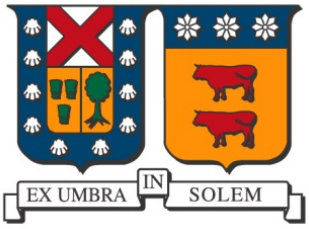
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“DESARROLLO DE UN SISTEMA DE GENERACIÓN AUTOMÁTICA DE EJERCICIOS Y CASOS DE PRUEBA UTILIZANDO LARGE LANGUAGE MODELS PARA CURSOS INTRODUCTORIOS DE PROGRAMACIÓN”

OSCAR HERNÁN ACEVEDO BONVIN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Federico Meza
Profesor Correferente: Andrea Vásquez Guerra

Diciembre - 2024

DEDICATORIA

Este trabajo es dedicado para todas aquellas personas que creen que lograr sus metas es imposible o pertenecen a un sueño muy lejano.

El fracaso no me ha detenido antes, no lo hará ahora.

AGRADECIMIENTOS

Quiero agradecer a mi familia por confiar en mis capacidades y en las esperanzas que me dieron a lo largo de mi carrera. Agradezco a mis padres por su paciencia y por permitirme perseguir mis sueños. Agradezco a mis hermanos por estar conmigo en todo momento y por no rendirse en el camino que la vida les ofreció. Verlos crecer y adaptarse a los desafíos me hizo creer que rendirse nunca es la opción correcta.

A mis compañeros y amigos que me han acompañado desde 2019, gracias a ellos pude conocer a diferentes personas maravillosas y establecer amistades que espero duren muchos años.

Agradezco a Matías González por mostrarme lo maravilloso y grandioso que puede ser el mundo de la informática. Sin su ayuda y presencia, no creo que este trabajo hubiera sido posible. También quiero agradecer a Juan Pablo Cassis; gracias a él entendí que un amigo no es aquel con el que hablas todos los días, sino aquel que está disponible para ayudarte incluso en los momentos más confusos de tu vida.

Agradezco a Nicolás ”Hatziven” Córdoba, Martín Rodríguez y Patricio Pérez por hacerme reír todos estos años, ya sea jugando conmigo o mostrándome cómo juegan. Pero, aún más importante, por mostrarme que siempre se puede encontrar una forma de mejorar en esta vida.

A Federico Meza y Andrea Vásquez, por darme la oportunidad de trabajar en esta memoria. Estuvieron siempre comprometidos con el desarrollo correcto del proyecto y, aún más importante, con el desarrollo integral de todos los estudiantes de la asignatura de Programación. Gracias por su constante esfuerzo y dedicación.

A veces un corazón frío, necesita una sonrisa caluros.

RESUMEN

Resumen— La creación de material didáctico para cursos masivos de programación es una tarea compleja y laboriosa, que requiere muchas horas para desarrollar contenido que se reutiliza a lo largo de varios semestres académicos y pueden ser muy complejos de resolver para los estudiantes que están empezando a programar. En esta Memoria se presenta una mejora a la plataforma de SMOJ capaz de generar ejercicios junto con sus correspondientes casos de prueba para cursos introductorios de programación. Además, genera una ruta de aprendizaje para los estudiantes dependiendo de su propia capacidad para resolver los problemas generados, con el fin de mejorar la eficacia de los estudiantes en la programación. Los resultados luego de probar la plataforma muestran que los estudiantes se sienten más confiados en resolver ejercicios cuando comienzan haciendo ejercicios más adecuados a sus niveles y pudieron sentirse mas cómodos con la materia presentada.

Palabras Clave— Modelos de Lenguaje Grandes Educación en programación; Generación de ejercicios.

ABSTRACT

***Abstract*—** Creating educational material for massive programming courses is a complex and labor-intensive task that requires many hours to develop content that will be reused over several academic semesters and can be very challenging for students who are new to programming. This report presents an enhancement to the SMOJ platform capable of generating exercises along with their corresponding test cases for introductory programming courses. Additionally, it creates a learning path for students based on their ability to solve the generated problems, aiming to improve students effectiveness in programming. The results after testing the platform show that students feel more confident in solving exercises when they start with tasks that are more suited to their levels and feel more comfortable with the material presented.

***Keywords*—** Large Language Models; Programming education; excercise generation.

GLOSARIO

DI: Departamento de Informática.
UTFSM: Universidad Técnica Federico Santa María.
LLMs: Large Language Models
SMOJ: Santa Maria Online Judge

ÍNDICE DE CONTENIDOS

RESUMEN iv

ABSTRACT iv

GLOSARIO v

ÍNDICE DE FIGURAS viii

ÍNDICE DE TABLAS viii

INTRODUCCIÓN 1

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA 3
1.1 Contexto . . . 3
1.2 Situación Actual . . . 4
1.3 Propuesta . . . 5
1.4 Objetivos . . . 6
1.4.1 Objetivo General . . . 6
1.4.2 Objetivos Específicos . . . 6

CAPÍTULO 2: MARCO CONCEPTUAL 7
2.1 Educación de la programación . . . 7
2.2 Dificultades en la educación de programación . . . 8
2.3 Large Language Models . . . 9
2.4 Large Language Models en la Educación . . . 10
2.5 Casos de prueba . . . 10
2.6 Jueces en linea . . . 11
2.7 Trabajos relacionados . . . 12

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN 13
3.0.1 Arquitectura . . . 13
3.1 Implementación con SMOJ . . . 14
3.1.1 Página web de SMOJ . . . 14
3.1.2 Base de datos de SMOJ . . . 15
3.2 Sección de LLM . . . 16
3.2.1 Tecnologías Usadas . . . 16
3.2.2 Generación de Ejercicios con LLMs . . . 16
3.2.3 Revisión de los ejercicios generados . . . 21
3.2.4 Generación de casos de pruebas con LLMs . . . 23
3.2.5 Revisión de los casos de pruebas generados . . . 26

4 VALIDACIÓN DE LA SOLUCIÓN 28
4.1 Metodología . . . 28

4.2 Pruebas con Estudiantes 28
4.2.1 Contexto 28
4.2.2 Encuesta pre-experiencia 28
4.2.3 Resultados encuesta pre experiencia 29
4.2.4 Preguntas post experiencia 31
4.2.5 Resultados encuesta post experiencia 31
4.3 Encuesta de Docentes 33
4.3.1 Metodología 33
4.3.2 Preguntas generadas con la plataforma 34
4.3.3 Respuestas de Docentes 37
4.4 Limitaciones 39

