por los autores del problema y, dependiendo del resultado de la comparación, se proporciona un veredicto u otro. El abanico de posibles veredictos varía con cada juez, pero los más usados son:

- Aceptado (AC): la salida dada por el programa encaja exactamente con la esperada.
- Error de presentación (PE): la salida es correcta salvo por los separadores. Ocurre, por ejemplo, si la salida contiene espacios o saltos de línea sobrantes.
- Respuesta incorrecta (WA): la salida no encaja con la esperada.
- Error de ejecución (RTE): el programa falló durante la ejecución y no llegó a terminar. Es el veredicto habitual cuando un programa en C/C++ realiza una operación inválida, o uno en Java genera una excepción no controlada.
- Límite de tiempo excedido (TLE): el programa estaba tardando demasiado tiempo en ejecutarse, y fue detenido antes de que pudiera terminar.
- Límite de memoria excedido (MLE): el programa solicitó demasiada memoria, y fue detenido antes de terminar.
- Error de compilación (CE): el código enviado no se ha podido compilar.

Dado que se tratan de sistemas web, los usuarios que los utilizan no están sujetos a horarios y pueden escoger dónde y cuándo enfrentarse a los problemas propuestos. Esto los hace especialmente útiles en procesos de aprendizaje, además de ser fácilmente adaptables a metodologías universitarias. Al ser los envíos corregidos automáticamente, aporta imparcialidad y una mayor agilidad a la hora de practicar y mejorar habilidades de estructuras de datos, algoritmos y programación.

2.1.1. Estudio de jueces en linea actuales

Existe un gran número de jueces en línea disponibles. De ellos nos interesa conocer los sistemas o métodos que utilizan para guiar a sus nuevos

usuarios y, si dispone de un sistema de recomendación, averiguar que tipo de datos utilizan para recomendar problemas a los usuarios.

Es importante comentar que muchos de los jueces investigados no son totalmente trasparentes acerca de cómo funciona su sistema. Incluso en algunos se desconoce qué tipo de sistema usan para sus recomendaciones.

2.1.1.1. Kattis

Kattis[1] es un juez en línea que, además de evaluar soluciones para un repertorio de problemas, permite a los usuarios organizar concursos de programación en los que diversos equipos compiten para resolver una selección de problemas.

Los usuarios registrados cuentan con dos tipos de puntuaciones: una puntuación visible (*score*) generada a partir de la suma de la dificultad de los problemas que llevan resueltos, y una puntuación invisible (*rating*), calculada mediante una variante del sistema de puntuación ELO, la cual no se encuentra especificada en la pagina web. El *rating* de un usuario se incrementará conforme vaya resolviendo problemas, y tendrá en cuenta la calidad los envíos realizados.

Por otra parte, la asignación de la dificultad de los problemas se puede resumir de la siguiente manera:

- Se asignan valores de dificultad bajos a aquellos problemas que son resueltos por un alto número de usuarios en pocos envíos.
- Los problemas que son intentados frecuentemente pero que raramente se resuelven obtienen un valor de dificultad alto.
- Se asignan valores de dificultad media a aquellos problemas que no cuentan con un número de envíos suficiente para ser clasificados.

Con estos datos, Kattis es capaz de recomendar una serie de problemas con un nivel de dificultad que considera adecuados para un usuario con un determinado *rating*. Estos problemas se dividen en 4 grupos: triviales, fáciles, medios y difíciles.

2.1.1.2. Timus

Timus[2] es un juez en línea que solo se limita a juzgar las soluciones enviadas por sus usuarios. Aunque no cuenta con un recomendador de problemas,

[1] https://open.kattis.com/
[2] http://acm.timus.ru/

los problemas y usuarios tienen asociado un valor numérico que indica la dificultad del problema y la puntuación del usuario. Según se indica en su página, el cálculo de la dificultad de los problemas tiene en cuenta el número de usuarios que han resuelto el problema y la antigüedad del mismo: "cuantos menos usuarios nuevos hayan resuelto el problema por unidad de tiempo, mayor será la dificultad del problema."

Mientras tanto, la puntuación del usuario se calcula sumando la dificultad de los problemas que ha resuelto. No cuenta con un sistema de recomendación, pero dispone de una categoría de problemas para usuarios principiantes.

2.1.1.3. Project Euler

Project Euler[3] es un repositorio de problemas en el que los usuarios no tienen que subir su código, sino solo las respuestas a los preguntas planteadas, las cuales suelen tener una fuerte componente matemática. Cada problema tiene asociado un valor de dificultad, pero no explican el método utilizado para generar este valor, por lo que hemos asumido que es asignado manualmente tras la publicación del mismo. Tampoco tiene sistema para estimar el nivel de habilidad del usuario. Este sitio no es un juez, puesto que solo se dedica a exponer enunciados de problemas y pedir una respuesta correcta, pero sirve para observar la estructura de asignación de dificultades.

Tampoco cuenta con un recomendador, pero los usuarios pueden ordenar los problemas según su nivel de dificultad y el número de veces que ha sido resuelto.

2.1.1.4. CodeChef

En CodeChef[4], la dificultad de los problemas es asignada manualmente mediante etiquetas. Se utiliza para puntuar participantes en competiciones individuales que la pagina suele presentar y utiliza una modificación del sistema de puntuación ELO. Cada vez que se participa en un concurso, el usuario compite en un duelo con todos los demás jugadores participantes. Según su calificación actual y la calificación de los otros jugadores participantes, se calcula su probabilidad de ganar contra cada uno de ellos. Su rendimiento real se compara con este rendimiento esperado y, en consecuencia, gana o pierde algunos puntos.

[3] https://projecteuler.net/
[4] https://www.codechef.com/

2.1.1.5. UVa Online Judge

UVa Online Judge[5], desarrollado en la Universidad de Valladolid en 1995 por Ciriaco García de Celis, es uno de los jueces en línea más antiguos y conocidos. En él tambien se organizan concursos de programación en los que cualquier usuario puede participar. UVa alberga más de 4900 problemas, los cuales están divididos en volúmenes o sets.

A pesar de su antigüedad, los únicos datos que sirven para asumir la dificultad de los problemas son los porcentajes de envíos correctos frente al total de envíos y porcentaje de usuarios que lo han resuelto frente al total que lo ha intentado. No cuenta con un sistema de recomendación de problemas ni con un sistema que mida la habilidad de los usuarios.

2.1.1.6. ¡Acepta el reto!

¡Acepta el reto![6] es un juez en línea creado por dos profesores de la Universidad Complutense de Madrid en 2014 (Gómez-Martín y Gómez-Martín, 2017). Es utilizado por los alumnos de grado de la Facultad de Informática y ciclos formativos como herramienta de aprendizaje en sus asignaturas y también como medio de celebración de concursos de programación. Acepta soluciones programadas utilizando C, C++ y Java y para cada envío realizado informa del tiempo y de la memoria total consumida.

Contiene más de 400 problemas de programación en español, creados pensando en los alumnos de las diferentes asignaturas de los grados de Informática de la UCM y ciclos formativos. El nivel de dificultad de los problemas es amplio, existiendo problemas diseñados específicamente para las primeras semanas de los cursos de introducción a la programación donde únicamente se conocen variables y expresiones, hasta problemas complejos que requieren algoritmos sobre grafos o conocimiento de teoría de números.

Los problemas están categorizados de acuerdo a diferentes ejes, varios directamente relacionados con los contenidos impartidos en las asignaturas de programación de los grados y los ciclos formativos. Esto hace posible que los alumnos puedan escoger problemas relacionados con los conceptos que están aprendiendo o que quieren ampliar, y permite a los profesores recomendar problemas que consideran aptos para el nivel de sus alumnos.

A pesar de esto, no cuenta con un sistema de recomendación ni con un sistema puntuación que permita medir la habilidad de sus usuarios o la dificultad de los problemas.

[5] https://uva.onlinejudge.org
[6] https://www.aceptaelreto.com/

2.2. Sistemas de Recomendación

Son algoritmos capaces de predecir las "preferencias" o predisposición de los usuarios a la hora de calificar una serie de productos o ítems, los cuales luego ofrece como recomendación. Estos sistemas son una parte fundamental de aquellas aplicaciones que buscan vender productos (Amazon, AliExpress, etc.), aplicaciones que ofrecen servicios de streaming (Netflix, Hulu, etc.), e incluso aplicaciones que buscan conectar unos usuarios con otros (LinkedIn, Tinder, etc.), ya que permite que sus usuarios descubran nuevos productos o personas que se ajustan a su perfil (Ricci et al., 2010). Las técnicas utilizadas para realizar estas predicciones son variadas, pero destacan dos, que se detallan a continuación.

2.2.1. Recomendadores basados en el contenido

Para estos sistemas, los productos cuentan con información que les identifica (descripciones, etiquetas, etc.), y los usuarios cuentan con un perfil que identifica sus gustos y preferencias. Estos perfiles pueden ser temporales y no requerir que el usuario se registre o inicie sesión, y en muchos casos se generan de forma oculta, sin necesidad de que se introduzcan datos personales.

A la hora de recomendar un producto, el sistema usa la información disponible del usuario, y la compara con la información de los productos, recomendando aquellos cuyo grado de similitud sea mayor y descartando el resto (Terveen y Hill, 2001).

Un ejemplo claro de esto son los sitios dedicados al comercio electrónico, que utilizan información de los productos que el usuario ha visualizado para recomendar el mismo tipo de producto pero de otra marca, de otro vendedor, o productos que son distintos pero que pertenecen a la misma categoría.

2.2.2. Recomendadores basados en filtrado colaborativo

Estos sistemas hacen uso del historial de actividad y las valoraciones realizadas por el usuario: los productos que ha comprado, visitado o incluido en su lista de deseos, las películas que ha visto recientemente, los artistas musicales que le han gustado, etc[7].

Normalmente, estas calificaciones de productos se representan mediante matrices usuario-producto, en las cuales las filas representan los usuarios, las columnas representan los productos y el contenido de las celdas indica

[7] https://blogs.ams.org/mathgradblog/2015/05/25/online-recommender-systems-website-want/

la calificación de un usuario X para un producto Y. Entre sus variantes se encuentran:

- Vecino más próximo: Se basa en la similitud que existe entre usuarios y productos. Para un usuario X, se busca a otros usuarios que pertenezcan al mismo "vecindario", o lo que es lo mismo, que tengan perfiles similares. Una vez obtenidos los vecinos más cercanos, se realiza una predicción sobre la calificación que el usuario X otorgaría a aquellos productos que sus vecinos ya han calificado, pero que él aún no ha revisado. Dependiendo de las calificaciones generadas, se recomendaran unos productos u otros.

- Basado en el producto: La principal diferencia con el método anterior es que, en lugar de calcular la similitud entre usuarios, se centra en determinar la similitud existente entre los productos que le gustan al usuario con el resto de productos de la base de datos.

2.2.3. Obstáculos

Estas técnicas de recomendación no son perfectas. Dependiendo de la cantidad de información manejada por las aplicaciones, el tipo de datos asociados a los productos y el tipo de usuarios que admiten, nos podemos encontrar con distintos obstáculos:

- Arranque en frío: Se da cuando se introduce un nuevo usuario o producto al sistema. Dado que no existen datos suficientes para crear un perfil, las recomendaciones que se hagan a estos usuarios no serán tan acertadas como aquellas que se hagan a usuarios antiguos con un extenso historial de actividad.

- Productos sin calificar: Ocurre en sistemas que cuentan con una gran cantidad de productos. En estos casos, los usuarios no serán capaces de revisar todos los productos del sistema, provocando que muchos de estos productos no dispongan de ningún tipo de calificación.

- Escalabilidad: Problema especifico de sistemas que manejan una gran cantidad de datos y que viene a decir que, cuantos más datos se tengan que procesar, mayor será el coste del hardware y software necesario para implementar y/o mantener el sistema.

- Problema de la 'Oveja Negra': Con *Oveja Negra* nos referimos al tipo de usuario cuyas opiniones o calificaciones sobre los productos ofrecidos son "insólitas" al compararlas con los de otros usuarios. Un ejemplo de este tipo de usuario sería aquel que, usando una aplicación similar a Netflix, califica con 1/5 estrellas a *El padrino*, pero califica *El padrino*

II y *El padrino III* con 5/5 estrellas. Afecta a los recomendadores basados en filtrado colaborativo.

- Ataques 'Shilling': Ocurre en aquellas aplicaciones que permiten que cualquier usuario pueda opinar o calificar un producto más de una vez. Estos ataques consisten en que un grupo de usuarios se organice y califique positivamente sus propios productos, o negativamente a los de la competencia, alterando las recomendaciones de tal manera que les beneficie.

2.3. Sistemas de Puntuación

Los sistemas de puntuación suelen ser utilizados para medir el nivel de habilidad de los jugadores que participan en un juego. No obstante, el estudio de jueces en línea reveló que aquellos que tratan de medir la habilidad de sus usuarios lo hace mediante sistemas de puntuación, siendo el más habitual el sistema de puntuación ELO.

El sistema de puntuación ELO (Elo, 1978) fue desarrollado originalmente para calcular puntuaciones basadas en cálculos estadísticos de jugadores de ajedrez en la United States Chess Federation (USCF)[8] con el objetivo de determinar su nivel de habilidad de una manera cuantitativa. Actualmente se usa para muchos más ámbitos tales como videojuegos o juegos de mesa[9].

El funcionamiento del sistema es simple: cada jugador tiene asociado una puntuación, la cual se actualiza con cada enfrentamiento que tenga con otro jugador. Esta modificación de puntuación, que puede ser positiva o negativa, depende de un valor denominado 'expectativa', el cual se puede interpretar como la probabilidad que tiene un jugador de ganar la partida. Esta expectativa se calcula a partir de las puntuaciones de los usuarios enfrentados: si un jugador X cuenta con una puntuación superior a la de su adversario Y, la probabilidad de ganar del jugador X será mayor que la del jugador Y. Para los casos en los que los jugadores cuentan con la misma puntuación, la probabilidad para ambos será del 50 %. Se debe cumplir que la suma de las expectativas de dos jugadores enfrentados sea igual a 1 (100 %).

Este valor es uno de los factores que determinará cuan grande será la ganancia o perdida de puntos tras un encuentro. Si un jugador pierde una partida contra un jugador con la misma puntuación (expectativa del 50 %), su perdida de puntuación será mucho menor que la perdida generada tras una partida perdida en la que su expectativa rondaba el 90 %.

[8] https://new.uschess.org/about/
[9] https://mwomercs.com/news/2017/06/1839-patch-notes-14120-20jun2017

En el sistema ELO la puntuación de un usuario se conoce con el nombre de puntuación o ranking ELO. Si el jugador A tiene una puntuación ELO R_a y el jugador B tiene una puntuación ELO R_b, la fórmula exacta (usando una función logística) de la expectativa de A y B (E_a y E_b respectivamente) son:

$$E_a = \frac{1}{1 + 10^{\frac{R_b - R_a}{400}}} \tag{2.1}$$

$$E_b = \frac{1}{1 + 10^{\frac{R_a - R_b}{400}}} \tag{2.2}$$

El 400 no es un valor aleatorio, sino que representa la diferencia de puntuación que debe haber entre 2 jugadores para que la probabilidad de ganar de uno de ellos sea 10 veces mayor que la probabilidad de ganar de su adversario. Este valor también depende del rango de puntuaciones ELO con el que se trabaje.

Una vez finalizado el enfrentamiento y obtenidas las expectativas de ambos jugadores, las nuevas puntuaciones se obtienen usando las siguientes fórmula:

$$R'_a = R_a + K(S_a - E_a) \tag{2.3}$$

$$R'_b = R_b + K(S_b - E_b) \tag{2.4}$$

Las puntuaciones se actualizarán en cuanto se conozca el resultado de la partida, S_a y S_b. Si el jugador A vence al jugador B, S_a valdrá 1 y S_b valdrá 0. En el caso contrario, S_b valdrá 1 y S_a valdrá 0. En caso de empate S_a y S_b valdrán 0.5.

La constante K, denominada "factor K", es una variable que modifica el margen de variación en la modificación de las puntuaciones. Este factor se establece de acuerdo con la puntuación ELO de un jugador. Actualmente en la USCF su valor es de 32 para jugadores con puntuaciones por debajo de 2100, 24 para jugadores entre 2100 y 2400 y 16 para jugadores por encima de 2400. Esto permite a jugadores principiantes (con baja puntuación ELO) incrementar su puntuación de forma más rápida utilizando un factor K alto, a diferencia de jugadores con alta puntuación ELO, dado que estos utilizaran un factor K más bajo.

Este sistema cuenta con métodos que permiten conocer el valor real o aproximado de la habilidad de sus jugadores, lo cual facilita el empareja-

miento de aquellos jugadores con puntuaciones similares.

Uno de los aspectos positivos del sistema ELO es que es fácil de configurar y extender. Numerosas mejoras se han desarrollado para adecuar el sistema a necesidades especificas. Entre ellas destacamos dos: Glicko y TrueSkill, descritos a continuación.

2.3.1. Sistema de puntuación Glicko

El sistema Glicko (Glickman, 1995), fue desarrollado por Mark Glickman en 1995. Su principal contribución con respecto al modelo ELO original es la inclusión de una nueva variable de "confiabilidad de la puntuación", denominada RD (Rating Deviation) o desviación de la puntuación, la cual mide la precisión de la puntuación de un jugador.

Por ejemplo un jugador que participa en partidas con poca diferencia de tiempo entre ellas tiene un nivel RD bajo, representativo de que su puntuación se considera más precisa. Por otra parte, un jugador que juega menos a menudo tiene un RD alto debido a la poca información reciente que se tiene de éste, considerando que su puntuación verdadera no es realmente conocida.

Un jugador con puntuación de 1500 y una RD de 50 tiene una puntuación real entre 1400 y 1600 con una confianza del 95 %. Se suma y se resta dos veces la RD de la puntuación del jugador para calcular este rango. Después de una partida, el incremento de la puntuación depende de la RD: el cambio es menor cuando la RD es baja, y también cuando la RD de su oponente es alta. La RD disminuye después de cada partida, pero se incrementa lentamente tras un tiempo de inactividad.

2.3.2. Sistema de puntuación TrueSkill

TrueSkill (Herbrich et al., 2007) es una extensión del sistema ELO de aproximación bayesiana desarrollado por Microsoft para Xbox Live con el objetivo de soportar enfrentamiento entre facciones o partidas de más de 2 jugadores. TrueSkill solo usa la clasificación final de todos los equipos en una partida para actualizar las estimaciones de habilidades (rangos) de todos los jugadores que jueguen en esa partida.

La mayor diferencia con otros sistemas de clasificación es que en TrueSkill la habilidad se caracteriza por dos factores:

- La habilidad media del jugador (μ)
- El grado de incertidumbre en la habilidad del jugador (σ)

El sistema de clasificación mantiene una creencia en la habilidad de cada jugador al usar estos dos números. Si la incertidumbre es alta, el sistema de clasificación aún no conoce exactamente la habilidad del jugador. En contraste, si la incertidumbre es pequeña, el sistema de clasificación tiene una fuerte creencia de que la habilidad del jugador está cerca de la habilidad promedio.

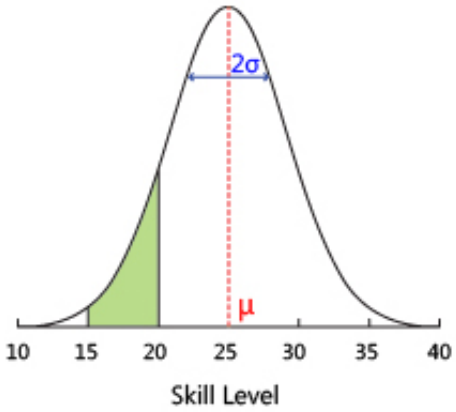


Figura 2.1: Nivel de habilidad de un jugador.