CAPÍTULO 5: CONCLUSIONES **40**
5.1 Conclusiones Generales 40
5.2 Trabajo a futuro 40

REFERENCIAS BIBLIOGRÁFICAS **42**

ÍNDICE DE FIGURAS

. 14
. 18
. 20
. 22
. 22
. 23
. 24
. 25
. 26
. 26
. 27
. 34
. 35
. 36

ÍNDICE DE TABLAS

1 Preguntas pre experiencia. 29
2 Preguntas post Experiencia. 31
3 Preguntas para Docentes. 33

INTRODUCCIÓN

La consolidación de la informática como una disciplina fundamental en la era digital ha provocado un aumento en la cantidad de personas interesadas en estudiar programación, lo que ha incrementado la demanda de cursos introductorios. Estos cursos suelen ser el primer acercamiento al mundo de la programación para los estudiantes, por lo que su creación conlleva una responsabilidad y un trabajo considerable para profesores y ayudantes.

La cantidad de trabajo para crear estos cursos introductorios no se limita a la producción de material audiovisual para enseñar a los estudiantes; también incluye la elaboración de material práctico, como ejercicios de programación que ponen a prueba los conocimientos y la capacidad de los estudiantes para resolver problemas.

Sin embargo, crear estos ejercicios junto con sus casos de prueba requiere una gran cantidad de tiempo y esfuerzo. Por ello, la Universidad Técnica Federico Santa María ha desarrollado su propia plataforma, el Santa María Online Judge (SMOJ). Esta plataforma permite a los estudiantes resolver problemas propuestos por profesores y ayudantes, y recibir retroalimentación sobre el funcionamiento de sus códigos con diferentes casos de prueba.

Con la evolución de la inteligencia artificial y los modelos de lenguaje de gran tamaño (LLMs), se ha mostrado un gran potencial para automatizar diversas actividades. Con la disponibilidad de servicios de LLMs proporcionados por diferentes empresas, como las APIs de ChatGPT, existe la posibilidad de implementar estos modelos en diversas plataformas.

En esta memoria, se propone la implementación de LLMs en la plataforma SMOJ. Esta implementación se basa en la creación automática de ejercicios con sus casos de prueba, adaptados a diferentes características y niveles de dificultad, y que incluyan errores comunes que los estudiantes cometen al programar.

Esta memoria se desarrolla en paralelo con un proyecto de Matías González, que busca incorporar mecanismos para asegurar la comprensión de problemas de programación por parte de los alumnos, así como asistirlos en la construcción de soluciones mediante la retroalimentación sobre su código incorrecto. De ese proyecto se obtienen los errores que cometen los estudiantes, los cuales se utilizan para generar los ejercicios.

El presente documento se divide en 5 grandes capítulos.

El primer capítulo consiste en la definición del problema, donde se describe el contexto respectivo, la situación actual, un resumen de la propuesta y los objetivos que se definieron.

El segundo capitulo presenta el marco conceptual. En esta sección se describen las bases conceptuales sobre la que esta fundamentada la solución planteada.

El tercer capitulo corresponde a la propuesta de solución. En esta sección se detalla el desa-

rrollo de la solución propuesta. Se incluyen metodología y explicaciones necesarias para una una mejor comprensión de la solución.

El cuarto capitulo corresponde a la validación de la solución. En este se valida la solución propuesta, demostrando con diferentes pruebas que la implementación es valida en el entorno. Estas pruebas fueron apoyadas por encuestas y entrevistas.

El quinto y ultimo capitulo son las conclusiones. Se explican los aprendizajes del trabajo realizado, también se mencionan ciertas consideraciones, el alcance del proyecto y el trabajo a futuro para aquellos que deseen profundizarlo.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Contexto

La informática se ha consolidado como una disciplina fundamental en la era de transformación digital, obteniendo un papel central en impulsar avances tecnológicos y en facilitar soluciones a problemas contemporáneos. Se ha convertido en una competencia esencial en una amplia gama de campos profesionales, abarcando desde la ingeniería hasta el desarrollo web e investigación, entre muchos otros [Council, 2021].

Dado el papel crítico de la informática, ha aumentado el número de personas interesadas en unirse a este ámbito, así como la necesidad de conocimientos informáticos en otras carreras. Esto ha generado una demanda creciente, tanto en cantidad como en calidad, de cursos de informática, subrayando la importancia de educar adecuadamente a los futuros profesionales [Council, 2021].

Es crucial destacar la importancia de una educación académica de calidad, especialmente en los cursos introductorios de programación. Estos cursos no solo deben ser fundamentales y bien estructurados, sino también interesantes y atractivos para los estudiantes. El material complementario, como los ejercicios, juega un papel crucial en este proceso, requiriendo un alto nivel de diseño y adaptabilidad para satisfacer las diversas necesidades de los estudiantes [Whalley *et al.*, 2006].

Sin embargo, crear estos cursos no es tarea fácil. Se debe considerar la diversidad en los niveles de conocimiento previo de los estudiantes, como en matemáticas, y adaptar el contenido del curso para abordar estas diferencias. Estas variaciones en los niveles de habilidad añaden complejidad a la generación de ejercicios complementarios de programación.

Por lo tanto, diseñar cursos introductorios de programación es un desafío significativo, especialmente dada la rápida evolución de las tecnologías. Se enfrenta la tarea de elegir la metodología adecuada, diseñar el contenido y el flujo del curso y, sobre todo, proporcionar una variedad de problemas adecuados para los estudiantes [Falkner y Sheard, 2019]. Estos problemas deben tener características específicas, como un nivel de dificultad apropiado y un enfoque claro, teniendo en cuenta las necesidades y opiniones de los estudiantes.

Los ejercicios y problemas propuestos son esenciales para el desarrollo integral del estudiante. No solo ofrecen la oportunidad de aplicar lo aprendido en clase, sino que también facilitan la retroalimentación valiosa de profesores y ayudantes. Al resolver estos ejercicios, los estudiantes desarrollan habilidades para interpretar y abordar adecuadamente los desafíos presentados [Sanders *et al.*, 2017]. Por lo tanto, es fundamental contar con una diversidad de ejercicios de distintos niveles de dificultad en estos cursos, acompañados de métodos

efectivos de evaluación y retroalimentación.

1.2. Situación Actual

La UTFSM es actualmente una de las universidades más prestigiosas de Chile, destacándose en sus áreas de ingeniería y proclamándose líderes en ciencia y tecnología. La universidad se considera como un semillero de futuros profesionales, con uno de los cursos introductorios a las carreras de ingeniería siendo Programación, con siglas “IWI-131”, este curso es considerado un pilar en la formación de futuros ingenieros, no solo de informáticos.

Este curso simboliza el primer acercamiento de muchos estudiantes en la programación. Actualmente, se inscriben 3000 estudiantes cada semestre al curso, pero esto representa un desafío a la hora de enseñar, puesto que al ser tantos estudiantes, resulta difícil gestionar todos los recursos del ramo, también es complicado el crear las planificaciones del ramo con sus contenidos, y además hay que tener en cuenta el nivel de cada estudiante, puesto que existen alumnos con experiencia previa en programación que encontraran el curso poco estimulante. Esto lleva a una metodología flexible y adaptable para poder satisfacer la gran variedad de estudiantes.

Actualmente, el curso de programación se centra en la formación del pensamiento algorítmico y en la resolución de problema usando el lenguaje de programación Python. El material que se ha de proporcionar a los estudiantes ha de ser didáctico, comprensible y desafiante. Uno de los aspectos más críticos es la generación de los ejercicios y sus casos de prueba. Estos ejercicios son generados manualmente por los docentes de la universidad, provocando que sea un proceso que conlleva considerable tiempo y esfuerzo para asegurar su efectividad en el proceso educativo de los estudiantes.

La creación manual de los ejercicios es bastante problemática y laboriosa. Como los ejercicios propuestos han de ser estimulantes y relevantes en los contenidos del curso, provoca una dificultad a la hora de generarlos. Está dificultad se mantiene a la hora de desarrollar los casos de pruebas. Los casos de pruebas han de evaluar correctamente el código creado por los estudiantes, han de ser capaces de trabajar con los casos bordes o los casos más complicados del ejercicio. Esto provoca la reutilización de contenido de semestres anteriores, el que puede volverse monótono y aburrido.

La UTFSM cuenta con una plataforma online llamada Santa Maria Online Judge(SMOJ) [1], en la cual los estudiantes son capaces de conectarse con su cuenta institucional y resolver ejercicios propuestos para cada unidad. Todos estos ejercicios son generados con la metodológica manual explicada. Con esta plataforma los estudiantes son capaces de aplicar los conocimientos y contenidos explicados en clase de una manera mas práctica.

Actualmente, SMOJ funciona como una gran librería de ejercicios del curso. Estos ejercicios

[1] https://smoj.inf.utfsm.cl/

son creados manualmente por profesores y ayudantes, pero son los mismos ejercicios para todos los estudiantes cursando el ramo y suelen repetirse entre semestres. También se han implementado mejoras al SMOJ, tal como [LEYTON, 2022], donde Hugo Leyton mejoró la plataforma con tal de que los estudiantes pudieran conseguir ejercicios basándonos en las capacidades de los mismos estudiantes.

La metodología actual de SMOJ es una liberación de ejercicios dependiendo de la unidad que se esté cursando. Esta unidad suele cambiar cada 1 a 2 semanas aproximadamente y los estudiantes tienen como obligación el responder completamente al menos un ejercicio por semana, escogiendo cualquiera de los ejercicios que se encuentran disponibles de la unidad.

1.3. Propuesta

Para abordar la problemática identificada, se sugiere una mejora en la plataforma existente del SMOJ, integrando la capacidad de generar ejercicios de programación automáticamente mediante el uso de Large Language Models (LLMs). Esta innovación permitiría a los estudiantes acceder a ejercicios nuevos y personalizados, diseñados para ajustarse a su nivel de habilidad. Los profesores, por su parte, se centrarían en la revisión y, de ser necesario, en la modificación de los ejercicios generados, optimizando así su tiempo y recursos.

La propuesta incluye la implementación de una sección de generación automática en la plataforma, donde los estudiantes puedan solicitar ejercicios específicos relacionados con la unidad temática que se esté abordando en el curso. Un aspecto fundamental de este sistema es la capacidad de medir tanto la dificultad de los ejercicios como la habilidad del estudiante para resolverlos. Para ello, se contempla el desarrollo de estrategias para formular las solicitudes a los LLMs en un lenguaje humano, especificando claramente el nivel de dificultad deseado, ya sea básico, intermedio o avanzado. Incluso se considera la posibilidad de que el propio LLM determine la dificultad del ejercicio tras generarlo, basándose en parámetros preestablecidos.

El objetivo principal de esta propuesta es enriquecer el proceso de aprendizaje en programación, incidiendo directamente en la cantidad y en la adaptabilidad de la dificultad de los ejercicios ofrecidos a los estudiantes. Se espera que esto no solo aumente el volumen de ejercicios disponibles, sino que también mejore su relevancia pedagógica.

Esta innovación también busca minimizar la frustración de los estudiantes al enfrentarse a desafíos que superan su capacidad actual, evitando así el desánimo y la aversión a la programación. Al mismo tiempo, se pretende evitar una falsa sensación de seguridad al resolver ejercicios demasiado fáciles, manteniendo así un equilibrio que fomente el progreso constante y la comprensión profunda de los conceptos clave.

Finalmente, otro beneficio significativo de este sistema es su impacto positivo en la labor de los profesores y ayudantes de curso. Al reducir la necesidad de crear manualmente mate-

riales complementarios, podrán dedicar más tiempo a proporcionar retroalimentación personalizada y de mayor calidad a los estudiantes, mejorando la experiencia educativa en su conjunto.

1.4. Objetivos

Tanto el objetivo general como los específicos son nombrados a continuación:

1.4.1. Objetivo General

- Implementar un sistema para la generación automática de ejercicios de distintas características y dificultades con sus casos de prueba para apoyar cursos introductorios de programación con el uso de Large Language Models.