Mantener un valor de incertidumbre permite al sistema realizar grandes cambios en las estimaciones de habilidades al principio. Estos cambios en la estimación irán descendiendo en magnitud conforme aumente el número de partidas completadas por el jugador. Esto permite que el sistema de clasificación TrueSkill pueda identificar las habilidades de los jugadores individuales a partir de un número reducido de partidas.

En definitiva, este sistema es capaz de rastrear la incertidumbre acerca de las habilidades de los jugadores, modelar explícitamente el emparejamiento, puede lidiar con cualquier número de entidades competidoras y puede inferir habilidades individuales de los resultados del equipo.

En la figura 2.1 podemos observar un ejemplo de la estimación del nivel de un jugador en la que el área verde es la creencia de TrueSkill de que el jugador tiene una habilidad entre los niveles 15 y 20.

2.4. Conclusiones

A pesar de la gran cantidad de jueces en línea disponibles en Internet, hemos podido comprobar que la muchos de ellos carecen de sistemas de recomendación o sistemas que designen la dificultad de los problemas o el nivel

de habilidad de los usuarios, y si los tienen, suelen estar ocultos o explicados de manera simplificada. Un dato a resaltar es que aquellos jueces que si contaban con un sistema de puntuación utilizan una variación del sistema de puntuación ELO.

Por otra parte, hemos aprendido que los sistemas de recomendación no son herramientas perfectas libres de inconvenientes, sino que presentan distintos obstáculos según el entorno y datos que se quieren tratar.

Con todo esto en mente, procederemos a adaptar el sistema de puntuación ELO para su uso en jueces en línea en el siguiente capítulo.

Capítulo 3

Adaptación de ELO para recomendación en jueces en línea

"Deja de preocuparte por envejecer y piensa en crecer"
— El animal moribundo, Philip Roth (2001)

En este capítulo se describirán los pasos que seguimos para adaptar el sistema de puntuación ELO para incluirlo en jueces en línea y los tipos de recomendación que desarrollamos.

Tras realizar la investigación de los sistemas de recomendación empleados tanto en algunos jueces en línea como en otros entornos, concluimos que desarrollar un sistema de recomendación basado en filtrado colaborativo no solo tenía asociado más obstáculos que otros, sino que además suponía una mayor complejidad a la hora de ponerlo en funcionamiento en un juez en línea, puesto que tendría que incluir la opción de valorar problemas, lo que supondría reestructurar la forma en la que se tratan los problemas y usuarios del sitio.

Esto, unido al hecho de que los jueces en línea no suelen guardar información para generar perfiles basados en preferencias, nos hizo considerar la opción de desarrollar un nuevo tipo de sistema de recomendación basado en el sistema de puntuación ELO que, en lugar de utilizar datos como las descripciones de problemas, emparejase a usuarios y problemas basándose en valores de habilidad y dificultad obtenidos a través de las interacciones que realicen estos entre sí.

El sistema de puntuación de ELO es flexible: puede modelar habilidades cambiantes, pero no hace suposiciones fijas sobre la naturaleza del aprendizaje. Debido a esto, no se debe esperar que proporcione resultados precisos

para una situación particular. Pero se puede aplicar fácilmente en una amplia gama de situaciones y proporciona una precisión razonable. También es fácil de adaptar el sistema ELO para diferentes situaciones como, por ejemplo, pruebas que contienen preguntas con respuestas múltiples o que incorporan el tiempo de las pruebas como otro factor a tener en cuenta en el cálculo de puntuaciones (Pelanek, 2016).

A partir de esto, podemos decir que el sistema ELO es simple, robusto, efectivo y que puede ser fácilmente adaptado para su uso en el desarrollo de jueces en línea.

En los siguientes apartados explicaremos los cambios que se tuvieron que realizar para adaptar el sistema de puntuación ELO al modelo de los jueces en línea, y como se diseñó un sistema de recomendación que solo hace uso de estas puntuaciones.

3.1. Adaptación del sistema ELO

Como se vio en la sección 2.3 del capítulo anterior, el sistema de puntuación ELO se basa en el enfrentamiento entre dos usuarios. Para adaptar este sistema a un juez en línea consideraremos el envío de una solución a un problema como un enfrentamiento entre el usuario y el problema. Esto quiere decir que interpretaremos el envío de una solución a un problema como un juego o "partida" entre un usuario y el problema que intenta resolver. De esta manera, asignaremos un valor para cada estudiante y un valor para cada problema que nos indicaran su puntuación ELO, que no será otra cosa que la representación estadística y simbólica de sus niveles de habilidad y dificultad respectivamente.

El resultado de un envío dará como "ganador" a un usuario en caso de que el juez devuelva un veredicto que indique que la solución recibida es correcta, o como perdedor en caso de obtener algún tipo de veredicto de tipo erróneo, modificando la puntuación de ambos en cualquiera de los dos casos. De esta manera, podemos estimar dinámicamente la habilidad de los usuarios y la dificultad de los problemas.

Se debe tener en cuenta que, a diferencia del ajedrez, los enfrentamientos entre usuario y problema no pueden terminar como empate. Con respecto a las puntuaciones ELO que pueden alcanzarse, éstas estarán comprendidas entre 0 y 16 puntos, ambos inclusive.

La fórmula para actualizar la puntuación de un usuario (3.1) consta de cuatro componentes principales: la puntuación actual del usuario R_u, un

factor K, el resultado del enfrentamiento S_u y la expectativa E_u que tiene el usuario de ganar el enfrentamiento.

$$R'_u = R_u + K \times (S_u - E_u) \tag{3.1}$$

En los siguientes apartados se detallan las modificaciones que se tuvieron que realizar a esta fórmula para adaptarlas a un juez en línea.

3.1.1. Expectativa

En el sistema ELO, la expectativa de ganar una partida viene dada por la función logística con respecto a la diferencia entre las puntuaciones de los usuarios que se enfrentan. En el caso de un juez en línea, si un usuario tiene una puntuación ELO R_u y el problema al que se enfrenta tiene una puntuación ELO R_p, la expectativa de cada uno de ellos, E_u y E_p, se calcula de la siguiente manera.

$$E_u = \frac{1}{1 + 10^{\frac{R_p - R_u}{MAXDIFF}}} \tag{3.2}$$

$$E_p = \frac{1}{1 + 10^{\frac{R_u - R_p}{MAXDIFF}}} \tag{3.3}$$

Un usuario o problema con $MAXDIFF$ puntos de ventaja sobre su oponente, tendrá una probabilidad de ganar 10 veces mayor a la probabilidad de ganar de su oponente.

En el sistema ELO empleado por USCF, el rango de puntuaciones va desde 0 hasta 3000, y utilizan un $MAXDIFF$ de 400. Dado que nosotros utilizamos un rango de puntuaciones que va de 0 a 16, y que la máxima diferencia de puntuaciones solo puede ser de 16 y no 400, fue necesario calcular un nuevo valor que generase resultados similares a los obtenidos utilizando la fórmula original. Tras muchas pruebas, establecimos el valor de $MAXDIFF$ a 8 dado que cumple la condición mencionada en el párrafo anterior.

Las expectativas se utilizarán para calcular la nueva puntuación para el usuario y problema usando las siguientes formulas:

$$R'_{\mathrm{u}} = R_{\mathrm{u}} + K \times (S_u - E_{\mathrm{u}}) \tag{3.4}$$

$$R'_{\mathrm{p}} = R_{\mathrm{p}} + K \times (S_p - E_{\mathrm{p}}) \tag{3.5}$$

Donde R'_{u}, R'_{p}, E_{u} y E_{p} representan las puntuaciones y expectativas del usuario y problema respectivamente.

Es importante indicar que el valor que tomarán S_u y S_p serán 0 en caso de derrota y 1 en caso de victoria. Si el usuario gana el enfrentamiento, S_u tomará el valor de 1 y S_p será 0. En caso contrario S_p tomará el valor de 1 y S_u será 0.

Esto quiere decir que el resultado de $K \times (S_u - E_{\mathrm{u}})$ o $K \times (S_p - E_{\mathrm{p}})$ solo será positivo para el ganador de la partida, lo cual añadirá X puntos a su actual puntuación, mientras que el otro contrincante verá su puntuación reducida en X puntos.

3.1.2. Factor K

Como ya se comento en la sección 2.3, el valor de la constante K en la fórmula de actualización determina el comportamiento del sistema. Si K es pequeño, la estimación converge lentamente; si K es grande, la estimación es inestable ya que da una ponderación muy grande.

Una diferencia importante entre las aplicaciones del sistema de puntuación de ELO en juegos y educación es una asimetría entre estudiantes y problemas a resolver. Para los problemas esperamos que su dificultad sea aproximadamente constante. En algunos casos, puede ser útil hacer un seguimiento de los cambios en la dificultad, pero tales casos son excepciones ya que no se espera que los cambios en la dificultad sean grandes. Por otro lado, sí se esperan cambios en las habilidades de los estudiantes, puesto que ese es el objetivo principal de los sistemas educativos (Papousek et al., 2014).

En muchas aplicaciones educativas también hay una asimetría entre las respuestas correctas e incorrectas. Una respuesta a un problema no solo es evidencia del conocimiento del estudiante, sino también una oportunidad para aprender. En tales situaciones podemos realizar distintos ajustes, por ejemplo, añadiendo constantes para que el cálculo del nuevo ELO tenga en cuenta el número total de envíos correctos y erróneos, o el tipo de error que devuelve el juez y, dependiendo de este, establecer un valor K más o menos generoso en el nuevo cálculo.

En el principio se realizaron pruebas preliminares con distintos tipos de ecuaciones para calcular el factor K adecuado para los enfrentamientos entre usuarios y problemas. Tras estas pruebas preliminares se decidió establecer 3 tipos o variaciones del sistema ELO descrito, en los cuales el factor K se calcula de distinta manera.

Para entender las fórmulas expuestas en los siguientes apartados, es necesario explicar el significado de las variables que aparecen en ellas:

- ΔR: Es el valor absoluto de la diferencia de puntuaciones del Usuario y Problema.
- R_{MAX}: La puntuación máxima que pueden alcanzar usuarios y problemas. Para nuestro sistema, este valor es 16.
- $TRIES_{\text{MAX}}$: El número máximo de envíos que se cree que un usuario necesita para resolver en media cualquier problema.
- N_{TRIES}: El número de envíos que lleva un usuario para el mismo problema.

3.1.2.1. Tipo 1

Para esta variación del sistema de puntuación, el cálculo de nuevas puntuaciones se realizará si se cumple una de las siguientes condiciones:

- El usuario consigue un veredicto aceptado. Esto convierte al usuario en "ganador de la partida", dado que ha conseguido resolver el problema.
- El usuario abandona un problema. Abandonar implica que el usuario deja de lado un problema sin resolver en favor de intentar otro problema distinto. Esto resulta en una victoria del problema que se ha abandonado.

Distinguimos 2 casos para el cálculo de K:

- Caso en el que el ganador tiene una puntuación inferior a la de su oponente.

$$K = \frac{\Delta R + R_{\text{MAX}}}{2 \times R_{\text{MAX}}} \tag{3.6}$$

- Caso en el que el ganador tiene una puntuación superior a la de su oponente.

$$K = \left(1 - \frac{\Delta R}{R_{\text{MAX}}}\right) \times \frac{1}{2} \tag{3.7}$$

La razón por la que se hace esta distinción a la hora de calcular K es que, dada la naturaleza de los jueces en línea, los usuarios tienen libertad para intentar resolver cualquiera de los problemas que existen. Esto significa que se pueden dar los casos en los que un usuario de puntuación muy alta resuelva un problema de puntuación baja, o el caso contrario, en el que un usuario con puntuación muy baja resuelva un problema de puntuación alta.

Partiendo del caso en el que un usuario con puntuación baja resuelve un problema de puntuación alta, la fórmula 3.6 se encargaría de generar un K que modifique la puntuación del usuario de tal manera que refleje que su puntuación antigua no era acertada, y que estaba muy por debajo de su habilidad real. Por el contrario, si se usase la fórmula 3.7 para este caso, la K obtenida sería mucho más pequeña, resultando en un incremento de la puntuación del usuario mucho menor, y menos merecida.

Lo mismo ocurre en el caso en el que el ganador cuente con una puntuación mucho mayor a la de su oponente: si se aplica la fórmula 3.7, el ganador experimentará un incremento muy bajo de su puntuación, puesto que al tener una puntuación superior, su expectativa de ganar era mucho mayor, por lo cual su victoria no es una sorpresa, y por lo tanto no merece ganar tanta puntuación. Pero si se aplicase la fórmula 3.6, el incremento de su puntuación sería mucho mayor, lo que provocaría que la puntuación de su oponente disminuya de la misma manera, lo cual no sería justo.

El uso de dos fórmulas es fundamental para conseguir que los ajustes de puntuaciones se hagan de manera justa y proporcionada.

3.1.2.2. Tipo 2

Deriva del tipo 1, con la diferencia de que ahora se tiene en cuenta el número de envíos fallidos.

Las fórmulas para calcular el valor de K son las siguientes:

- Caso en el que el ganador tiene una puntuación inferior a la de su oponente.

$$K = \frac{\Delta R + R_{\text{MAX}}}{2 \times R_{\text{MAX}} \times N_{\text{TRIES}}} \tag{3.8}$$

- Caso en el que el ganador tiene una puntuación superior a la de su oponente.

$$K = \left(1 - \frac{\Delta R}{R_{\text{MAX}}}\right) \times \frac{1}{2 \times N_{\text{TRIES}}} \tag{3.9}$$

La principal diferencia con respecto al tipo 1 es la inclusión de la variable N_{TRIES}, cuyo valor es el número de intentos usados por el usuario para resolver el problema. Para este tipo, N_{TRIES} puede tomar valores de 1 hasta $TRIES_{\text{MAX}}$. Para los casos en los que el usuario haya enviado más de $TRIES_{\text{MAX}}$ soluciones, N_{TRIES} tomará el valor de $TRIES_{\text{MAX}}$.

3.1.2.3. Tipo 3

Este tipo es un derivado del Tipo 2, y se diferencia principalmente en que no solo aplicará el cálculo de ELO al resolver o abandonar un problema, sino que también se aplicará si el número de intentos realizados para un problema es igual a $TRIES_{\text{MAX}}$. Esto significa que, si el usuario no consigue resolver el problema en $TRIES_{\text{MAX}}$, se considerará que el usuario ha perdido la partida, y se le reducirá la puntuación. Tras esto, el contador de intentos se reiniciará y el usuario tendrá otros $TRIES_{\text{MAX}}$ intentos para resolver el problema antes de volver a decrementar su puntuación.

Las fórmulas para calcular el valor de K son las siguientes:

- Caso en el que el ganador tiene una puntuación inferior a la de su oponente.

$$K = \frac{\Delta R + R_{\text{MAX}}}{2 \times R_{\text{MAX}} \times N_{\text{TRIES}}} \tag{3.10}$$

- Caso en el que el ganador tiene una puntuación superior a la de su oponente.

$$K = \left(1 - \frac{\Delta R}{R_{\text{MAX}}}\right) \times \frac{1}{2 \times N_{\text{TRIES}}} \tag{3.11}$$

3.1.3. Cálculo de la nueva puntuación ELO

El cálculo de la nueva puntuación procesará la expectativa del usuario y del problema del envío en cuestión, y junto con el ajuste de K establecerá una nueva puntuación ELO para ambos.

Las ecuaciones son:

$$R'_{\text{winner}} = R_{\text{winner}} + K(S_{\text{winner}} - E_{\text{winner}}) \tag{3.12}$$

y

$$R'_{\text{loser}} = R_{\text{loser}} + K(S_{\text{loser}} - E_{\text{loser}}) \tag{3.13}$$

Donde R'_{winner}, R'_{loser}, E_{winner} y E_{loser} representan las puntuaciones y expectativas del ganador y perdedor respectivamente. S_{winner} tomará el valor de 1 y S_{loser} tomará el valor de 0.

3.2. Recomendación

Antes de explicar como se generan las recomendaciones es necesario indicar que los usuarios cuentan con dos tipos de puntuaciones distintas. El primer tipo de puntuación, denominada puntuación global, es aquella puntuación que se actualiza tras cualquier enfrentamiento contra un problema, y sirve para medir la habilidad del usuario a la hora de resolver problemas de cualquier índole. El segundo tipo de puntuación se denomina puntuación por categoría, y como su nombre indica, es la puntuación que un usuario tiene asociada por cada una de las categorías de problemas que existen en un juez en línea. Las puntuaciones por categoría solo se actualizarán si el usuario se enfrenta a problemas que pertenecen a una categoría especifica.

La idea de almacenar una puntuación distinta por cada categoría existente surgió tras el estudio de los jueces en línea, en el cual pudimos observar que en algunos de ellos, como *¡Acepta el reto!*, los problemas suelen estar categorizados según distintos "ejes": conceptos de programación, temática del enunciado, aparición en concursos, en exámenes, etc. Esta división sirve a los usuarios como guía para encontrar problemas que puedan resolver con sus conocimientos actuales, y nos sirve a nosotros para determinar el tipo de problemas que se deben priorizar a la hora de generar una recomendación.

3.2.1. Proceso de recomendación

Una vez conocidas las puntuaciones de los usuarios y problemas, el proceso de recomendación consistirá en encontrar aquellos problemas que tengan puntuaciones similares a la del usuario y que aún no hayan sido resueltos por él. Para ello se incluyen los problemas en una lista, la cual se ordena por distancia a la puntuación del usuario, y de la cual se escogen aquellos que tengan las puntuaciones más cercanas.

Con este proceso en mente y conociendo los tipos de puntuaciones asociados a los usuarios, se decidió proponer dos métodos de recomendación distintos:

3.2.1.1. Recomendaciones globales

Este tipo de recomendación se aplica a todos los usuarios de la plataforma. El cálculo de la similitud entre usuario y problemas se realizará utilizando estrictamente la puntuación global del usuario.

$$\Delta R = \mid R_{\text{u Global}} - R_p \mid \tag{3.14}$$

Se decidió limitar el número de problemas recomendados por este método a tres.

3.2.1.2. Recomendaciones por categoría

La idea detrás de este tipo de recomendación parte del supuesto de que si un usuario ha dedicado el 99 % de sus envíos a problemas que pertenecen a una determinada categoría, por ejemplo "Recursión", lo más lógico será asumir que sus siguientes envíos van a estar dirigidos a problemas pertenecientes a la misma categoría, por lo que resulta de sumo interés recomendar problemas de la misma categoría que además sean asequibles para el usuario.

Este tipo de recomendación solo estará disponible para usuarios que hayan visto su puntuación modificada tras ganar o perder un enfrentamiento contra problemas que pertenecen a alguna categoría.

El proceso de selección de problemas es similar al anterior, con la diferencia de que, en lugar de usar la puntuación global del usuario para calcular la similitud, se utiliza la puntuación o puntuaciones del usuario asociadas a las categorías del problema que intenta resolver:

$$\Delta R = \mid R_{\text{u Categoria}} - R_p \mid \tag{3.15}$$

Cabe resaltar que se realiza una selección de problemas por cada una de las categorías con las que el usuario ha interactuado. Esto significa que, si limitamos el número de problemas recomendados por categoría a tres, un usuario que haya interactuado con N categorías podría recibir un máximo de $3 \times N$ problemas recomendados.

3.3. Conclusiones

En este capítulo no solo se ha explicado las recomendaciones a realizar y los cambios realizados a la fórmulas originales del sistema ELO, sino que también hemos establecidos los distintos casos en los cuales consideramos que es necesario actualizar las puntuaciones y en cuales no, desarrollando así 3 distintas variaciones del sistema ELO original. Estas variaciones serán puestas a prueba en el siguiente capítulo, en donde se usarán con datos reales de un juez en línea.