1.4.2. Objetivos Específicos

- Gestionar la ruta de aprendizaje de los estudiantes en base a diferentes características de los problemas.
- Integrar el sistema desarrollado con el sistema de evaluación de la plataforma SMOJ y con un sistema de análisis de las soluciones de los estudiantes.
- Evaluar el impacto del sistema desarrollado en la auto eficacia y percepción de los estudiantes y analizar la potencial mejora en el aprendizaje.

CAPÍTULO 2
MARCO CONCEPTUAL

2.1. Educación de la programación

La educación en los temas de programación no es un tema sencillo, y es bastante común que exista una cantidad notable de estudiantes que fallan los cursos de programación básica [Vatchkov *et al.*, 2007], por lo que en las últimas décadas se ha investigado y estudiado el porqué le cuesta aprender a los estudiantes y el cómo mejorar la enseñanza de la programación.

Existen varios estudios que han buscado las mayores problemáticas a la hora de aprender programación, [Demir, 2021] explica que los nuevos estudiantes tienen problemas a la hora de entender la lógica de la programación y el alto nivel de abstracción que demanda la programación. Esto provoca que los estudiantes no sean capaces de entender bien los algoritmos y, por tanto, presentan dificultades a la hora de poner en practica los conceptos enseñados en clases.

Otro concepto que aborda Vatchkov [Vatchkov *et al.*, 2007] es que los cursos de programación son diferentes en comparación a otras materias y cursos, esto se debe a que la "programación requiere de mucha práctica y estudio intensivo", pero esta práctica puede generar estrés en los estudiantes, dado que, en una gran cantidad de ocasiones, ellos no serán capaces de poder resolver los problemas correctamente en sus primeros intentos y tendrán errores de compilación que les generará una cantidad de estrés, la cual podría provocar una deserción del curso.

Por estas razones se han popularizado diferentes metodologías de aprendizajes. Una metodología común es la "metodología de proyecto". En este caso se trabaja con un proyecto en el semestre y se va enseñando programación acorde se va progresando con el proyecto como se hizo en [Drini, 2018], donde se mezcló la metodología de proyecto con una metodología activa para aumentar la compresión y retención del material.

En el ámbito de la enseñanza de programación, el estudio de Fincher [Fincher, 1999] destaca cuatro metodologías principales: "syntax free", "Literacy", "problem solving" y "computation as Interaction". Las metodologías"syntax free" y "Literacy" se enfocan en la enseñanza de conceptos abstractos y el pensamiento algorítmico, prescindiendo del uso directo del código. Esta aproximación permite a los estudiantes comprender los fundamentos de la programación sin la barrera inicial de la sintaxis compleja. Por otro lado, el enfoque "problem solving" se centra en la resolución de problemas, pero podría no cubrir adecuadamente los conceptos de programación, lo cual es esencial para un curso introductorio. Estos enfoques varían en su énfasis en los aspectos prácticos y teóricos de la programación y pueden influir significativamente en cómo los estudiantes perciben y asimilan los fundamentos de la

informática.

2.2. Dificultades en la educación de programación

Globalmente, los cursos de programación siempre han tenido un alto nivel de abandono y una alta tasa de reprobación. Por esto se han creado varias herramientas para asistir tanto a docentes como estudiantes en el proceso de aprendizaje, sin embargo, con el paso del tiempo, estos problemas han cambiado según las diferentes metodologías o se han presentado problemas nuevos en las nuevas generaciones.

Estos factores han sido estudiados extensivamente. Chin Soon Cheah estudio la literatura sobre esto y describió cuatro factores, que se explicaran a continuación, para poder entender y perfeccionar un lenguaje de programación principales [Cheah, 2020]: etapas de la programación, habilidades de resolución de problemas, pedagogía ineficiente y, finalmente, rasgos personales y actitudes.

Las etapas de la programación fueron descritas por Dale y Weems [Dale y Weems, 2016], los cuales explican que existen 2 fases principales para poder dominar un lenguaje de programación, la primera etapa esta relacionada con la resolución de problemas, donde el estudiante ha de aprender los conceptos de la programación y las estructuras de datos para entender y analizar los problemas de manera correcta, esta es una fase teórica en su mayoría, donde se fomenta el desarrollo correcto de un algoritmo y la capacidad de pensamiento. La segunda fase de implementación es considerada la etapa práctica, donde podrán implementar el algoritmo creado en la fase anterior y poder corregir los errores teóricos de la primera fase.

Para el factor de habilidades de resolución de problemas, los estudiantes enfrentan diversas dificultades en este campo, una de las cuales es dificultad para comprender el problema, dificultad para diseñar el problema e incluso problemas para comprender y aplicar ciertos conceptos de programación, Chin Soon Cheah destaca las dificultades a la hora de entender los ciclos. [Cheah, 2020].

Otro factor es una pedagogía ineficiente, esto se puede deber a diferentes características, como problemas con el material educativo, problemas con la estrategia e incluso problemas con los docentes. Se destaca el material educativo, debido a que los cursos usualmente siguen un libro guía que se entrega a los estudiantes, pero estos libros no son interactivos ni sencillos de leer, resultando que la mayoría de los estudiantes no los considere a la hora de estudiar.

El último factor es la personalidad y actitud de los estudiantes, la actitud que toma un estudiante al afrontar un ramo es el factor más importante a la hora de aprender. El estudio de Chin soon Cheah identificó 3 áreas principales, el conocimiento previo, percepción negativa y la actitud. Hablando del conocimiento previo, muchos estudiantes no tienen una base de resolución de problemas, lo que dificulta las cosas, puesto que ahora han de aprender a

programar y generar una base fundamental, provocando una actitud negativa que conlleva dificultades a la hora de querer aprender a programar. Finalmente, está la percepción negativa sobre los programadores y los estudiantes de programación, por lo que las personas pueden mirar mal a estudiar estos cursos y a sus estudiantes.