Capítulo 4

Análisis y Evaluación

"No todo lo que es de oro reluce, ni toda la gente errante anda perdida"
— El Señor de los Anillos, J.R.R. Tolkien (1954)

Con el fin de analizar el comportamiento de los sistemas de puntuación y recomendación propuestos en el capítulo anterior, en este capítulo realizaremos tres estudios distintos. Por una parte se hará un estudio de las características de la base de datos de *¡Acepta el reto!* y se explicarán las modificaciones y filtrado que se tuvieron que realizar sobre ésta para adaptarla a nuestros sistemas de puntuación.

Por otra parte, se realizará un análisis de los sistemas de puntuación propuestos en el capítulo anterior, en donde aplicaremos cada uno de estos a una misma base de datos para calcular puntuaciones y se evaluarán las diferencias entre los resultados obtenidos.

El último estudio consistirá en evaluar la precisión de las recomendaciones realizadas mediante el análisis de cuantos usuarios muestran interés en los problemas que se le recomiendan usando la métrica 1-hit.

4.1. Base de Datos de *¡Acepta el reto!*

Para desarrollar este TFG se nos proporcionó una copia de la base de datos de *¡Acepta el reto!*, la cual contenía toda la información relevante sobre los usuarios que utilizan el juez, de los problemas a los que se enfrentan, de las categorías a las que pertenecen, y de los envíos que se han realizado desde el momento de su concepción hasta la fecha en la que lo recibimos.

La copia que recibimos cuenta con las siguientes tablas:

- Tabla "submission", con aproximadamente 260.000 entradas, es la tabla principal y contiene información relacionada con los envíos realizados por todos los usuarios. Los campos a destacar son:

  - id: Identificador numérico del envío.
  - problem_id: Identificador numérico del problema.
  - user_id: Identificador numérico del usuario.
  - status: Veredicto del envío.

- Tabla "problem", que contiene toda la información relacionada con los problemas de *¡Acepta el reto!*: titulo, número de veces que ha sido resuelto, número de veces que ha sido intentado, número de usuarios que lo han intentado, etc.

- Tabla "category", que contiene toda la información relacionada con las categorías de problemas de *¡Acepta el reto!*: número de problemas pertenecientes a cada categoría, número de envíos erróneos de cada categoría, etc.

- Tabla "problemcategories", que contiene la relación de los problemas y las categorías a las que pertenecen.

Existen otras tablas además de las mencionadas anteriormente, pero la información que contienen no es relevante para el desarrollo de este proyecto.

La Figura 4.2 muestra como, año tras año, el número de envíos realizados en *¡Acepta el reto!* ha ido aumentando progresivamente. Con este dato, y tras un análisis más a fondo, nos encontramos con que los envíos realizados a partir del 2017 equivalen a más del 50 % de los envíos realizados desde la concepción del juez en línea.

No obstante, el análisis también reveló que, a pesar del abundante número de envíos, casi el 20 % de los usuarios registrados hasta la fecha no ha conseguido resolver un solo problema. La Figura 4.1 muestra como muchos de ellos (35 %) no han realizado más de 5 envíos desde que se registraron.

En la Tabla 4.1 se pueden observar las características principales de la base de datos original, como el número de envíos realizados para los distintos tipos de veredicto que hay, el número de usuarios que han resuelto como mínimo un problema, y los que no lo han conseguido aún, la media de envíos realizados por los usuarios, la media de envíos que reciben los problemas, etc.

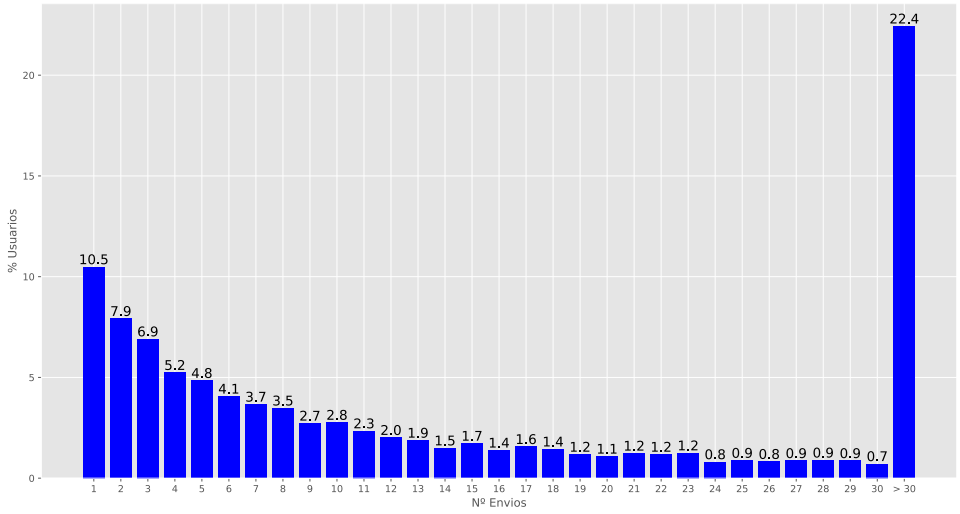


Figura 4.1: Porcentaje de usuarios que han realizado un máximo de X envíos

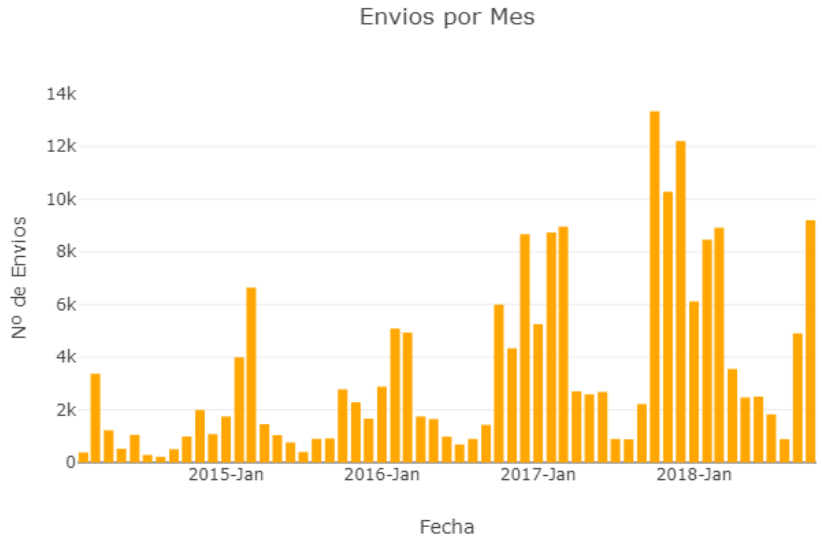


Figura 4.2: Diagrama de barras que muestra el incremento del número de envíos realizados por mes a lo largo de 4 años.

4.1.1. Filtrado de la base de datos

Una característica de *¡Acepta el reto!* es que los usuarios pueden resolver los mismos problemas más de una vez. Teniendo en cuenta que una de las condiciones para aplicar nuestros sistemas de puntuación ELO es el envío de una solución con estado AC / PE, cualquier envío con estos estados que realice un usuario a un problema que ya ha resuelto resultaría en un incremento constante de la puntuación del usuario y un decremento idéntico de la puntuación del problema.

Este tipo de envíos no es inusual y es que muchos usuarios no se conforman con enviar una solución correcta, sino que también buscan que su código consuma la menor cantidad de memoria posible, que se ejecute en el menor tiempo posible, etc. También hay que tener en cuenta que estamos considerando los envíos con estado PE como correctos, mientras que los usuarios de *¡Acepta el reto!* toman este veredicto como error, por lo que siguen enviando soluciones hasta alcanzar un estado AC.

Todo esto nos obligó a tomar la decisión de filtrar este tipo de envíos. Para ello, se seleccionaron todas las parejas usuario - problema de la tabla submission con al menos un envío con estado PE / AC, y se eliminaron todos los envíos posteriores a su primer envío correcto. Esto resultó en una reducción del 25 % (65,315) del número total de envíos original.

En la Tabla 4.1 se muestran las principales diferencias entre la base de datos original y la obtenida tras filtrar envíos innecesarios. Se puede apreciar que, además de la reducción del número total de envíos, la base filtrada presenta una reducción de más del 50 % de envíos con estado AC con respecto a la original. Este dato confirma que los usuarios de *¡Acepta el reto!* dedican gran parte de sus envíos a mejorar sus estadísticas con respecto a un problema (ranking, memoria, tiempo).

Otro tipo de filtrado que se realizó fue el borrado de algunas entradas de la tabla de categorías de problemas. El motivo de este borrado es que algunos de los identificadores de problemas de esta tabla no están presentes en la tabla de problemas, lo cual puede generar conflictos a la hora de insertar nuevos problemas mediante nuestra aplicación.

Un dato a resaltar de la tabla filtrada es que, como se puede apreciar en la Figura 4.3, más del 90 % de los usuarios que consiguen resolver sus problemas lo hacen utilizando 10 o menos intentos. Esto nos hizo considerar 10 como el número máximo de intentos que un usuario ha de realizar con un problema antes de penalizar su puntuación. Solo se tiene en cuenta en el sistema ELO de tipo 2 y tipo 3, descritos en las secciones 3.1.2.2 y 3.1.2.3.

	BBDD Original	**BBDD Filtrada**
Envíos		
Nº de Envíos	260.341	195.026
Nº de Envíos aceptados (AC o PE)	94.201	46.568
Nº de Envíos con estado AC	90.764	45.167
Nº de Envíos con estado PE	3.437	1.401
Nº de Envíos erróneos (Ni AC ni PE)	166.140	148.458
Nº de Envíos con estado WA	78.640	70.483
Nº de Envíos con estado TL	29.283	27.104
Nº de Envíos con estado RTE	32.390	29.141
Nº de Envíos con estado MLE	6.429	6.007
Nº de Envíos con estado CE	16.607	13.266
Nº de Envíos con otros estados	2.791	2.457
Media en Envíos por Mes	21.695	16.252
Fecha del Primer Envío	2014-02-17	2014-02-17
Fecha del Último Envío	2018-10-23	2018-10-23
Usuarios		
Nº de Usuarios	7.317	7.317
Nº de Usuarios con al menos un envío correcto	5.910	5.910
Nº de Usuarios con cero envíos correctos	1.407	1.407
Nº mínimo de Envíos por un Usuario	1	1
Nº máximo de Envíos por un Usuario	4.735	2.260
Media de Envíos por Usuario	35,5	26,6
Problemas		
Nº de Problemas	374	374
Nº de Problemas resueltos por al menos 1 usuario	374	374
Nº de Problemas sin resolver	0	0
Nº mínimo de Envíos para un Problema	19	5
Número máximo de Envíos para un Problema	11.938	6.780
Media de Envíos por problema	696	521,4

Tabla 4.1: Comparación entre la Base de datos original y la filtrada

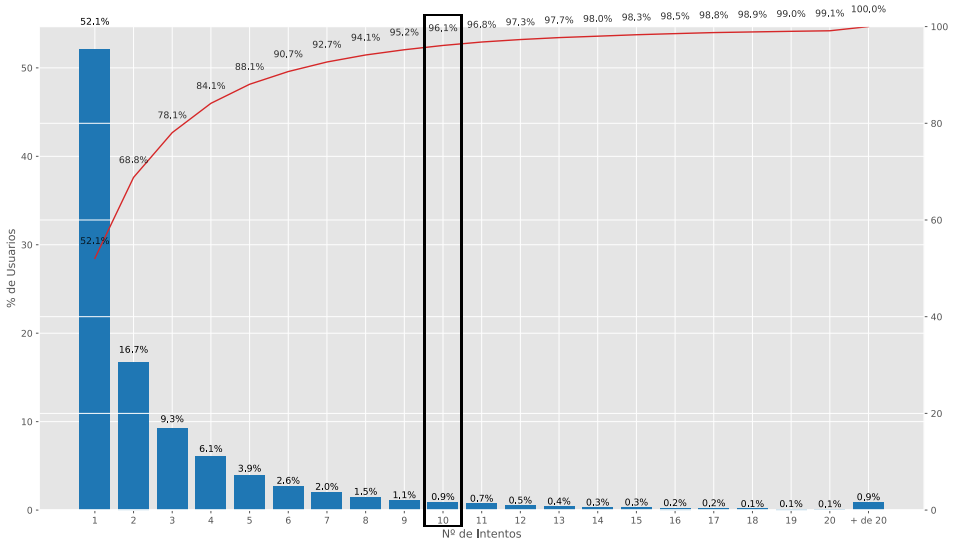


Figura 4.3: Distribución de los usuarios con respecto al número de intentos que requieren para resolver un problema.

4.1.2. Modificaciones realizadas sobre la base de datos

La base de datos que recibimos no estaba diseñada para almacenar información relacionada con la dificultad de los problemas o la habilidad de los usuarios. Por ello, antes de poder realizar algún tipo de prueba, era necesario hacer algunas modificaciones al esquema original.

Estas modificaciones incluyen:

- La adición de 2 nuevas columnas a la tabla "submission":
  - user_elo, que consiste en un número comprendido entre 0 y 16 y representa la puntuación del usuario en el momento del envío. Para que los cambios de puntuación no parezcan minúsculos, ésta se multiplicará por 100 antes de mostrarse al público.
  - problem_elo, idéntico a user_elo, con la diferencia de que representa la puntuación del problema.
- La creación de 2 nuevas tablas:
  - user_scores, que contiene el identificador del usuario, su puntuación ELO "global", y otras puntuaciones ELO correspondiente a distintas categorías de problemas. La puntuación "global" es la

puntuación principal del usuario, y se modifica al resolver o abandonar cualquier problema, independientemente de la categoría a la que pertenezca. Por otro lado, las puntuaciones ELO que corresponden a las categorías de problemas solo se modifican cuando el usuario se enfrenta a problemas de esa determinada categoría.

- problem_scores, que contiene el identificador del problema y su puntuación ELO.

4.2. Análisis de los distintos sistemas ELO

En esta sección se evaluarán los distintos sistemas ELO desarrollados en el capítulo anterior: el sistema ELO de Tipo 1, el del Tipo 2 y el del Tipo 3. Intentaremos medir cuales de ellos consiguen calcular puntuaciones que se aproximen al verdadero nivel de habilidad de los usuarios y la dificultad de los problemas. La metodología a seguir se compone de los siguientes pasos:

1. Se dividirá la tabla de envíos (ordenada cronológicamente) en dos mitades con el mismo número de entradas: datos de entrenamiento y datos de prueba.

2. Los datos de entrenamiento serán los datos a los cuales se aplicarán los sistemas ELO, calculando las puntuaciones de los problemas y de los usuarios que hayan realizado envíos correctos o abandonado algún problema.

3. Una vez calculadas las puntuaciones con los datos de entrenamiento, se compararán las distribuciones de puntuaciones de problemas y usuarios de los tres tipos en busca de diferencias significativas.

4. Por último, se realizará un análisis para determinar si los usuarios tienden a intentar problemas que poseen una puntuación similar al suyo o si, por el contrario, se inclinan por escoger problemas con una dificultad muy por encima (o por debajo) de su habilidad.

4.2.1. Diferencias en la distribución de puntuaciones de usuarios

Como se puede apreciar en la Figura 4.4, Figura 4.5, Figura 4.6, las distribuciones son casi idénticas para los 3 sistemas ELO. La razón por la cual la mayoría de usuarios cuenta con una puntuación ELO entre 7 y 9 es que una gran parte de ellos (51.6 %) no han realizado más de 9 envíos. Por otra parte, el porcentaje de usuarios que cuentan con al menos 3 envíos correctos es de solo 33.7 %. Si tenemos en cuenta que el número total de usuarios en el conjunto de datos de entrenamiento es de 4.031, esto significa que, hasta la

fecha en la que se realizó el último envío del conjunto de entrenamiento, más del 50 % de usuarios no eran muy activos, y que solo el 33 % había conseguido más de 3 envíos correctos. Hay que tener en cuenta que el incremento de la puntuación no depende solo del número de envíos correctos, sino también de la dificultad del problema al que se enfrentan. Esto quiere decir que si los problemas que los usuarios han resuelto tenían una puntuación ELO muy baja, el incremento de la puntuación del usuario sería de tan solo unas centésimas.

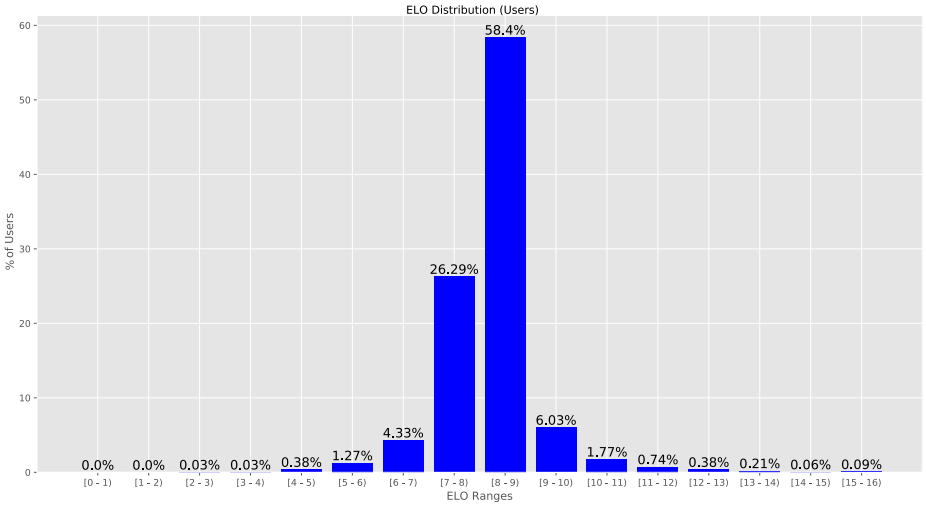


Figura 4.4: Distribución ELO de usuarios tras aplicar el sistema ELO de tipo 1 al conjunto de entrenamiento.

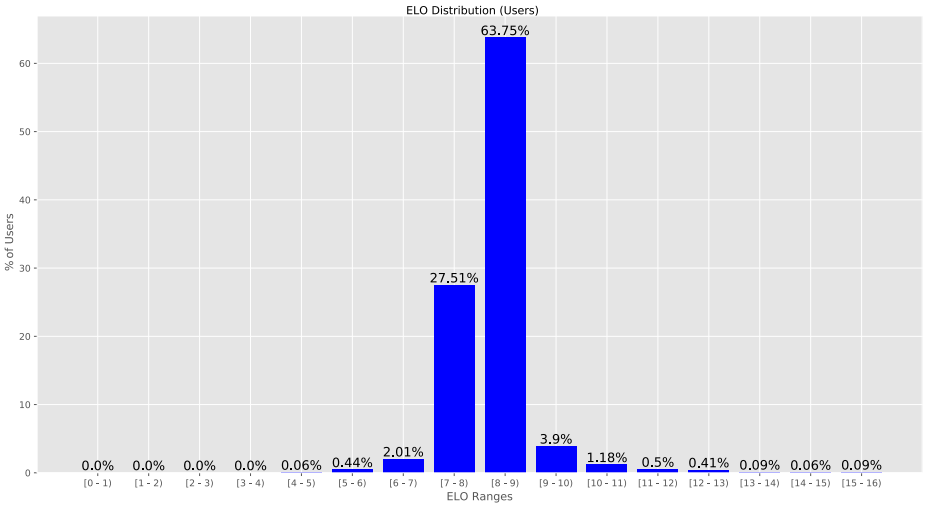


Figura 4.5: Distribución ELO de usuarios tras aplicar el sistema ELO de tipo 2 al conjunto de entrenamiento.

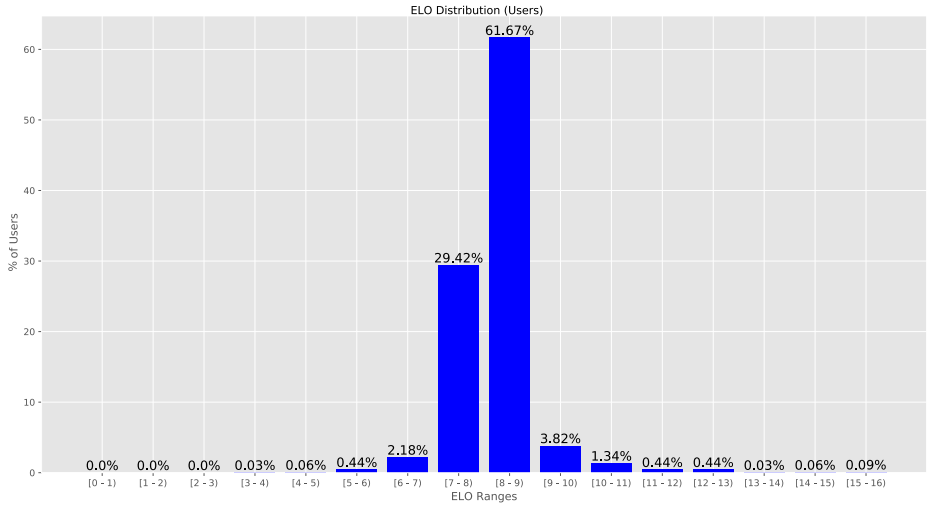


Figura 4.6: Distribución ELO de usuarios tras aplicar el sistema ELO de tipo 3 al conjunto de entrenamiento.

4.2.2. Diferencias en la evolución de la puntuación de los usuarios

Tras aplicar los sistemas de puntuación a todos los usuarios del conjunto de entrenamiento, se generaron gráficas que mostrasen la evolución de sus puntuaciones a lo largo del tiempo. Tras un estudio de todas ellas decidimos escoger al usuario 206 para comparar y estudiar las diferencias de aplicar cada uno de los sistemas de puntuación a un mismo usuario. Este usuario es especial por distintas razones:

- Es el autor de un gran número de envíos, lo que hace posible observar el comportamiento de nuestros sistemas ELO a largo plazo.
- No se ha limitado a enfrentarse a problemas de una sola categoría, sino que ha intentado resolver problemas de distintas categorías.
- Muchos de sus envíos son erróneos, por lo cual su puntuación oscilará mucho más que la de un usuario que acierta en todos sus intentos.
- Es un usuario que no se suele rendir, es decir, que no deja de enviar soluciones para un mismo problema hasta que consigue resolverlo.

Si observamos las 3 gráficas correspondientes con la evolución de la puntuación global en los distintos tipos (Figura 4.7, Figura 4.8, Figura 4.9), podemos observar que:

- La puntuación final que se obtiene para el tipo 1 (Figura 4.7) es menor que la obtenida en el tipo 2, a pesar de que no tiene en cuenta el número

de intentos en el cálculo de la K. La razón de esto es que la fórmula de K que no tiene en cuenta el número de intentos no verá su resultado divido si el usuario utiliza más de 1 intento para resolver el problema. Esto provocará que la K tenga un valor más grande en comparación con los K que se pueden obtener en el tipo 2 y, por consiguiente, que las perdidas de puntuación que experimente el usuario al abandonar un problema sean mayores.

- La evolución en la que se obtiene la puntuación más alta, además de ser la más estable, es el que corresponde con el tipo 2 (Figura 4.8). Esto ocurre porque en tipo 2 la fórmula que calcula el factor K empieza a tener en cuenta el número de intentos, y dado que nuestro usuario no suele abandonar los problemas que empieza, el número de intentos con el que se calculará la K será, en la mayoría de casos, el valor establecido como máximo (10). Esto significa que, si no abandona nunca, su puntuación solo podrá ir en aumento, aunque sea con un valor de K bajo.
- La puntuación final que se obtiene para el tipo 3 (Figura 4.9) es la menor de todas. Esto se debe a que esta variación del sistema ELO penaliza a los usuarios que más intentos de los que se consideran necesarios para resolver un problema. Dado que nuestro usuario no deja de enviar soluciones hasta resolver el problema, la perdida de puntuación por penalizaciones será mucho mayor que en los otros dos tipos.

Un aspecto importante a tener en cuenta al observar estas gráficas son las oscilaciones de las puntuaciones a lo largo del tiempo. En las gráficas del Tipo 1 (Figura 4.7) y Tipo 3 (Figura 4.9) podemos observar que las puntuaciones oscilan de forma más brusca y más frecuentemente que en el Tipo 2 (Figura 4.8). Esta oscilación es importante porque, dado que estamos intentando medir la habilidad real de un usuario, lo más normal es que su puntuación no experimente cambios muy bruscos una vez que haya participado en una gran cantidad de enfrentamientos, sino que se estabilice, cosa que solo vemos reflejada en la gráfica del Tipo 2.

En la Figura 4.10, Figura 4.11, Figura 4.12, que representan las puntuaciones correspondientes a algunas de las distintas categorías que existen en *¡Acepta el reto!*, podemos ver que lo mencionado para la puntuación global se cumple para algunas de las puntuaciones por categoría: las puntuaciones para la mayoría de categorías con el Tipo 2 son mayores que las del Tipo 3, el cual tiene las puntuaciones más bajas de los tres.

Cabe resaltar que, para las puntuaciones por categorías, el Tipo 1 es el que presenta las puntuaciones más altas. Creemos que esto se debe a que la puntuación de una categoría solo se actualiza si el problema que se intenta

resolver pertenece a esa categoría, de lo contrario queda intacta. Esto, unido al hecho de que el Tipo 1 no tiene en cuenta el número de envíos para actualizar la puntuación, produce unas puntuaciones más altas que la de los otros sistemas ELO.

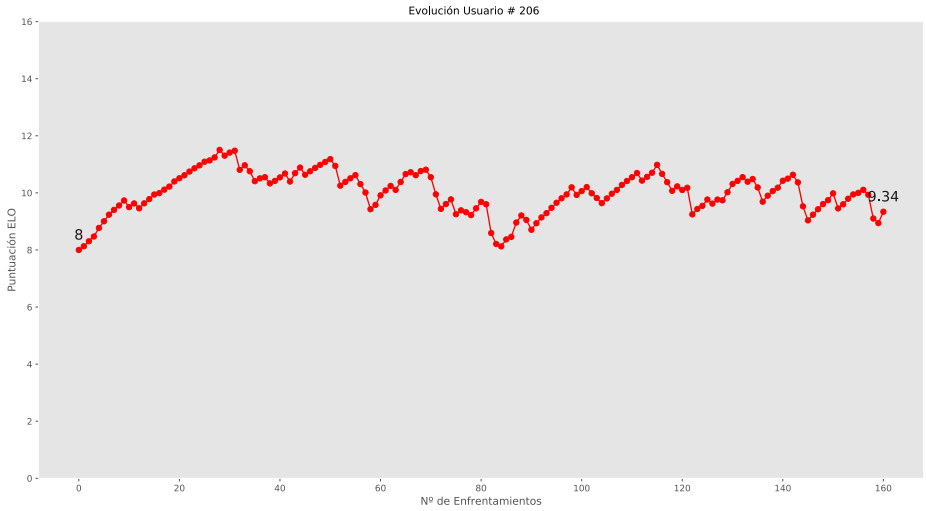


Figura 4.7: Evolución del ELO del usuario 206 usando Tipo 1

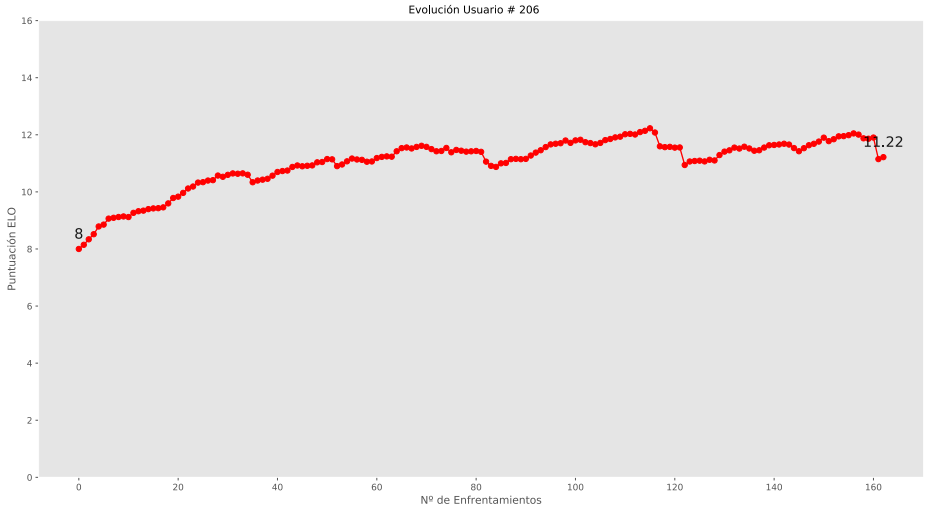


Figura 4.8: Evolución del ELO del usuario 206 usando Tipo 2

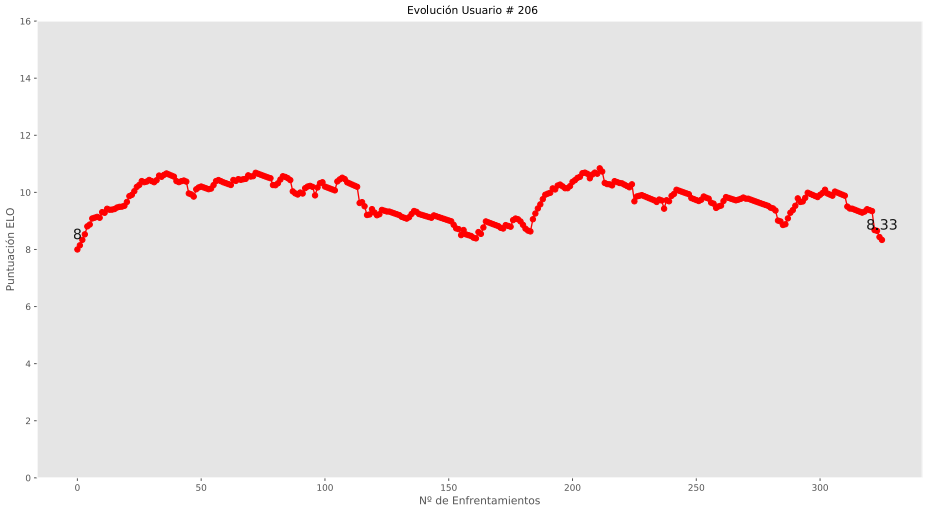


Figura 4.9: Evolución del ELO del usuario 206 usando Tipo 3

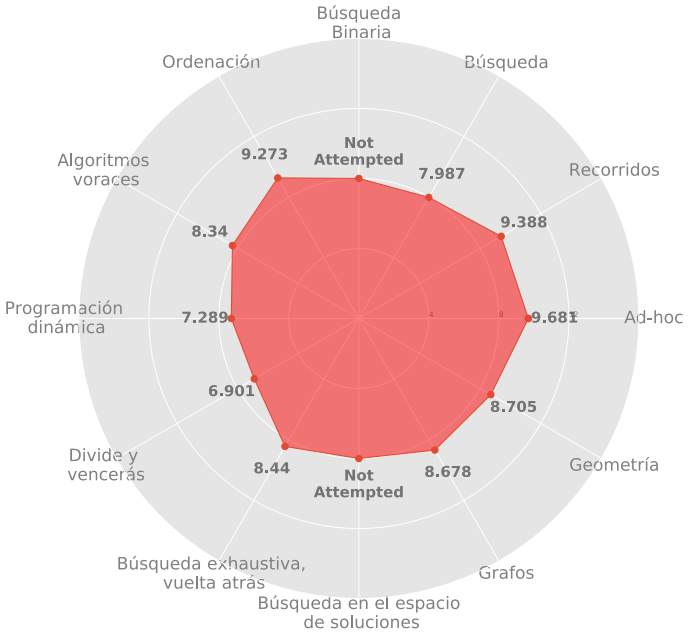


Figura 4.10: Evolución de las categorías del usuario 206 usando Tipo 1

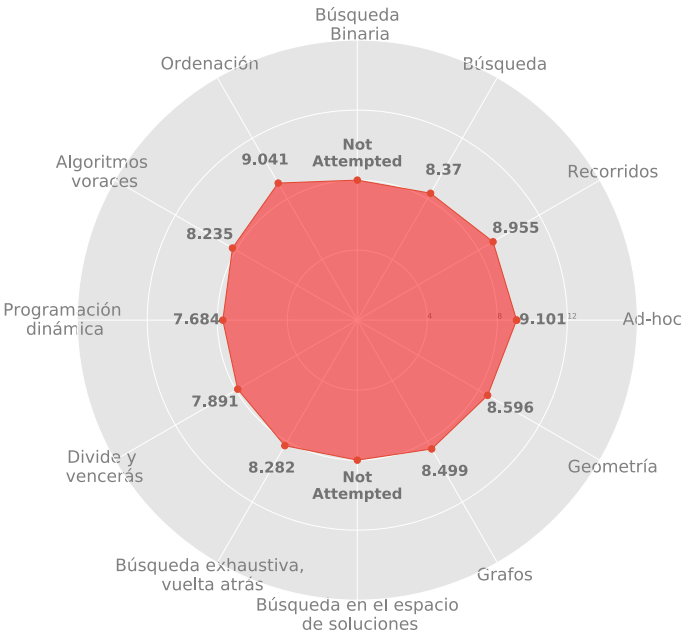


Figura 4.11: Evolución de las categorías del usuario 206 usando Tipo 2

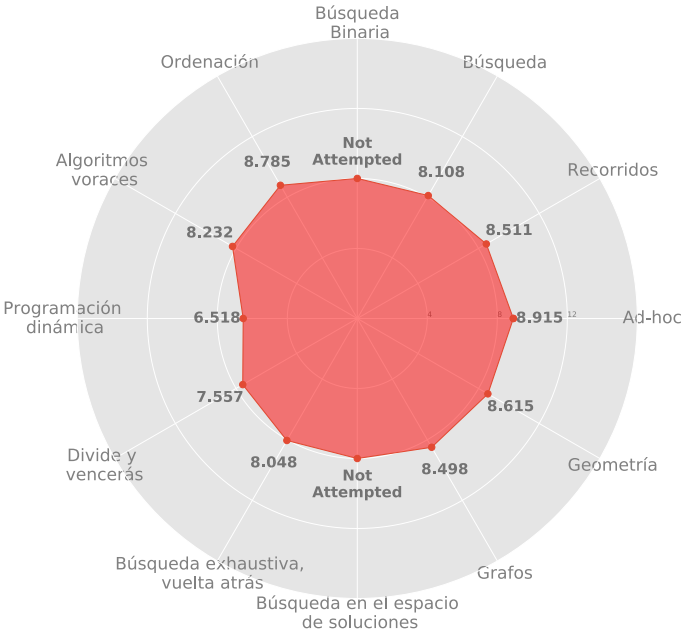


Figura 4.12: Evolución de las categorías del usuario 206 usando Tipo 3

4.2.3. Diferencias en la distribución de puntuaciones de problemas

En esta sección explicaremos las principales diferencias encontradas tras calcular las puntuaciones de todos los problemas usando los tres sistemas de puntuación ELO.

La distribución de puntuaciones del Tipo 1 (Figura 4.13), Tipo 2 (Figura 4.14) y Tipo 3 (Figura 4.15) son muy parecidas, pero presentan algunas diferencias que es necesario tener en cuenta:

- El porcentaje de problemas con una puntuación comprendida entre 7 y 8 es de: 20.42 % en la Figura 4.13, 25.26 % en la Figura 4.14, y 23.18 % en la Figura 4.15.
- En la distribución del sistema ELO de tipo 2 se puede observar que el 80 % de los problemas tienen una puntuación por debajo de 8, un 3 % por encima de las otras distribuciones. Esta pequeña diferencia se puede deber a que el sistema ELO de tipo 2 favorece a los usuarios que no abandonan problemas, puesto que no se les penaliza si sobrepasa cierto número de intentos, como lo hace el de tipo 3.
- La Figura 4.13 del Tipo 1 muestra que existen problemas cuyas puntuaciones superan el 11, mientras que en las otras dos distribuciones ninguno de los problemas supera este valor.

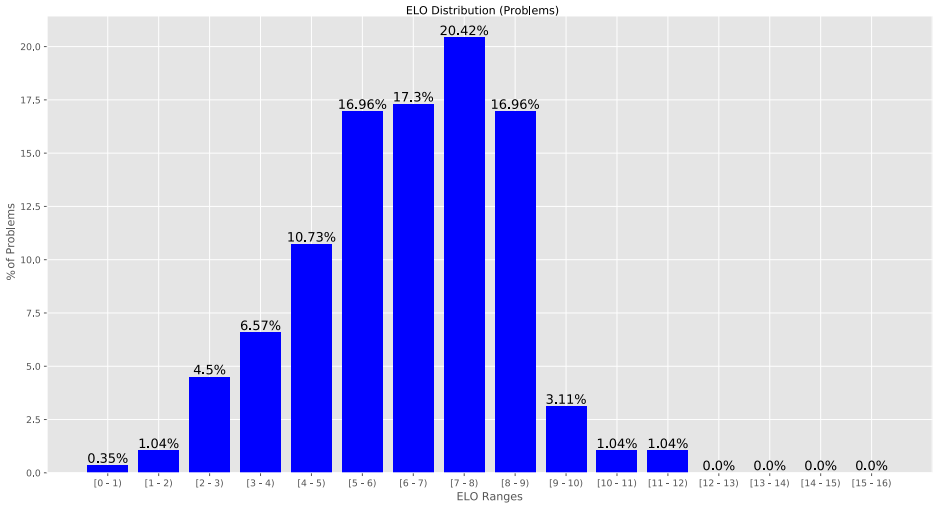


Figura 4.13: Distribución ELO de problemas tras aplicar el sistema ELO de tipo 1 al conjunto de entrenamiento.

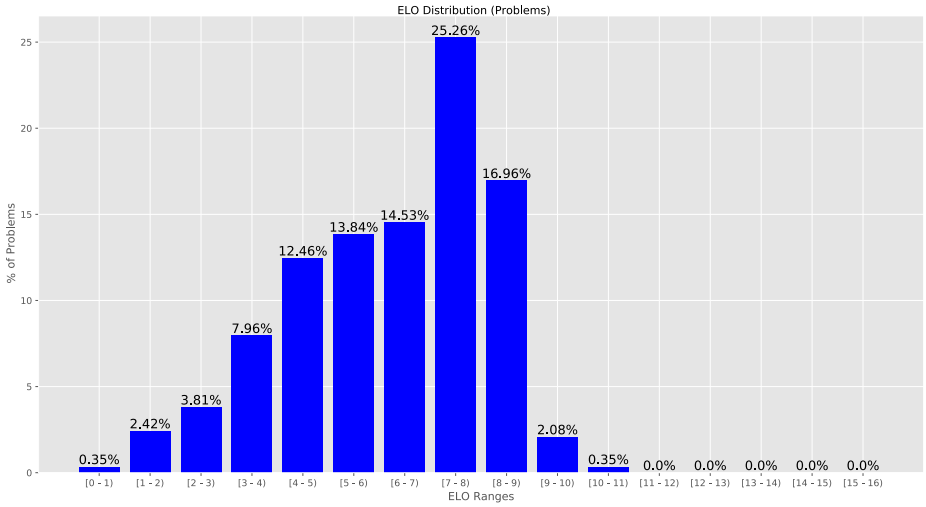


Figura 4.14: Distribución ELO de problemas tras aplicar el sistema ELO de tipo 2 al conjunto de entrenamiento.