2.3. Large Language Models

En la era digital actual, la escuela de la informática mas popular es la Inteligencia Artificial(IA), la cual es una forma de crear aplicaciones que realicen tareas sin la necesidad de la intervención humana, un ejemplo famoso serian las IAs que juegan ajedrez[Maharaj *et al.*, 2022]. Pero el termino de IA usualmente es confundido con el termino de Machine Learning, un enfoque que se encarga de aprender mientras consumen datos.

Los avances de la Inteligencia Artificial en los últimos años ha evolucionado de manera significativa, creando varias ramas de estudios, como Machine Learning[Jordan y Mitchell, 2015], con uno de los últimos avances importantes estando en el procesamiento del lenguaje natural, esto ha permitido que las IAs sean capaces de entender el lenguaje humano y ser capaces de escribirlo. Con estos avances de entendimiento del lenguaje humano, se comenzó el desarrollo de IAs que sean capaces de aprender de este y responder imitando a una persona virtual.

Dentro de las ciencias de la Inteligencia Artificial, se experimenta con el aprendizaje automático, un enfoque que permite a las maquinas aprender de datos e información para mejorar su desempeño con el tiempo sin estar programadas de manera explicita, esto les da una posibilidad de aprender sin la necesidad de que una persona modifique su codigo. El Deep Learning, una subdivisión del aprendizaje automático, hace uso de las redes neuronales para poder analizar y aprender a partir de una gran cantidad de datos.

Estos avances llevaron a la creación de IAs, estas son entrenadas con una cantidad masiva de datos con el propósito de la obtención de una base fundamental, en la que la que se busca minimizar la supervisión humana y ser capaces de funcionar de manera autónoma, estas fueron llamadas Large Language Models(LLMs), con el ejemplo mas famoso en la actualidad siendo Chat GPT de OpenAI [OpenAI, 2022][2].

Estos modelos generalmente están basados en arquitecturas como Transformers [Wang *et al.*, 2019], las cuales son un mecanismo de atención para gestionar la relación entre diferentes palabras en un texto. Con esta tecnología son capaces de generar respuestas coherentes y basándose en el contexto de las conversaciones, traducciones, resúmenes y mas. Por todo este potencial muchas empresas han comenzado a invertir en equipos de desarrollo con tal de crear su propio LLM [Wei *et al.*, 2022], entre estas

[2]https://openai.com/chatgpt

empresas se encuentran Facebook Llama [Jayaseelan, 2023][3], Google Bard [Pichai, 2023][4] y la propuesta de Microsoft con CodeBERT [Feng *et al.*, 2020][5].

2.4. Large Language Models en la Educación

La integración de los Large Language Models (LLM) en las metodologías de enseñanza actuales ofrece un panorama prometedor en el campo de la educación. Estos modelos avanzados de procesamiento de lenguaje natural pueden complementar y enriquecer las estrategias pedagógicas, aportando nuevas dimensiones y capacidades al proceso educativo. Por lo que en varios estudios recientes se ha explorado las diferentes ventajas y desventajas de usar los LLM en varios ámbitos educativos.

Los avances en la inteligencia artificial han abierto un amplio abanico de posibilidades en la educación. Como se menciona en [Kasneci *et al.*, 2023], los LLM se han aplicado en variados ámbitos, desde asistir a estudiantes de enseñanza básica en lectura y escritura hasta apoyar el desarrollo profesional de los docentes, ofreciéndoles resúmenes y nuevas ideas para enriquecer su enseñanza. Un ejemplo notable es el uso de ChatGPT-3, que puede generar ejercicios educativos y proporcionar puntos clave para su resolución, orientados desde la perspectiva del estudiante.

Dada la evidente capacidad de los LLM, se han realizado varios estudios sobre su aplicación en la educación. Dijkstra, por ejemplo, investigó la generación de problemas de comprensión de textos utilizando varios LLM, creando cuestionarios con preguntas de selección múltiple basadas en textos dados [R. Dijkstra; Z. Genç, 2022]. Aunque los cuestionarios no eran perfectos, concluyó que existe un potencial significativo y direcciones claras para su evolución.

Mostrando que una implementación práctica de los LLM en entornos educativos abarca una gama de aplicaciones, desde la generación automatizada de materiales de estudio hasta la creación de sistemas de tutoría y soporte. Estos modelos pueden mejorar significativamente la eficiencia en la creación de materiales didácticos y ofrecer una enseñanza más personalizada. Esta sección detalla cómo los LLM pueden ser utilizados para adaptar la enseñanza a las necesidades individuales de cada estudiante, promoviendo así un aprendizaje más efectivo y atractivo.

2.5. Casos de prueba

En el contexto educativo, la corrección de trabajos y ejercicios es una tarea fundamental asociada a los docentes. Esta labor no solo implica la creación y revisión del material, sino

[3] https://ai.meta.com/llama/
[4] https://bard.google.com/
[5] https://github.com/microsoft/CodeBERT

también la evaluación del progreso de los estudiantes y la generación de oportunidades de retroalimentación. A través de esta retroalimentación, los docentes pueden guiar a los estudiantes en su trayectoria de aprendizaje. La revisión, por lo tanto, constituye un pilar esencial en el proceso educativo, siendo una instancia clave para evaluar y fomentar el desarrollo académico.

En el campo de la programación, sin embargo, la revisión y evaluación de los ejercicios presentan desafíos únicos en comparación con otras áreas académicas. Mientras que en asignaturas como las ciencias sociales o las matemáticas las pruebas pueden incluir preguntas de verdadero/falso, selección múltiple o respuestas desarrolladas, en programación se requiere la creación de casos de prueba específicos para evaluar la eficacia y revisión del código. Estos casos de prueba son cruciales para determinar no solo si el código funciona, sino también para asegurar que actúe de manera adecuada ante una variedad de situaciones y entradas, incluyendo aquellas delicadas o específicas.

La importancia de una evaluación rigurosa del código se ve reflejada en estudios como el de Palomba, quien examinó la relación entre la cobertura del código y la calidad de las pruebas [Palomba *et al.*, 2016]. Palomba señala que "*Poorly designed tests are known to have a negative impact on test maintenance, as they are more difficult to adjust when production code changes*", resaltando cómo la calidad de las pruebas incide directamente en la efectividad del proceso de enseñanza y aprendizaje de la programación. Este enfoque subraya la necesidad de diseñar casos de prueba bien estructurados para garantizar una evaluación precisa y completa del código desarrollado por los estudiantes.