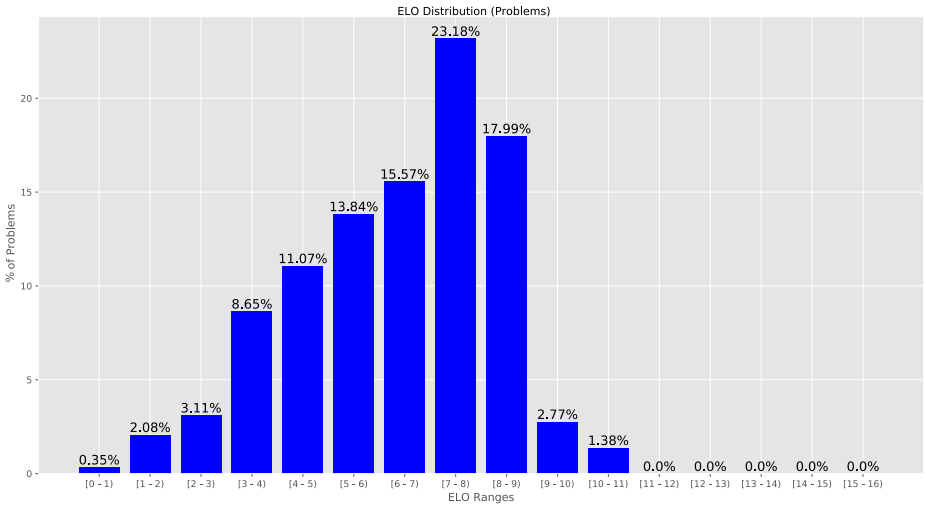


Figura 4.15: Distribución ELO de problemas tras aplicar el sistema ELO de tipo 3 al conjunto de entrenamiento.

4.2.4. Diferencias en la evolución de la puntuación de los problemas

Una vez calculadas las puntuaciones de los problemas, se estudiaron las gráficas de evolución generadas y se decidió escoger el problema con ID 258 para comparar y estudiar las diferencias de aplicar los distintos sistemas de puntuación a un mismo problema. Este problema destaca por ser uno de los que más envíos ha recibido, y por tanto su puntuación ha experimentado numerosos cambios a lo largo del tiempo.

Si observamos las 3 gráficas correspondientes con la evolución de la puntuación del problema en los distintos tipos (Figura 4.16, Figura 4.17, Figura 4.18), podemos observar que:

- La evolución para el tipo 2 (Figura 4.17) y tipo 3 (Figura 4.18) son más estables que las del tipo 1 (Figura 4.16), además de que ambas obtienen una puntuación similar al final. Esto puede deberse a que ambas tienen en cuenta el número de intentos a la hora de calcular el factor K, lo que implicaría que la K obtenida en los enfrentamientos en los que el número de intentos es grande resultaría en perdidas de puntuación pequeñas.
- La evolución para el tipo 1 (Figura 4.16) oscila más que las otras 2, y también obtiene la mayor puntuación de las 3. Al igual que en el caso anterior, esto es el resultado de que el valor de K para este tipo suele ser mayor que en los otros 2 tipos, debido a que no hay que dividirlo entre el número de intentos. Esto se traduce en ganancias y perdidas de puntuación mayores.

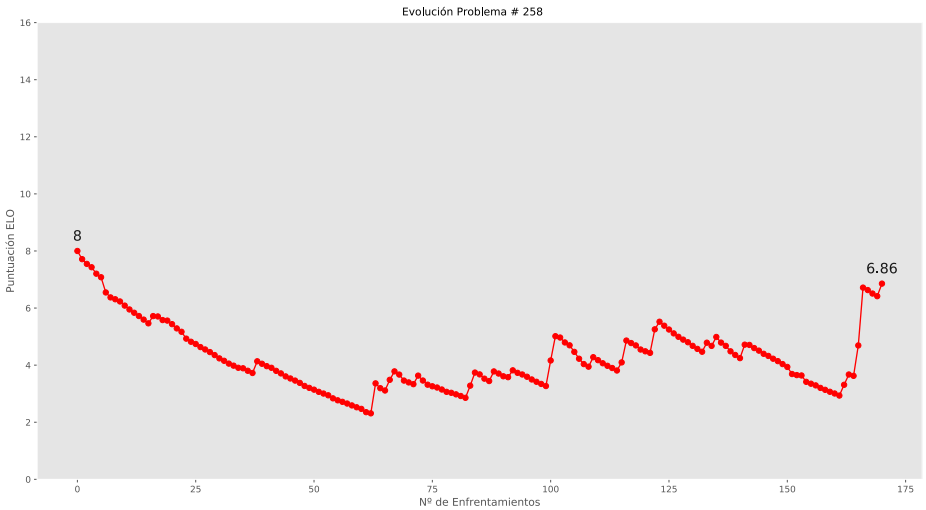


Figura 4.16: Evolución del ELO del problema 258 usando Tipo 1

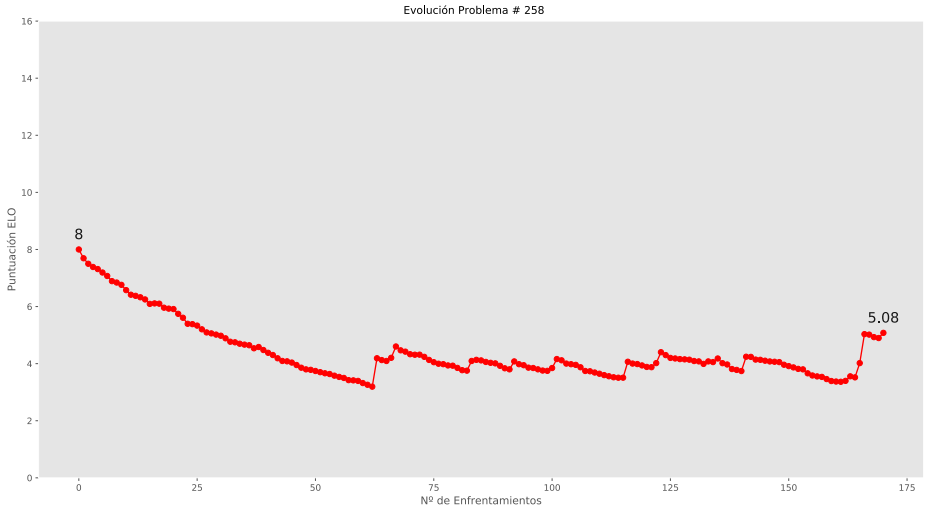


Figura 4.17: Evolución del ELO del problema 258 usando Tipo 2

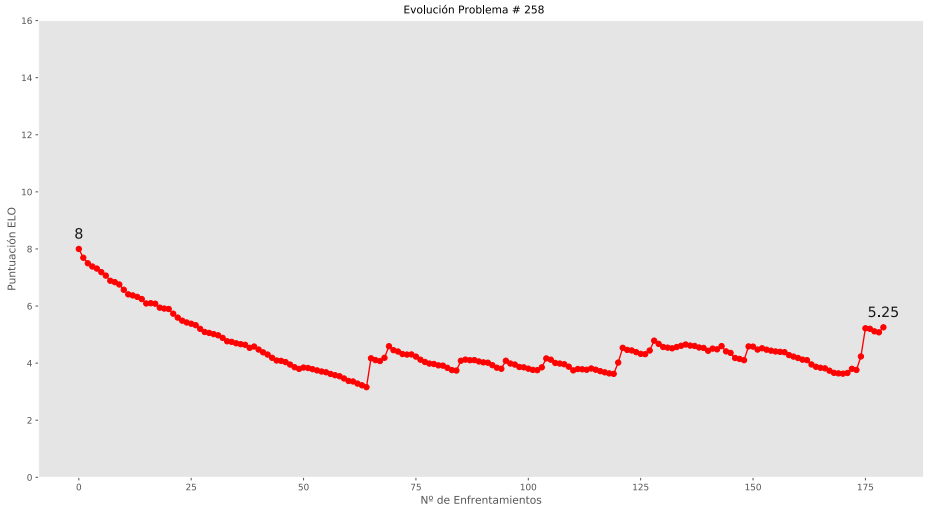


Figura 4.18: Evolución del ELO del problema 258 usando Tipo 3

Al igual que con la evolución de los usuarios, podemos ver que usando el sistema ELO de Tipo 1 se obtiene una gráfica en la que la puntuación del problema oscila de una manera más brusca que la del Tipo 2 o 3. Esto nos hace creer que, dado que la puntuación del Tipo 1 no parece estabilizarse, no es el tipo más adecuado para medir la dificultad de un problema.

4.3. Análisis de la precisión de las recomendaciones

El objetivo de este análisis es el de conocer la calidad de las recomendaciones. Para ello se recomendarán problemas a un conjunto de usuarios y se observarán cuantos de ellos consiguen resolver o al menos intentar alguno de los problemas, y se usará la métrica One-hit para calcular la precisión de las recomendaciones.

Es necesario mencionar que el número de categorías con las que cuenta *¡Acepta el reto!* es muy grande y hacer un seguimiento de todas ellas para cada usuario resultaría en un aumento de la complejidad con el que no se había contado en un principio. Por ello decidimos que, para poner a prueba nuestras suposiciones, era necesario escoger un número reducido de categorías y hacer un seguimiento de ellas para cada usuario.

Las categorías que se escogieron fueron doce: Ad-hoc, Recorridos, Búsqueda, Búsqueda binaria, Ordenación, Algoritmos voraces, Programación dinámica, Divide y vencerás, Búsqueda exhaustiva / vuelta atrás, Búsqueda en el espacio de soluciones, Grafos y Geometría.

Como ya se ha explicado antes, el sistema de recomendación desarrollado se encarga de escoger los problemas cuyas puntuaciones sean similares a las del usuarios, y recomendárselos. Esta operación se realiza tanto para la puntuación global del usuario como para las puntuaciones asociadas a las 12 categorías escogidas.

Para tener una idea de la precisión de este recomendador se decidió hacer uso del conjunto de datos de entrenamiento y el de prueba:

1. El primer paso es el de calcular las puntuaciones de los usuarios y problemas que aparecen en el conjunto de entrenamiento.
2. A partir de estas puntuaciones, el recomendador empezará a generar recomendaciones para los usuarios entrenados. Se recomendarán un máximo de tres problemas por categoría.
3. Utilizando los envíos del conjunto de prueba, se comprobará si los usuarios que aparecen en el conjunto de entrenamiento tienden a resolver o intentar los problemas que se le habrían recomendado.

Debido al alto número de usuarios inactivos con los que contaba el conjunto de entrenamiento, se decidió realizar el análisis con aquellos usuarios que hayan resuelto un mínimo número de problemas en el conjunto de entrenamiento y en el conjunto de pruebas.

Para calcular la precisión se optó por utilizar la métrica de evaluación One-hit (1-hit). Esta métrica considerará como éxito los casos en los que un usuario acaba interactuando con al menos uno de los problemas que se le han recomendado, y como fracaso el caso contrario. El resultado final será la división de la tasa de 1-hit entre el número de recomendaciones realizadas.

Con esto en cuenta, decidimos calcular el valor 1-hit dos veces por cada tipo de sistema ELO desarrollado:

- El primer cálculo considerará como éxito los casos en los que el usuario ha intentado (no necesariamente resuelto) al menos uno de los problemas recomendados. Esto significaría que el usuario habría mostrado algún interés en resolver uno de los problemas recomendados.
- El segundo cálculo solo considerará como éxito los casos en los que el usuario haya conseguido resolver al menos uno de los problema recomendados. Este hecho nos indicaría que la dificultad del problema era la adecuada para el nivel de habilidad del usuario.

4.3.1. Resultados

La Tabla 4.2 muestra los valores 1-hit obtenidos para las recomendaciones generadas usando los tres tipos de sistema ELO desarrollados, en los que se resalta con negrita los valores más altos en cada caso.

La primera observación que se puede hacer es que los valores 1-hit son mayores cuanto más problemas haya resuelto un usuario, o lo que es lo mismo, cuanto más cerca este el usuario de alcanzar su nivel de habilidad real. Estos valores son ligeramente mayores cuando el cálculo del 1-hit se hace tomando los intentos a problemas recomendados como éxito.

El valor 1-hit más alto (0.565) corresponde al tipo 2, al que le sigue el del tipo 3 (0.528) y el tipo 1 (0.491). Estos valores, que se corresponden con los usuarios que han intentado al menos uno de los problemas recomendados, indican que más de la mitad de estos usuarios se habrían decantado por intentar resolver alguno de los problemas recomendados. Para los usuarios que han resuelto al menos un problema recomendado, observamos que los valores más altos para los tres tipos sobrepasan el 0.4, lo que significa que más de un 40 % de estos usuarios intentarían resolver los problemas que se le recomiendan. Cabe mencionar que, aunque se haya marcado con negrita el 1-hit 0.408 para el Tipo 1 con usuarios que han resuelto al menos un problema recomendado, la diferencia con el siguiente 1-hit más alto, 0.407 (mínimo de 10 problemas en el conj. de entrenamiento y 10 en el de prueba) es de solo una milésima, por lo que ambos son igual de significativos.

Mínimo de probs. resueltos en el conj. de entrenamiento	Mínimo de probs. resueltos en el conj. de prueba	Al menos uno intentado			Al menos uno resuelto		
		Tipo 1	**Tipo 2**	**Tipo 3**	**Tipo 1**	**Tipo 2**	**Tipo 3**
1	1	0.167	0.165	0.153	0.113	0.12	0.095
	3	0.237	0.24	0.216	0.175	0.178	0.142
	5	0.288	0.304	0.274	0.214	0.224	0.181
	8	0.35	0.401	0.359	0.29	0.309	0.249
	10	0.376	0.444	0.402	0.323	0.344	0.286
3	1	0.204	0.204	0.186	0.138	0.147	0.12
	3	0.27	0.277	0.251	0.199	0.205	0.166
	5	0.322	0.347	0.314	0.238	0.255	0.209
	8	0.39	0.452	0.401	0.316	0.345	0.282
	10	0.41	0.494	0.442	0.346	0.378	0.321
5	1	0.224	0.232	0.213	0.151	0.174	0.143
	3	0.294	0.313	0.286	0.214	0.242	0.198
	5	0.341	0.375	0.337	0.255	0.288	0.236
	8	0.414	0.484	0.433	0.331	0.382	0.312
	10	0.429	0.521	0.471	0.357	0.414	0.35
8	1	0.275	0.279	0.256	0.198	0.214	0.187
	3	0.354	0.364	0.323	0.263	0.283	0.242
	5	0.404	0.428	0.38	0.307	0.331	0.283
	8	0.47	0.523	0.47	0.379	0.417	0.356
	10	0.483	0.558	0.5	**0.408**	0.45	0.392
10	1	0.298	0.303	0.289	0.216	0.229	0.216
	3	0.375	0.387	0.357	0.28	0.298	0.274
	5	0.427	0.448	0.413	0.322	0.343	0.315
	8	0.479	0.53	0.496	0.385	0.419	0.385
	10	**0.491**	**0.565**	**0.528**	0.407	**0.454**	**0.417**

Tabla 4.2: Comparación de valores 1-hit de las recomendaciones hechas a usuarios entrenados con los tres Sistemas ELO.

4.4. Conclusiones

Desde el principio de este Trabajo de Fin de Grado, el objetivo que se persiguió era el de crear un sistema de recomendación que fuera fiable, eficaz y preciso, capaz de procesar grandes cantidades de datos y cuyo mecanismo interno sea fácil de entender y adaptar a otras necesidades.

Con los resultados obtenidos tras las distintas pruebas realizadas podemos concluir que:

- Los variaciones del sistema ELO desarrollados cumplen con su objetivo de aproximar el nivel de habilidad de los usuarios y de dificultad de los problemas con precisión.
- De los tipos de sistemas ELO probados destaca el tipo 2, el cual no penaliza a los usuarios por enviar muchas soluciones, pero tampoco las ignora a la hora de calcular el factor K, el cual determina cuanto puede cambiar la puntuación.
- Cuantos más enfrentamientos tengan los usuarios o problemas, más se acercarán a sus niveles de habilidad o dificultad reales.
- El sistema de recomendación basado en la similitud de puntuaciones ELO de usuarios y problemas es lo bastante bueno como para ser de utilidad a más del 50 % de usuarios que lo utilicen.
- La clasificación de los problemas a través de categorías es imprescindible si se quiere conseguir recomendaciones personalizadas que tengan en cuenta las preferencias del usuario.

Capítulo 5

Desarrollo de Aplicación Web

"Caminando en línea recta no puede uno llegar muy lejos"
— El Principito, Antoine de Saint-Exupèry (1943)

Parte de este trabajo es la construcción de una Aplicación Web que sirva como plataforma de pruebas en la que un investigador pueda acceder a la información relacionada con usuarios y problemas de la base de datos de *¡Acepta el reto!* y otras estadísticas, así como al sistema de recomendación y puntuación implementado. Esta aplicación le servirá para experimentar con los datos y estudiar el comportamiento de estos en diferentes contextos, con diferentes configuraciones del sistema de puntuación y de recomendación. Para ello, la aplicación le permitirá crear nuevos usuarios y problemas, simular envíos de soluciones y escoger como se calculan las puntuaciones.

La aplicación ha sido implementada usando Python como lenguaje principal. Gracias a ello pudimos utilizar frameworks que nos permitieron agilizar su desarrollo e implementación, como Flask, el cual es un componente principal del funcionamiento de la aplicación. Con una interfaz clara, la aplicación permite acceder a numerosas vistas en donde nos encontramos con información sobre varios aspectos de *¡Acepta el reto!* y sus sistemas de puntuación y recomendación. La parte que compone la lógica del sistema de puntuación y las recomendaciones fueron desarrollados en Python; mientras que la parte de la interfaz ha sido desarrollada con lenguaje HTML, CSS y JavaScript.

La aplicación ha sido encapsulada con la ayuda de Docker[1] de manera que el despliegue, al ser totalmente automatizado, sea mucho más liviano, portable y sencillo, pudiéndolo probar en cualquier maquina que tenga instalado Docker con tan solo la ejecución de unos comandos, sin la necesidad de añadir bibliotecas o dependencias.

[1] https://www.docker.com/

El código fuente de la Aplicación Web y su despliegue está disponible en GitHub en: `https://github.com/jlsomeg/TFG-ELORecommender`.

5.1. Funcionalidades

La funcionalidad de la aplicación esta construida desde la perspectiva de un investigador donde, de una manera intuitiva y visual, puede consultar, añadir y simular distintos elementos de la base de datos de *¡Acepta el reto!*, lo que otorga gran libertad a la hora de experimentar con los datos y analizar los resultados.