2.6. Jueces en linea

Una metodología emergente para la evaluación rápida, segura y transparente en la educación de programación es el uso de jueces en línea. Kurnia los define como "plataformas en línea que admiten una evaluación completamente automatizada y en tiempo real del código fuente, binarios o incluso salidas textuales presentadas por los participantes en un desafío particular" [Kurnia *et al.*, 2001]. Originalmente utilizados en la programación competitiva, una característica distintiva de estos jueces es su adhesión a reglas fijas de corrección, como el tiempo máximo de compilación y ejecución, y el límite de memoria utilizada, características que pueden ser utilizadas de manera sencilla para asignar puntajes a códigos en base a diferentes casos de pruebas en evaluaciones educativas[Luo *et al.*, 2008].

Existen varios jueces en línea notables, incluyendo el Santa María Online Judge de la UTFSM[6], desarrollado y mejorado continuamente por estudiantes de la universidad. Aunque muchos jueces en línea se utilizan predominantemente para la programación competitiva, como Codeforces [7] y el Pekin University Online Judge [8], su papel en la educación de programación

[6] smoj.inf.utfsm.cl
[7] codeforces.com
[8] poj.org

está ganando relevancia, un ejemplo seria mejorándolos para que provean de manera inmediata retroalimentación al estudiante según su código[Zhou *et al.*, 2018]. Estas plataformas no solo ofrecen un medio eficiente para evaluar el código, sino que también proporcionan un entorno donde los estudiantes pueden practicar y mejorar sus habilidades de programación en un contexto estructurado y desafiante.

2.7. Trabajos relacionados

El uso de LLMs en áreas pedagógicas es un ámbito completamente diferente, en especial en el tema de generar problemas y resoluciones útiles para los estudiantes. Un primer vistazo, en el área de las matemáticas, donde [Drori *et al.*, 2022] mostró la posibilidad de usar una red neuronal preentrenada y calibrada que puede ser usada para generar estos ejercicios con sus soluciones, el modelo utilizado es el de OpenAI's Codex. Este modelo fue preentrenado con diferentes ejercicios del MIT en varios ámbitos, como Pre álgebra, álgebra lineal y Computación Científica.

También existe el proceso de creación de los materiales de estudio para Computer Science 1, CS1, termino genérico usado para referirse al primer curso introductorio de programación en las universidades. MacNeil y sus colegas están investigando actualmente sobre como usarlo para generar explicaciones de código, los ejercicios e incluso el código de los ejercicios propuestos [MacNeil *et al.*, 2022].

Otro proyecto importante es [Wakatani y Maeda, 2015], donde Wakatani logró crear ejercicios usando PHP de manera automática, en el que todos estos ejercicios generados siguen diferentes plantillas y reglas, con el objetivo de que los estudiantes entiendan los conceptos de semántica y sintaxis en la programación. El problema siendo que al seguir plantillas, estos podrían llegar a ser repetitivos, provocando que los estudiantes se encuentren frente a experiencias similares y, por tanto, no puedan seguir aprendiendo los conceptos, resolviendo los ejercicios de manera más mecánica que analítica.

No obstante, se han descubierto formas para generar ejercicios de programación de mayor calidad, [Sarsa *et al.*, 2022] [Freitas *et al.*, 2023], en los cuales Sami Sarsa ha logrado generar problemas de CS1 con sus casos de pruebas e incluso explicaciones del código programado por los alumnos, esto con la ayuda de OpenAI Codex, pero con deficiencias, un ejemplo de estos problemas siendo de ortografía y gramática.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

Como se ha mencionado anteriormente, los estudiantes que cursan la asignatura de programación de la UTFSM, deben usar la plataforma de SMOJ para resolver problemas de programación. Esta plataforma les proporciona diferentes ejercicios semanalmente, creados por profesores y ayudantes del curso. Cada uno de estos ejercicios se descompone en dos componentes principales: enunciado y casos de prueba. Los enunciados están compuestos por los siguientes elementos:

- **Título**: Título del ejercicio.
- **Descripción**: Enunciado del ejercicio, que incluye una explicación de lo que se debe resolver, junto con un poco de historia para agregar contexto al ejercicio.
- **Entrada**: Explicación de cómo es la entrada del programa que se debe desarrollar.
- **Salida**: Explicación de cómo es la salida del programa que se debe desarrollar.
- **Observaciones**: Detalles del ejercicio, tales como restricciones del programa o consejos para resolver los ejercicios.

En cuanto a los casos de prueba en SMOJ, cada ejercicio tiene un mínimo de cinco casos de pruebas estáticos, donde cada uno tiene un puntaje asociado, de tal forma que la suma de todos los puntajes asociados a los casos de prueba del ejercicio sea 100. El propósito de esto es que la nota del estudiante corresponda a la cantidad de casos de prueba que su código puede solucionar correctamente.

Dado que todo ya está desarrollado en la plataforma SMOJ, seguiremos trabajando en ella y usaremos las funcionalidades ya creadas, como su base de datos para agregar ejercicios, junto con la revisión y agregación de información de los estudiantes. También usaremos su página web para interactuar con el usuario y, finalmente, su juez en línea para la corrección automática de los ejercicios usando los casos de prueba generados por el LLM.

3.0.1. Arquitectura

Lo que se planea desarrollar es una página web que interactúe con un servidor encargado únicamente de generar y formatear todos los ejercicios y casos de prueba; lo llamaremos ”servidor generador”. El estudiante interactúa con la página web, ingresando las características que desea en el ejercicio. Estas son: una dificultad inicial, la categoría para el ejercicio

y errores comunes que el estudiante comete [González, 2024]. Luego, se envían estas características al servidor generador, que genera diferentes peticiones para el ejercicio y los casos de prueba usando las características obtenidas de SMOJ. También creará dos peticiones para revisar que lo generado por ChatGPT sea correcto, y enviará las peticiones al servidor de ChatGPT. Una vez que el servidor de ChatGPT responda, el servidor generador subirá el ejercicio generado con los casos de prueba a la base de datos de SMOJ y enviará el ID del ejercicio generado a SMOJ para que redireccione al estudiante a la página del ejercicio. Si el estudiante luego quisiera obtener un ejercicio nuevo, mientras tenga intentos podrá pedirlo, pero no podrá seleccionar la dificultad del ejercicio, sino que se calculará una nueva dificultad según la cantidad de intentos y puntaje que obtuvo en la pregunta generada.



Figura 1: Diagrama básico de Arquitectura.
Fuente: Elaboración propia.

3.1. Implementación con SMOJ

3.1.1. Página web de SMOJ

Se realizaron pocas modificaciones a la plataforma de SMOJ. La principal y más grande es la creación de una página en la que se pide el ejercicio de ChatGPT. En esta página existen dos listas desplegables y un botón. La primera lista desplegable contiene valores desde 1 hasta 10, su objetivo es que el estudiante pueda escoger la dificultad del primer ejercicio que pida, una que el estudiante crea que puede resolver. La segunda lista desplegable contiene las categorías de los ejercicios, las cuales pertenecen a todas las unidades que se cursan en la asignatura de programación de la UTFSM. Estas categorías son:

- Programas Secuenciales
- Condicionales
- Ciclos
- Strings
- Funciones

- Listas
- Diccionarios

Para terminar con las modificaciones en la página de SMOJ, definiremos el cálculo de la dificultad, el cual funciona como una clasificación con una escala del 1 al 10. Para este cálculo se usan los intentos y el mejor puntaje que el estudiante haya conseguido en el último ejercicio generado. Se usa la siguiente fórmula:

Algorithm 1 Algoritmo de cálculo de clasificación

$$Bruto = Ultima_Dificultad - ((Intentos - 1) + (Mejor_Puntaje \times \tfrac{3}{15}))$$
$$Normal = \tfrac{Bruto-1}{40-1}$$
$$Dificultad_Final = \text{máx}(\text{mín}((Normal \times 9) + 1, 10), 1)$$

Esta clasificación se creo para poder determinar de manera automática que dificultad puede resolver cada estudiante, de esta manera se puede manejar la ruta de aprendizaje del estudiante con los resultados de los estudiantes. Para generar la dificultad de una manera justa, solo se penaliza por la cantidad de intentos, mientras que el puntaje obtenido solo puede subir la dificultad del estudiante.

3.1.2. Base de datos de SMOJ

Para poder almacenar la información de los estudiantes, como la dificultad del último ejercicio que generaron, cuántos ejercicios pueden generar y cuál fue el último ejercicio que pidieron, se modificó la base de datos de SMOJ creando una tabla llamada 'generated_exercises_by_alumns', la cual contiene las siguientes columnas:

- difficulty
- id_user
- amountExercisesAsked
- LastResolved

Centralizar esta información minimiza las peticiones a la base de datos de SMOJ y permite una mejor supervisión del estudiante. Además, esta información se usa para obtener los datos necesarios para calcular la dificultad del siguiente ejercicio.

3.2. Sección de LLM

3.2.1. Tecnologías Usadas

Para poder generar los ejercicios y casos de prueba de manera correcta y eficiente, se creó un servidor intermedio. Este servidor se encarga de realizar las llamadas a la API del servidor de ChatGPT, asegurarse de que todo lo generado siga ciertas reglas de formato y subirlo a la base de datos para poder mostrarse correctamente en la página web de SMOJ a cada estudiante. Para crear este servidor se usó Python 3.10 como lenguaje de programación y las siguientes librerías de Python:

Elección de modelo de LLM

- **Flask**[9]: Una librería utilizada para crear un servidor de APIs con Python.
- **OpenAI**[10]: Usada para la comunicación con el servidor de OpenAI y la generación de peticiones de manera correcta.
- **LangChain**[11]: Facilita la creación de modelos base y la realización de peticiones estructuradas al servidor de OpenAI, asegurando que las respuestas sigan un formato JSON con claves específicas para los componentes de los enunciados.
- **ChromaDB**[12]: Usada para crear una base de datos de vectores y agregar conocimientos al modelo de OpenAI mediante embeddings, esto es una forma de transformar el lenguaje natural humano en vectores que son mas fáciles de entender para el modelo. Esto permite que los ejercicios generados sean similares a los existentes en SMOJ.
- **PyMySQL**[13]: Se utilizó para solucionar problemas de comunicación entre SMOJ y el servidor generador de ejercicios, especialmente en la transferencia de texto con caracteres especiales.

3.2.2. Generación de Ejercicios con LLMs

Para la generación de ejercicios, se decidió utilizar el modelo GPT-4 de ChatGPT. Inicialmente, se consideró usar los modelos GPT-3 o GPT-3.5-turbo debido a sus costos más bajos. Sin embargo, se observó una notable diferencia en la calidad del texto generado. Muchos ejercicios creados con los modelos GPT-3 tenían inconsistencias entre la descripción del ejercicio

[9] https://flask.palletsprojects.com/en/3.0.x/
[10] https://platform.openai.com/
[11] https://www.LangChain.com/
[12] https://www.trychroma.com/
[13] https://pypi.org/project/PyMySQL/

y lo que se pedía como entrada, además de errores en los casos de prueba. Estos problemas disminuyeron considerablemente al usar el modelo GPT-4, aunque no se eliminaron por completo.

En cuanto al uso de ChromaDB, es importante considerar que cada vector agregado como embedding aumenta el costo de las peticiones a la API. Para reducir los costos generales, se crearon diferentes bases de datos en ChromaDB, cada una conteniendo ejercicios de distintas unidades del curso. Estos ejercicios provienen directamente de los dados en SMOJ durante el primer semestre de 2024 en la asignatura de programación de la UTFSM. Al tener varias bases de datos, es posible entregarle información mas especifica de como son los ejercicios en SMOJ en esa categoría especifica, con la ventaja de no confundir al modelo entregándole tantos ejercicios diferentes que no tienen correlación.

Las peticiones que se trabajan con OpenAI son bastante complejas de diseñar y requieren ser escritas de una manera muy literal. Es fundamental ser muy específico con lo que se quiere pedir, cómo se quiere pedir y cómo se desea recibir la respuesta. Un requisito especifico es que las peticiones deben hacerse en inglés. Aunque el modelo es capaz de funcionar correctamente en cualquier idioma soportado, se diseñaron las peticiones en inglés para reducir la cantidad de texto en cada solicitud. Además, debido a los diferentes modelos que se utilizan, los modelos más antiguos funcionan mejor en inglés que en español debido a la mayor cantidad de datos de entrenamiento en ese idioma. Con esto, logramos peticiones que envían y reciben menos texto, además de mejorar el desempeño del modelo.