5.1.1. Perfil Usuario

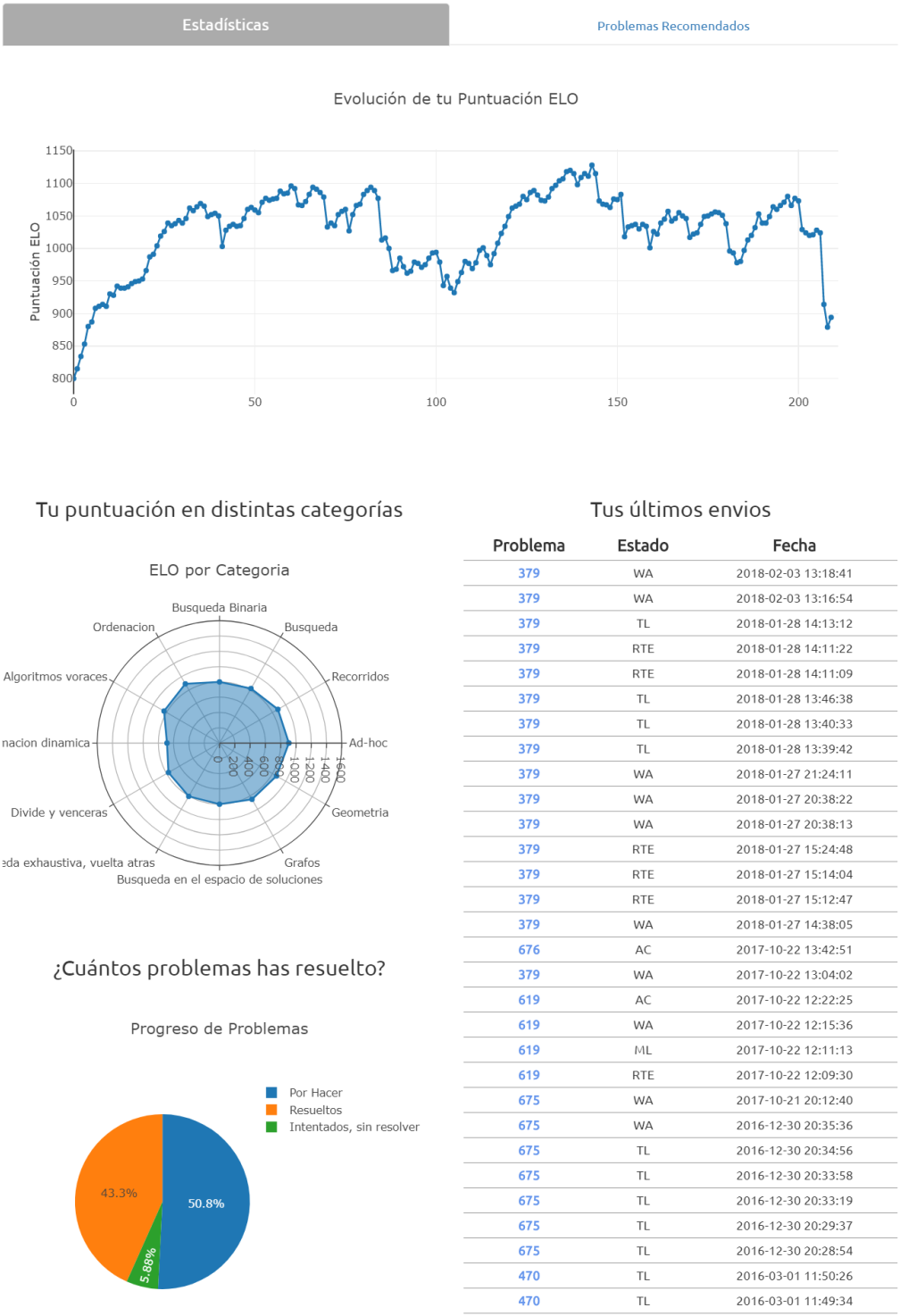
Esta vista se divide en dos pestañas: la primera muestra datos relacionados con la actividad y evolución del usuario, mientras que la segunda muestra una tabla con los problemas recomendados por el sistema.

La primera pestaña (Figura 5.1) cuenta con:

- Una gráfica que muestra la evolución de la puntuación ELO del usuario desde el momento en que se unió a la plataforma.
- Una tabla que muestra los últimos 30 envíos realizados, donde se indica el estado y fecha del envío.
- Un gráfico de radar que muestra las puntuaciones ELO del usuario para 12 categorías de problemas seleccionados previamente.
- Un gráfico de radar que muestra el porcentaje de problemas resueltos, intentados y los que aún no ha intentado.

La segunda pestaña (Figura 5.2) muestra una tabla que lista los problemas recomendados. Esta tabla contendrá dos tipos de recomendaciones: la recomendación de tipo "global", la cual hará uso de la puntuación ELO general para generar las recomendaciones, y las recomendaciones por categorías, que solo se generan si el usuario ha interactuado con problemas pertenecientes a algunas de las doce categorías mostradas en el gráfico de radar y que, para cada categoría explorada, generará una recomendación haciendo uso de la puntuación ELO que el usuario tiene asignada para tal categoría.

Es necesario aclarar que, dado el alto número de categorías de problemas, se optó por escoger doce categorías y hacer un seguimiento de ellas para cada usuario. Esto no solo simplificaba la presentación de la información, sino que también evitaba almacenar información de categorías que son raramente exploradas.



Problema	Estado	Fecha
379	WA	2018-02-03 13:18:41
379	WA	2018-02-03 13:16:54
379	TL	2018-01-28 14:13:12
379	RTE	2018-01-28 14:11:22
379	RTE	2018-01-28 14:11:09
379	TL	2018-01-28 13:46:38
379	TL	2018-01-28 13:40:33
379	TL	2018-01-28 13:39:42
379	WA	2018-01-27 21:24:11
379	WA	2018-01-27 20:38:22
379	WA	2018-01-27 20:38:13
379	RTE	2018-01-27 15:24:48
379	RTE	2018-01-27 15:14:04
379	RTE	2018-01-27 15:12:47
379	WA	2018-01-27 14:38:05
676	AC	2017-10-22 13:42:51
379	WA	2017-10-22 13:04:02
619	AC	2017-10-22 12:22:25
619	WA	2017-10-22 12:15:36
619	ML	2017-10-22 12:11:13
619	RTE	2017-10-22 12:09:30
675	WA	2017-10-21 20:12:40
675	WA	2016-12-30 20:35:36
675	TL	2016-12-30 20:34:56
675	TL	2016-12-30 20:33:58
675	TL	2016-12-30 20:33:19
675	TL	2016-12-30 20:29:37
675	TL	2016-12-30 20:28:54
470	TL	2016-03-01 11:50:26
470	TL	2016-03-01 11:49:34

Figura 5.1: Captura de Perfil de Usuario: Estadísticas

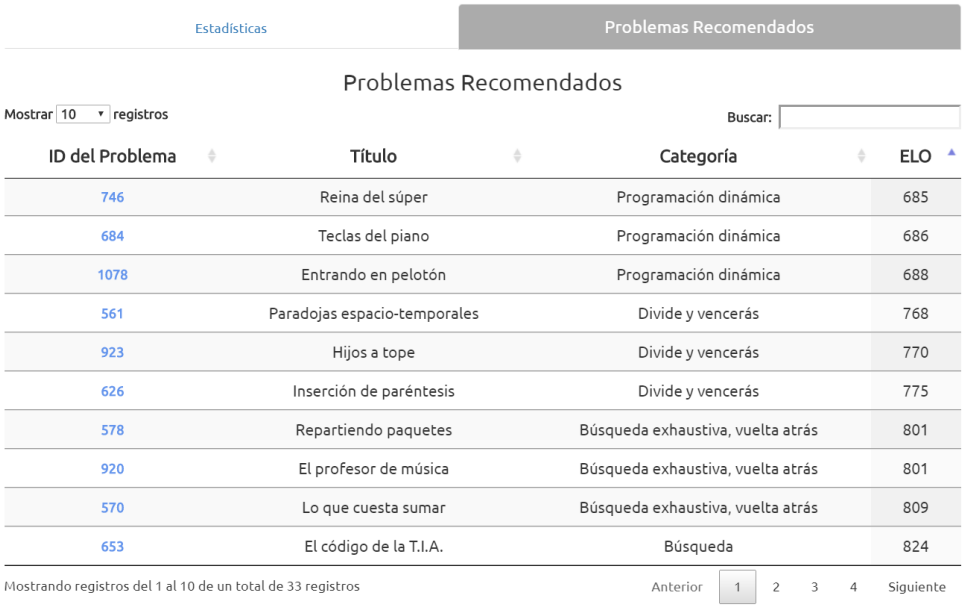


Figura 5.2: Captura de Perfil de Usuario: Recomendaciones

5.1.2. Insertar Usuario

Esta vista (Figura 5.3) que permite insertar un nuevo usuario a la base de datos siempre y cuando se le proporcione un ID que no exista con anterioridad. También se puede cambiar el valor de su puntuación ELO inicial, aunque si este campo se deja vacío se utilizará 8 como valor por defecto.

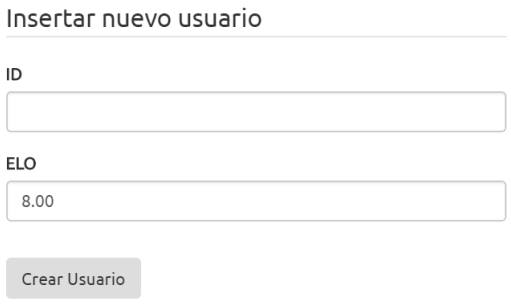


Figura 5.3: Captura de Insertar Usuario

5.1.3. Lista de Usuarios

Muestra una tabla (Figura 5.4) con datos de todos los usuarios de *¡Acepta el reto!*. Los datos mostrados para cada usuario son su identificador, su puntuación ELO, el número de problemas que ha intentado resolver y el número de problemas que ha resuelto. Esta tabla cuenta con la opción de reordenar los resultados a partir de alguno de los cuatro atributos mencionados, de limitar el número de resultados por página, y de realizar búsquedas usando palabras clave.

Los IDs de la primera columna sirven como hipervínculos que referencian al perfil de los usuarios, los cuales cuentan con mucha más información que la ofrecida en esta vista.

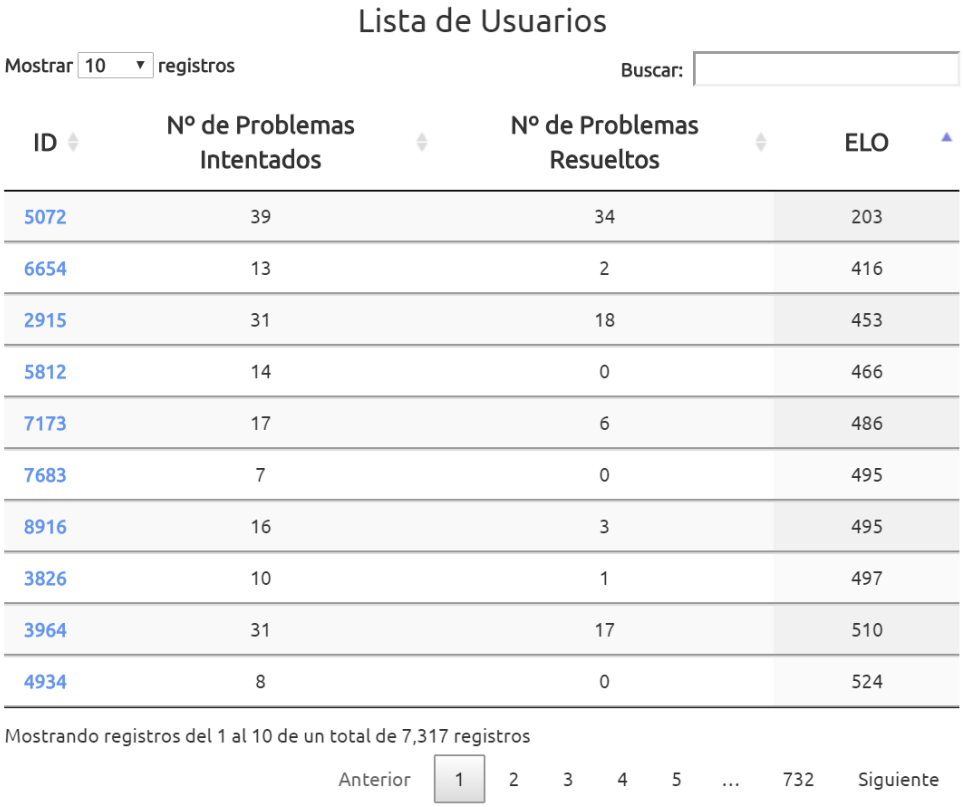


Figura 5.4: Captura de Lista de Usuarios

5.1.4. Perfil Problema

Esta vista (Figura 5.5) es similar a la del perfil de usuario, aunque no dispone de más de una pestaña. Cuenta con cinco elementos:

- Una gráfica que nos permite observar la evolución de la puntuación ELO del problema desde el momento en que se unió a la plataforma.
- Una gráfica que muestra el porcentaje de usuarios que han necesitado X intentos para resolver el problema.
- Un gráfico circular que muestra el porcentaje de usuarios que lo han resuelto y de aquellos que lo han intentado pero aún no lo han resuelto.
- Un gráfico circular que muestra los lenguajes de programación más utilizados a la hora de intentar resolver el problema.
- Una tabla que muestra los últimos 30 envíos realizados, donde se indica el estado y fecha del envío.

5.1.5. Lista de Problemas

Muestra una tabla (Figura 5.6) con datos de todos los problemas de *¡Acepta el reto!*. Los datos mostrados son: el ID del problema, su título, las categorías a las que pertenece, el número de envíos que han intentado resolver el problema, su puntuación ELO, el porcentaje de envíos correctos con respecto al total de envíos y el porcentaje de usuarios que lo han resuelto con respecto al total que lo ha intentado. Esta tabla cuenta con la opción de reordenar los resultados a partir de alguno de los atributos mencionados, de limitar el número de resultados por página, y de realizar búsquedas usando palabras clave.

Los IDs de la primera columna sirven como hipervínculos que referencian al perfil de los problemas.

5.1.6. Insertar Problema

Esta vista (Figura 5.7) permite insertar un nuevo problema a la base de datos. Se le debe proporcionar un ID que no exista con anterioridad, un título y se debe escoger una o varias categorías a las que se quiera que pertenezca el problema. También se puede cambiar el valor de su puntuación ELO inicial, aunque si este campo se deja vacío se utilizará el valor por defecto (8).

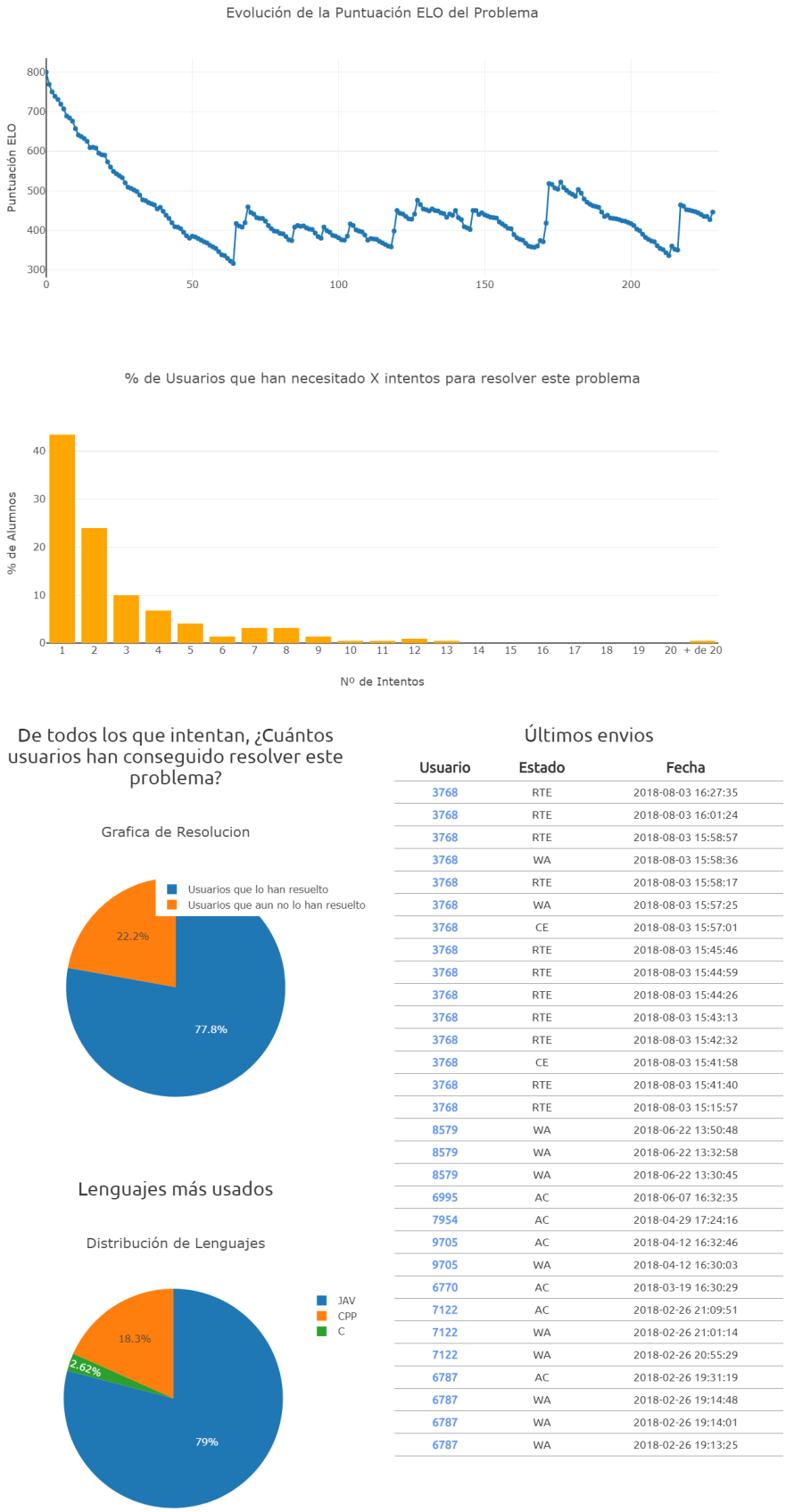


Figura 5.5: Captura de Perfil de Problema

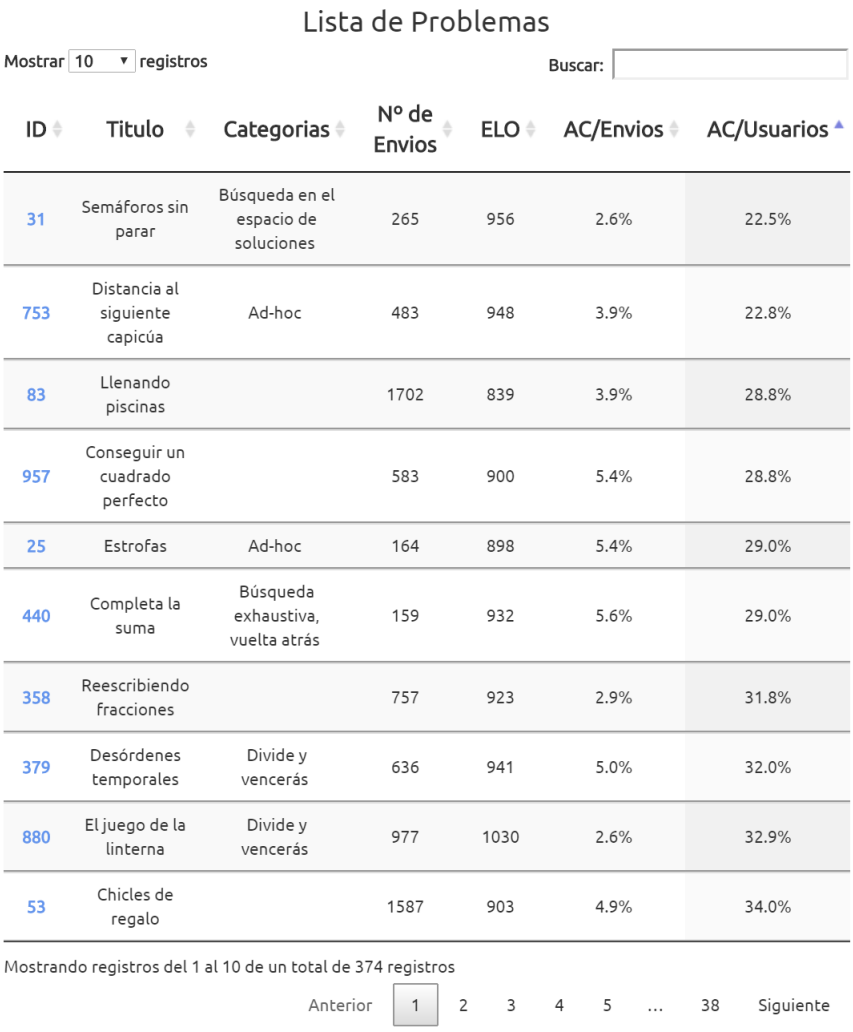


Figura 5.6: Captura de Lista de Problemas

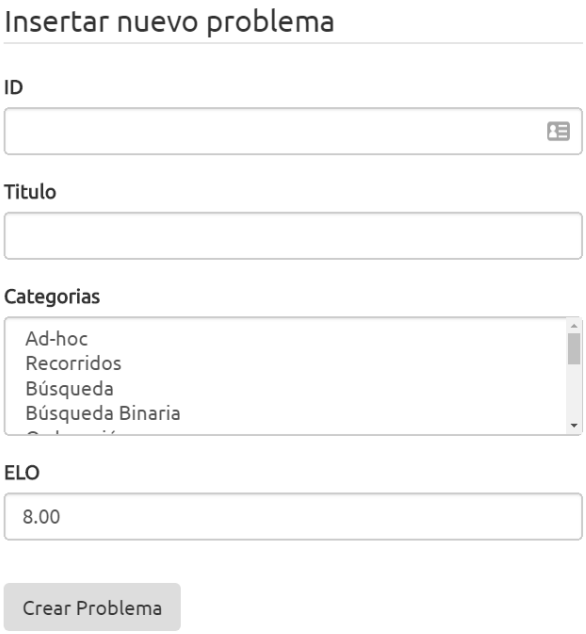


Figura 5.7: Captura de Insertar Problema

5.1.7. Estadísticas Globales

Esta vista (Figura 5.8) nos permite consultar las estadísticas globales de la base de datos a través de distintas gráficas:

- Un histograma que muestra el número de envíos realizados cada mes desde la concepción de *¡Acepta el reto!* hasta la fecha del último envío incluido en la base de datos.
- Dos histogramas que muestran la distribución de puntuaciones ELO de los usuarios (izquierda) y de los problemas (derecha).
- Un histograma que muestra la distribución de usuarios en función del número de intentos necesitados para resolver un problema.

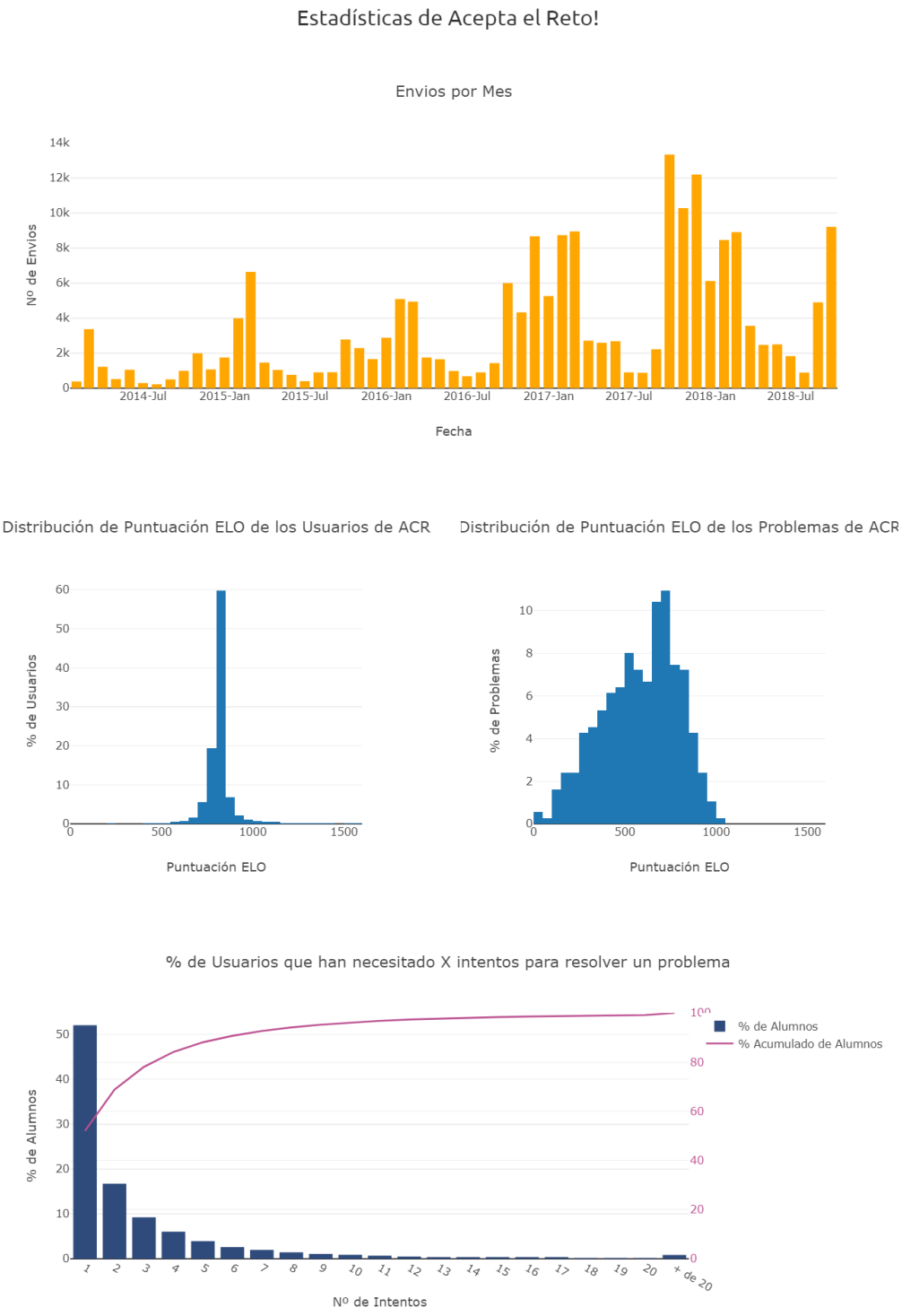


Figura 5.8: Captura de Estadísticas Globales

5.1.8. Recomendación para Novatos

En esta vista (Figura 5.9) se mostrará una lista de problemas de *¡Acepta el reto!* donde, para cada una de las 12 categorías de las que se está haciendo el seguimiento, se recomendaran los 5 problemas con la puntuación más baja. Aunque se puede interpretar como una recomendación, la función principal de esta vista es mostrar al investigador los problemas más asequibles de la base de datos en el caso en el que se necesite guiar y dirigir a usuarios primerizos hacia problemas que, al contar con puntuaciones tan bajas, pueden resultar más fáciles de resolver.

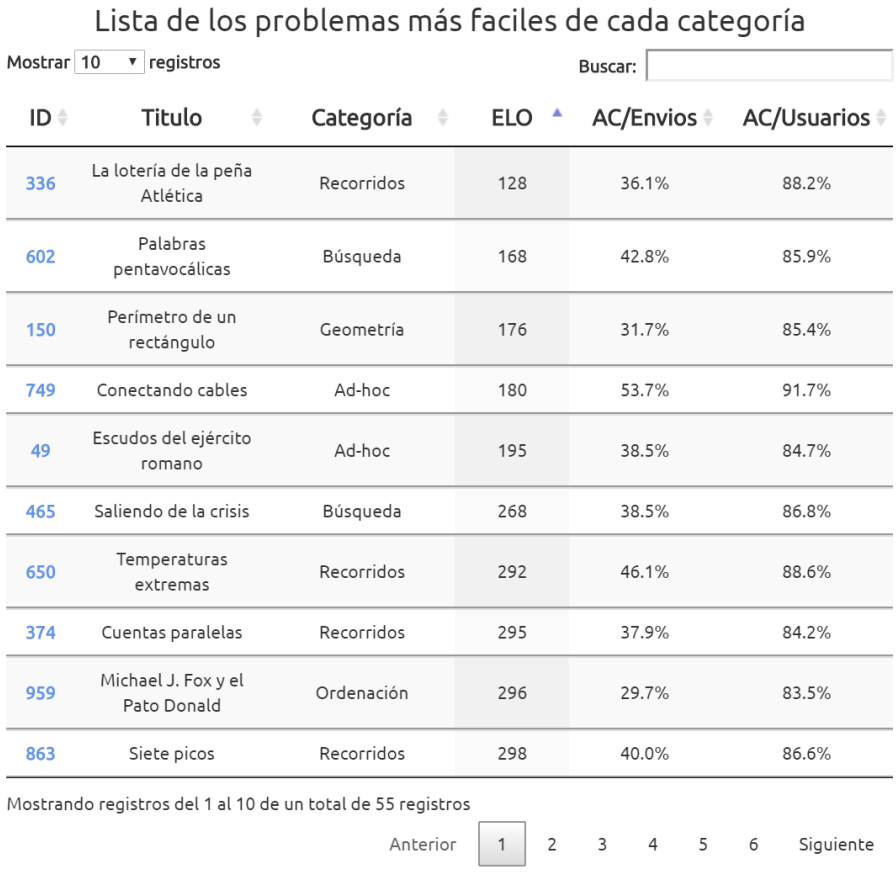


Figura 5.9: Captura de Recomendación para Novatos

5.1.9. Enviar Solución

Permite simular el envío de una solución por parte de un usuario a un problema, e insertará una nueva fila en la tabla de envíos. Si el envío cumple una de las condiciones que provocan el recalculo de las puntuaciones, estas se actualizaran acordemente. (Figura 5.10)

Para cada simulación se debe proporcionar los IDs de un usuario y problema existentes, y se debe cumplir que el usuario no haya resuelto el problema con algún envío anterior. Otros parámetros que se pueden modificar son: el lenguaje del código y el veredicto que se desea obtener.

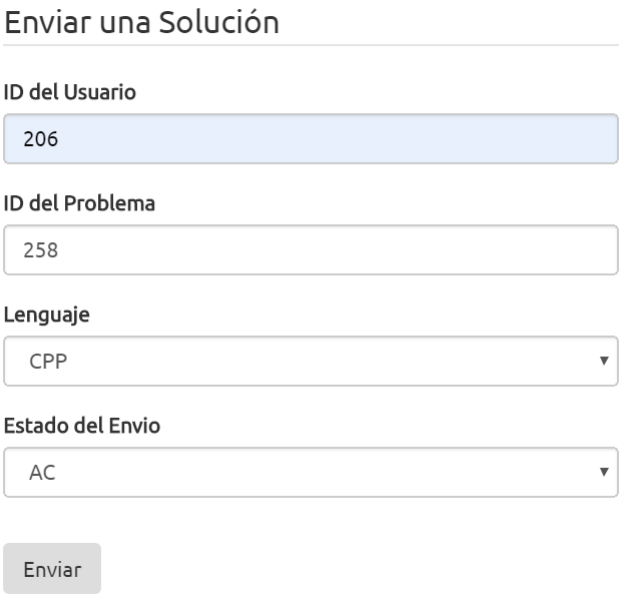


Figura 5.10: Captura de Enviar Solución

5.1.10. Cambiar ELO

Esta opción (Figura 5.11) permitirá al investigador seleccionar cual de los tres tipos de sistemas ELO desarrollados en este TFG se deberá aplicar para calcular las nuevas puntuaciones. La elección de un nuevo tipo afectará a toda la base de datos, puesto que se llevará a cabo un recalculo de las puntuaciones de usuarios y problemas desde el primer envío. El tiempo que tarde esta operación dependerá de la maquina que lo ejecute, pero suele tardar menos de un minuto.

Gracias a esta opción, se podrá experimentar con distintos tipos de escenarios en función de las necesidades y entorno en el que se trabaje.

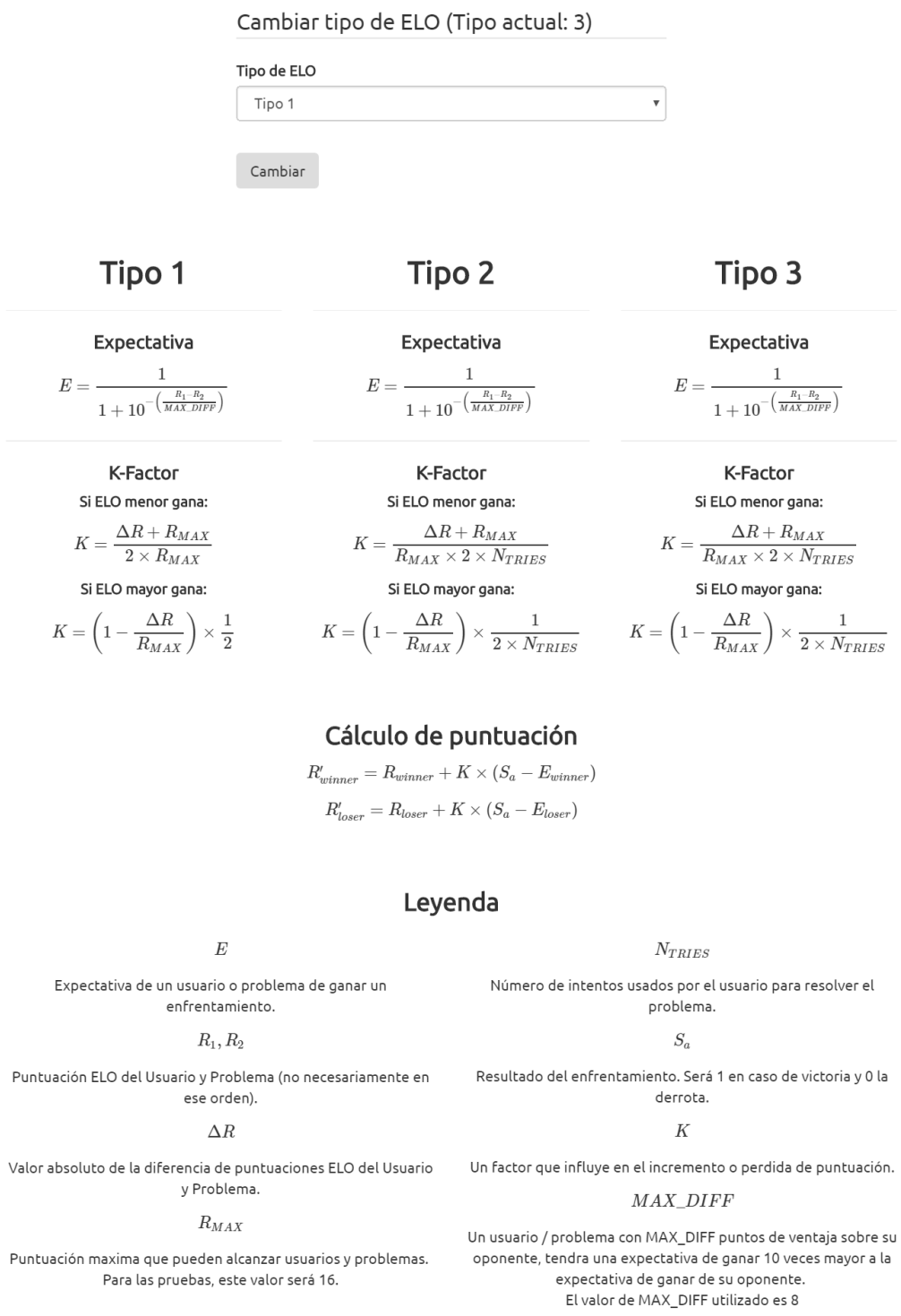


Figura 5.11: Captura de Cambiar ELO

5.2. Utilidad

La motivación detrás de esta aplicación era la de crear una plataforma en la que seamos capaces no solo de visualizar datos relacionados con usuarios y problemas, sino de poder escoger distintas maneras de calcular las puntuaciones y observar como afectaba a las recomendaciones.

La finalidad última de este banco de pruebas es el de poder experimentar con los distintos aspectos del sistema de recomendación y puntuación, el de ver cuanto cambian las puntuaciones si enfrentamos a usuarios de alto nivel con problemas de bajo nivel o viceversa, el de observar cuanto perjudica el enviar muchos envíos a un mismo problema sin resolverlo, etc.

5.3. Arquitectura

El diseño seguido es el de una arquitectura cliente-servidor en la que el cliente realiza peticiones, en este caso muestreo y modificación de la información sobre el recomendador, y el servidor las cumple. La arquitectura de nuestra aplicación tiene tres módulos diferenciados como se muestra aquí (Figura 5.12).

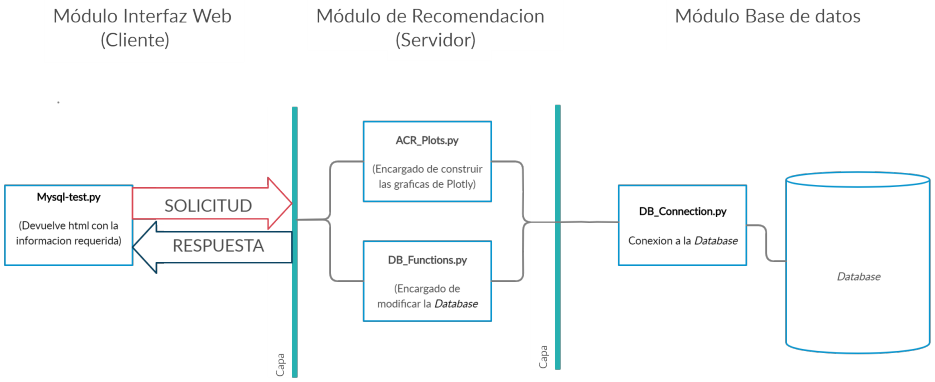


Figura 5.12: Modelo de tres capas de los archivos y paquetes involucrados.

5.3.1. Módulo de recomendación

Este conjunto actúa como el *backend* de la aplicación. En él se encuentran todos las funciones en python que conforman el sistema de recomendación y puntuación. Son los encargados de cargar y procesar datos de la copia de la base de datos de *¡Acepta el reto!* y realizar los cálculos de las puntuaciones de usuarios y problemas.

Contiene la lógica de las recomendaciones y es el encargado de las modificaciones y consultas en la base de datos. Ejecuta las acciones requeridas por el frontend y devuelve la información como estructuras de datos o como bloques de código HTML. El código HTML de las gráficas es generado por el módulo Plotly[2] (Software que provee herramientas sobre gráficos, análisis y estadística). En los archivos ACRPlots.py y DBFunctions.py está contenido la mayoría del código de las funcionalidades de este módulo, y son utilizados para el muestreo de gráficas con Plotly y para el muestreo y modificación de datos en la base de datos, respectivamente.

5.3.2. Interfaz web

Este conjunto actúa como el *frontend* de la aplicación. Nuestra aplicación ha sido implementada usando Flask[3], un microframework para Python que puede usarse para desarrollar aplicaciones web. La aplicación tiene varias rutas programadas a las que se puede acceder directamente a través de la URL. Este módulo es el que realiza las peticiones al módulo de recomendación y usa los datos recibidos para mostrarlos en la interfaz.

Estos datos se encapsulan en código HTML y contendrán la información requerida por la petición gracias al motor de plantillas Jinja2 que usa Flask. Estas plantillas web contienen sintaxis HTML con marcadores de posición intercalados para variables y expresiones (en este caso, expresiones de Python) que son luego reemplazados por sus respectivos valores una vez se compila el servicio. El archivo mysql-test.py contiene el código de este módulo.

5.3.3. Base de datos

Este conjunto es el encargado de realizar la identificación, apertura y cierre de la conexión a la base de datos y es invocada en la gran mayoría de las funciones de nuestra aplicación. El archivo DBConnection.py es donde esta contenido el código de este modulo.

5.4. Adaptabilidad

El diseño de la arquitectura de la aplicación y la moldeabilidad del sistema ELO dota a la aplicación de un potencial de adaptabilidad que permite, una vez calculadas las puntuaciones de usuarios y problemas, establecer un campo de pruebas útil para cualquier tipo de datos que requieran un sistema de recomendación y no solo para el de *¡Acepta el reto!*.

[2]https://plot.ly/python/
[3]http://flask.pocoo.org/

Esto es debido a que se ha construido pensando en usuarios que enfrentan problemas a través de un rango de ELO independientemente de los valores propios de su base de datos. Además fusionándolo con la opción del cambio de ELO podemos personalizar el campo de pruebas que más se adapte al ensayo que queramos realizar e incluso observar la evolución de los mismos usuarios y problemas bajo distintas reglas.

5.5. Despliegue

Docker[4] ofrece un método de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con sus dependencias ya incluidas en varios entornos. A partir de una imagen podremos construir contenedores ligeros y portables, que encapsulan la imagen para poder ser usada de manera dinámica. La virtualización basada en contenedores nos permite en un mismo sistema operativo compartido un aislamiento de los contenedores entre sí. Este servicio facilita la entrega del programa de la aplicación, permitiendo que las aplicaciones se ejecuten en cualquier entorno físico o virtual. Docker permite la programación de la implementación de la aplicación (o varios procesos que interactúan entre si y que constituyen una aplicación) en este entorno de contenedores. Un contenedor de Docker utiliza como motor el sistema operativo que tiene la máquina en la que se ejecuta a diferencia de una maquina virtual.

El contenedor que alberga nuestro servicio esta construido con la última versión de Ubuntu que exista y en el se copian los ficheros de la implementación realizada y diferentes herramientas y dependencias que van a ser utilizadas como pymysql o flask.

Además, nuestro servicio usa la herramienta Docker-Compose, que nos permite definir y correr aplicaciones multi-contenedor, interactuando entre sí, formando un sistema que trabaja con este conjunto como si fuera uno. Esto es un requerimiento debido a que la aplicación se encuentra en un contenedor y la base de datos en otro. La razón de generar esta división es apostar por las aplicaciones distribuidas, es decir, en lugar de encajar todas las piezas en un contenedor, poder mantener, actualizar y escalar los componentes de la aplicación por separado de manera que, en el caso de que se quiera realizar un cambio técnico o de contenido en la BD por ejemplo, o que se quiera utilizar la misma para ser conectada a otro contenedor diferente que no sea el que contiene la aplicación web, pueda hacerse de una manera más cómoda y funcional. Con el archivo "docker-compose.yaml" automatizamos la construcción y conexión de los dos contenedores especificando el puerto,

[4]https://www.docker.com/

y preparándolos para interactuar entre si en futuros proyectos.

Para realizar esta tarea, es necesario sincronizar los contenedores para que la aplicación no use la base de datos antes de que los datos se hayan cargado en ella. Para ello hemos usado "wait-for-it.sh"[5], un script para Docker-Compose que se encarga de poner un tiempo de espera para conexiones TCP entre host y puerto. Dispone de licencia "MIT License" que nos permite usarlo.

Cuando se ejecute el script de Docker, éste lanzará la aplicación y estará disponible en la dirección IP por defecto donde se ejecute nuestro servicio Docker, en el puerto 8181. La imagen ocupa alrededor de 2 GB con todos los requerimientos instalados para su pleno uso.

[5]https://github.com/vishnubob/wait-for-it

Capítulo 6

Conclusiones y Trabajo Futuro

En este capítulo hablamos de las conclusiones obtenidas durante el desarrollo de este TFG.

6.1. Conclusiones

El problema de la sobrecarga de información es un problema que sigue creciendo y que no tiene intención de desaparecer si tenemos en cuenta que cada día que pasa nuevos negocios deciden aventurarse a ofrecer sus productos en el gran mercado digital que es Internet. Un tipo de entorno que ha empezado a manifestar este tipo de problema son los jueces de programación en línea. Fue así que empezamos a estudiar las posibles soluciones que podrían aliviar este problema.

Tras haber realizado el estudio de jueces en línea, sistemas de puntuaciones y sistemas de recomendaciones nos topamos con que muchos jueces cuentan con métodos que puedan recomendar problemas de manera personalizada a sus usuarios. Con esto en mente empezamos a planear como abordar el problema, que tipo de sistemas de puntuaciones podrían utilizarse y como se generarían las recomendaciones.

Tras haber implementado variaciones del sistema de puntuación ELO y las maneras en las que haríamos las recomendaciones, empezamos a ponerlas a prueba con una copia de la base de datos de *¡Acepta el reto!* y analizamos los resultados para conocer la precisión de los sistemas de puntuación y recomendación.

Con toda la lógica desarrollada, decidimos utilizarla para crear un pequeño banco de pruebas en el que se podría consultar y gestionar información

de la base de datos y escoger que sistema de puntuación utilizar para calcular puntuaciones. Esta aplicación nos permitió experimentar con distintos aspectos de los sistemas y obtener más datos.

Creemos que los sistemas de puntuación y recomendación propuestos en este TFG son buenos a la hora de medir la habilidad y dificultad de usuarios y problemas, y que pueden adaptarse a otras necesidades sin demasiada complicación. El banco de pruebas desarrollado tiene una interfaz intuitiva que permite experimentar con distintos aspectos del sitio fácilmente.

6.2. Lineas de trabajo futuro

Una de las funciones que interesaría incorporar en la aplicación web es la de establecer métodos de cálculo del ELO distintos a los ya establecidos, relegando los tipos de ELO a meros estándares o métodos predeterminados y ajustando valores en relación con las necesidades del administrador del sitio. También habría que buscar una manera de reducir el tiempo empleado en el recalculo de las puntuaciones al cambiar el conjunto de fórmulas establecidas.

Otra de las tareas que resultaría interesante es la de incorporar el recomendador en *¡Acepta el reto!*, para poder poner a prueba la precisión de nuestro recomendador con usuarios reales, y para poder analizar el comportamiento de estos para realizar ajustes en las fórmulas desarrolladas en caso de ser necesario.

Por otra parte, debido a la baja complejidad del sistema de puntuación desarrollado, una oportunidad futura la de ampliar y extender la funcionalidad y personalización. Hay que tener en cuenta que la principal ventaja del sistema de calificación ELO no es el poder expresivo, sino su simplicidad y versatilidad. Debido a que no es obligatorio resultados binarios en el cálculo de ELO, pueden añadirse variables nuevas, como el tiempo de pasa entre un primer envió y un envío que provoque la modificación de la puntuación.

El sistema de recomendación desarrollado es extensible y general, por lo que un futuro trabajo puede ser la ampliación de este, fuera del marco de *¡Acepta el reto!*, de modo que pueda estudiarse su viabilidad y precisión sobre otros jueces en línea.

Chapter 6

Conclusions and Future Work

In this chapter we discuss the main conclusions that came out after working on this project.

6.1. Conclusions

Information overload is a problem that is still growing and that doesn't show any sign of slowing down anytime soon if we consider the number of businesses that decide to move into the digital world. This problem has also spread to other kind of sites, like online programming judges. This motivated us to study new ways to solve this problem.

After studying multiple online judges, rating systems and recommender systems we found out that many of these judges had no way of recommending problems to their users. With this in mind started thinking about which rating system would be the best to use and how it would help to generate recommendations.

Once we developed different versions of the ELO rating system and proposed ways to recommend problems we started testing them with a copy of the database from *¡Acepta el reto!*, and then we analyzed the outcome of these tests to measure how precise these systems were.

After coding both systems we decided to use them to build a testbed in which a user could check and manage information from the database, and choose which rating system to use to calculate new scores. This app allowed us to test different settings of the rating system and obtain valuable data after analyzing it.

We believe that both the rating system and recommender system proposed in this paper are good when it comes to measuring users' skills and problems' difficulty. They also are easy to adapt to different environments. The testbed has a clean and intuitive interface that lets you configure and test the different rating systems available.

6.2. Future lines of work

An interesting addition to the testbed would be the option to be able to not only choose the types of rating system to use from a set, but also give admins the liberty to make their own rating systems or recommender systems within the app, so they can try their ideas with little coding. It would also prove beneficial if someone were to optimize the code used to recalculate scores so it takes even less time.

Another interesting line of work would be to test our recommender system with live users from *¡Acepta el reto!*, which could help to measure the recommender system's true accuracy and possibly make adjustments to it.

Due to the low complexity of the developed rating systems, a future work could consist in extending its functionality and making it a bit more custom. We need to keep in mind that the main advantage of the ELO rating system is its simplicity and versatility. Since binary results are not mandatory in the ELO calculation, new variables could be added, such as the time between the first submission and the submission that causes the modification of the score, the time a user's been logged while trying to solve a problem, memory use, run time, etc.

The recommendation system we developed was made extensible and non-exclusive so that, in the near future, could be used outside of *¡Acepta el reto!*, so that its viability can be studied in other online judges.

Capítulo 7

Trabajo Personal

"Shoot for the moon. Even if you miss, you'll land among the stars."
— Norman Vincent Peale

7.1. Ederson Funes Castillo

En las primeras semanas del curso 2018/2019, tras la primera reunión con los directores del Trabajo de Fin de Grado, empezamos a investigar sobre los sistemas de recomendación. Yo me encargué de revisar las distintas técnicas utilizadas hasta el momento, tanto en jueces en línea como en otros entornos, y buscar artículos que ofrecieran nuevas alternativas más eficaces. Con la información obtenida de esta primera investigación, mi compañero y yo optamos por desarrollar un sistema de recomendación basado en el sistema de puntuación ELO.

Nuestros directores pusieron a nuestra disposición una copia de la base de datos de *¡Acepta el reto!* para conocer más a fondo el funcionamiento del juez y sus limitaciones. Yo me encargué de revisar los datos disponibles, analizar el comportamiento de los usuarios y filtrar los datos redundantes o irrelevantes para el desarrollo de nuestro trabajo.

Con esto en mente, empecé a escribir los primeros *scripts* para estudiar más a fondo el funcionamiento del sistema ELO y de las fórmulas que lo componen. En un principio utilicé las fórmulas originales del sistema ELO, las cuales se comportaban idóneamente si lo aplicábamos a jugadores de ajedrez, pero que generaba resultados extremos al aplicarlos a pruebas con los usuarios de *¡Acepta el reto!*, por lo que decidí que era necesario adaptar la mayoría de ellas.

Las fórmulas que adapté fueron: la del factor K, para asegurar que la perdida o ganancia de puntos entre contrincantes con grandes diferencias de expectativas sea lo más justa posible, y la fórmula del cálculo de la expectativa, cuya versión original estaba pensada para puntuaciones de hasta más de 2700 puntos, mientras en nuestro sistema el límite es de 16 puntos. Con las fórmulas definidas, realizamos múltiples pruebas y obtuvimos resultados que no nos fueron muy satisfactorios, en los que las puntuaciones oscilaban de forma radical y nunca se estabilizaban. Tras discutirlo con nuestros directores, se decidió generar tres modelos distintos, cuya principal diferencia eran las condiciones que tenían que cumplir los envíos para ser considerados como enfrentamientos validos.

Antes de definir el funcionamiento del sistema de recomendación, mi compañero y yo discutimos las distintas técnicas que se podrían utilizar a la hora de recomendar problemas a usuarios, y concluimos que lo mejor sería hacer recomendaciones basándonos en la similitud de puntuaciones entre usuarios y problemas, teniendo en cuenta las preferencias que tienen algunos usuarios por determinadas categorías de problemas. Yo fui el encargado de calcular la precisión de nuestro recomendador partiendo de cada uno de los modelos de puntuación desarrollados anteriormente.

Para el desarrollo de la aplicación web, mi compañero empezó a escribir las diferentes funcionalidades que se le ofrecerían a los usuarios. Yo estuve a cargo de adaptar el código original (desarrollado para pruebas en un entorno controlado) para poder incorporarlo en la aplicación. Fue en esta fase en la que me di cuenta de que la librería utilizada para generar las gráficas era insuficiente para visualizar los datos, puesto que generaba imágenes en formatos JPEG y PNG en lugar de generar código HTML que se pudiese incluir en una página web, por lo que tuve que buscar otras herramientas que satisficieran nuestras necesidades.

Dado que una de las funcionalidades que se quisieron incluir era la de cambiar el modo en el que se calculaban las puntuaciones, tuve que modificar el código para que fuese capaz de recalcular las puntuaciones procesando todas las entradas de la base de datos. Debido a la cantidad de datos a procesar y la complejidad de los cálculos, el tiempo de ejecución para esta función era de más de 10 minutos. Tras unos ajustes que provocaban un mayor uso de memoria, fui capaz de reducir el tiempo a algo menos de un minuto (en las máquinas utilizadas en las pruebas).

Por último, ayudé en la redacción y edición de la mayoría de capítulos incluidos en esta memoria.

7.2. José Luis Gómez Alonso

En la primera parte del proyecto, la investigación, me involucré junto a mi compañero en la investigación sobre distintos jueces de programación online acerca de su sistema de recomendación, si lo tenían, y de como trabajaban en relacionar sus problemas con los usuarios. Además, contamos con la revisión de artículos sobre estos sistemas, que más se acercaran a nuestro objetivo. Una vez formada una visión global del estado del arte, decidimos junto con nuestros tutores que la forma más interesante de abordar el problema era mediante un sistema de recomendación basado en el sistema de puntuación ELO. Nuestros tutores nos facilitaron una copia de la base de datos de *¡Acepta el reto!* con la que podíamos probar nuestro sistema con usuarios reales y procedimos a su estudio. A continuación yo revise gran cantidad de artículos sobre el sistema ELO acerca de en que sitios se aplicaba y en que forma; y si habían tenido alguna relación previa con algún tipo de sistema de aprendizaje y/o educacional. Escogí varias maneras de transformar el sistema ELO a un sistema con las peculiaridades funcionales y técnicas de *¡Acepta el reto!* y junto con mi compañero decidimos cuales fueron escogidas y desechadas usando como punto de referencia las posibilidades de la BD que nos facilitaron los tutores y el funcionamiento de *¡Acepta el reto!*

En la segunda parte del proyecto, el desarrollo de los scripts de recomendación, mi compañero se encargo del diseño, creación y desarrollo de los mismos y yo me involucre junto a el, el en testeo y depuración y en el análisis de los resultados. Nos dimos cuenta de que en un principio la adaptación del sistema ELO a los usuarios de *¡Acepta el reto!* había sido llevada a cabo correctamente, pero, en el análisis de los resultados, estos eran dispares y con tendencias a extremos; por lo que decidimos alterar las formulas originales de ELO para adaptarlas a la BD y a la forma de trabajar de *¡Acepta el reto!*. Las adaptaciones fueron numerosas y complejas pero las más reseñables son: la decisión entre escoger una función de distribución o un adecuado ajuste del factor K para equilibrar la perdida y ganancia de puntos entre contrincantes con grandes diferencias de expectativa de victoria, el ajuste de corrección en el denominador de la formula original de expectativa de ELO debido a nuestro bajo rango de diferencias entre niveles; y la generación de tres maneras diferentes de realizar el calculo del ELO en función de las condiciones que tenían que cumplir los envíos para ser considerados para el calculo del nuevo ELO, esto debido a un constante desajuste en la fluctuación de los valores de ELO de los usuarios a lo largo del tiempo. Una vez construido un sistema de puntuación que evolucionaba y era compatible con *¡Acepta el reto!*, mi compañero y yo trabajamos sobre cual era la mejor manera de abordar la recomendación a los usuarios de problemas y, junto con los tutores, tras haber investigado distintas opciones, nos decantamos por recomendar problemas

basándonos en cercanía de la diferencia absoluta entre problemas y usuarios, añadiendo además preferencias de los usuarios por categoría.

En la tercera parte del proyecto, el desarrollo de la Aplicación Web, diseñe y desarrolle en combinación de múltiples lenguajes y frameworks el esqueleto de la misma y las primeras funcionalidades que se ofrecían al usuario. Mi compañero adapto el código de los scripts de recomendación para su inclusión en la aplicación y después, realizamos el conjunto de correcciones y extensiones que junto con los tutores, nos pusimos de acuerdo para mejorar la aplicación. En la parte final del trabajo, mi compañero se encargo de realizar mejoras visuales y en la interfaz de la aplicación, mientras que yo me encargué de portabilizar la aplicación a un formato como docker, para que así pudiera ser usada con mayor facilidad. El proceso de *dockerizar* la aplicación y lanzarla operativa estuvo lleno de dificultades pero lo más destacado fue la inclusión de la aplicación y la base de datos en dos contenedores conectados entre si y más específicamente, la sincronización entre estos dos, puesto que la aplicación tenia que esperar a que la BD se generara por completo para empezar a realizar operaciones y la información existente para este problema es escasa.

Por ultimo, aprendí el funcionamiento de LaTeXy su edición, y ayude en la redacción y edición de la mayoría de capítulos incluidos en esta memoria.

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

Caro Martínez, M. Sistemas de recomendación basados en técnicas de predicción de enlaces para jueces en linea. 2017. Disponible en `https://eprints.ucm.es/43975/` (último acceso, May, 2019).

Elo, A. *The rating of chess players, past and present*. Arco Pub, New York, 1978. ISBN 9780668047210.

Glickman, D. M. E. The glicko system. 1995. Disponible en `http://www.glicko.net/glicko/glicko.pdf` (último acceso, May, 2019).

Gómez-Martín, P. P. y Gómez-Martín, M. A. ¡acepta el reto!: juez online para docencia en español. *Actas de las XXIII JENUI*, 2017.

Herbrich, R., Minka, T. y Graepel, T. Trueskill: A bayesian skill rating system. 2007. Disponible en `http://papers.nips.cc/paper/3079-trueskilltm-a-bayesian-skill-rating-system.pdf` (último acceso, May, 2019).

Jiménez Díaz, G., Gómez-Martín, P. P., Gómez-Martín, M. A. y Sánchez-Ruiz, A. A. Similarity metrics from social network analysis for content recommender systems. *AI Communications*, 2016.

Martínez, O. S., G-Bustelo, C. P., Crespo, R. G. y Franco, E. T. Using recommendation system for e-learning environments at degree level. *International Journal of Artificial Intelligence and Interactive Multimedia, Vol. 1, Nº 2.*, 2015.

Papousek, J., Pelánek, R. y Stanislav, V. Adaptive practice of facts in domains with varied priorknowledge. 2014. Disponible en https://pdfs.semanticscholar.org/6223/18ddafacc4e77d96f5e222a772df816b6214.pdf (último acceso, May, 2019).

Pelanek, R. Applications of the elo rating system in adaptive educational systems. 2016. Disponible en https://www.fi.muni.cz/~xpelanek/publications/CAE-elo.pdf (último acceso, May, 2019).

Reaney, P. Online sharing, information overload is worldwide problem: poll. 2012. Disponible en https://www.reuters.com/article/net-us-technology-mobile-poll/idUSBRE8840NF20120905 (último acceso, May, 2019).

Ricci, F., Rokach, L. y Shapira, B. Introduction to recommender systems handbook. 2010. Disponible en http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf (último acceso, Sep, 2019).

Terveen, L. y Hill, W. Beyond recommender systems: Helping people help each other. páginas 487–509, 2001. Disponible en http://files.grouplens.org/papers/rec-sys-overview.pdf (último acceso, Sep, 2019).

This is the way the world ends
This is the way the world ends
This is the way the world ends
Not with a bang but a whimper.

The Hollow Men
T. S. Eliot