Las peticiones se componen de 2 partes esenciales, en este caso las llamaremos 'Sistema'y 'Usuario'. Sistema es texto que especifica el como se tiene que comportar el modelo, quien es en esta llamada y que instrucciones especificas tiene que seguir. Por otro lado, 'Usuario'es lo que nosotros le pedimos al modelo y que responda siguiendo las instrucciones entregadas en 'Sistema'.

Para generar los ejercicios, la información que le entregamos al 'Sistema'es la siguiente:

Sistema

You are a group of teachers with the task to devise engaging and thought-provoking programming exercises for Computer Science 1 students on the spanish language. The exercises should be in Spanish and cater to the following specifications:
The exercises are crafted to closely resemble competitive programming exercises found on platforms like Leetcode.
Each exercise will include a title, description, observations, input description, and output description.
The exercises should revolve around a certain category. The students should only use contents from the specified category to solve the exercises.
Each exercise must involve real-life scenarios to help enhance student engagement.
The exercises should cultivate computational thinking and problem-solving aptitudes in the students.
You have to be very specific on how to solve the exercise and be very literal so that the student can understand the problem, you can provide mathematical functions if you think it's needed
Make sure the problem statements and solutions are clear. They must follow Python's stdin and stdout formats.
If a list is part of the problem's input, start with the count of elements followed by the elements themselves. Make sure that you describe how the code is going to receive the input for the exercise. Try not to use any float numbers and only integers
Given some typical mistakes that students often make when they program in Python, design a programming exercise that specifically targets at least one of these common mistakes. But keep in it mind that the exercise should not merely expose the mistake but should also provide a way for corrective learning. For example, if a common mistake is 'off-by-one in indexing', the exercise could require index manipulation where proper handling of edge scenarios is needed to get correct results.
Have in mind they know this about {CATEGORY}:
{CATEGORY CONTENTS}
Each exercise will be written inline with JSON format and includes the following keys:

- title
- description
- observations
- inputDescription
- outputDescription

Avoid providing any explanation for your thought-process or decisions . The output should strictly be the problem statement.

Figura 2: Rol del Sistema en generación de ejercicios
Fuente: Elaboración propia.

Descomponiendo el Sistema, primero se le pidió que se comportara como un grupo de profesores con el objetivo de generar ejercicios de programación para estudiantes de *Computer Science 1*, el curso introductorio de programación para estudiantes en Estados Unidos. Esta estrategia surgió debido a la baja calidad de los ejercicios generados cuando se le indicaba que trabajara como una Inteligencia Artificial. Posteriormente, se le solicitó que actuara como un solo profesor, lo que mejoró significativamente la calidad de los ejercicios. Sin embargo, los ejercicios seguían siendo monótonos y repetitivos. Esto se corrigió al volver a indicarle que se comportara como un grupo de profesores.

A continuación, se le especificaron las características que todos los ejercicios debían seguir, incluyendo los parámetros esenciales y el enfoque en la resolución de problemas de la vida real. Para lograr esto, se le pidió que se inspirara en ejercicios de Leetcode[14], una plataforma de programación competitiva. Los ejercicios de esta plataforma requieren que los usuarios piensen en cómo resolver los problemas, lo que mejoró la calidad de los ejercicios generados al imitar ejercicios populares de programación competitiva.

A cada rol asignado al sistema se le proporcionaron los contenidos de la unidad que los estudiantes deberían conocer. Por ejemplo, si utilizáramos la categoría de diccionarios, a CATEGORY se le daría el valor de Diccionarios y en CATEGORY CONTENTS se incluirían:

- Diccionarios, como colecciones que asocian llaves con valores. Propósito.
- Creación de diccionarios con o sin valores.
- Operaciones para manipular diccionarios: agregar, modificar y eliminar elementos del diccionario.
- Obtención de valores por medio de la llave: [k].
- Funciones sobre diccionarios: len, in.
- Eliminación de elementos con del.
- Recorrido de las llaves de un diccionario con for.
- Resolver problemas usando los diccionarios para contar/categorizar elementos.
- Resolver ejercicios en donde se mezclen estructuras de datos: listas de diccionarios, diccionarios con valores con listas de listas, diccionarios con otros diccionarios como valores, etc.

Finalmente, se le recordó en varias ocasiones que debía seguir el formato de los ejercicios en JSON. Esto se reiteró constantemente porque al darle más instrucciones, a veces dejaba de seguir ciertas reglas para cumplir otras.

[14] https://leetcode.com/

Usuario

Provided the context, category, difficulty level, and the common mistakes that need to be avoided, I need an exercise generated under these constraints. However, remember the following instructions:
The exercise should revolve strictly around the context and the category provided.
Pay careful attention to the common mistakes, the exercise should serve as a corrective measure towards them.
The output needs to be a valid JSON object with the following keys and value type:

- title : <str>
- description : <str>
- observations : <str>
- inputDescription : <str>
- outputDescription : <str>

Make sure that the input of the exercise can be taked into a list on python with ease, and try to put the example of the input on inputDescription, because they cannot use the eval() function.
Ensure that all the floating point numbers in the JSON output should be rounded to 1 decimal place.
Remember to give the exercise on spanish.
Remember to stick the exercise to categories mentioned before, don't use anything more than that.
Here are the details for the exercise

- Context : {Context}
- Category : {Category}
- Difficulty : {Difficulty}
- Student Errors : {Student Errors}

I know you really like to tell me what you did and why, but don't do that. I don't care. Your reponse should be only the exercise you wrote on the specified format i told you. No additional text, not before, not after.

Figura 3: Petición del Usuario
Fuente: Elaboración propia.

El usuario es el *prompt* que se entrega para solicitar el ejercicio. A este se le proporcionan diferentes parámetros: Contexto, Categoría, Dificultad y Errores del estudiante. El Contexto se refiere a la temática relacionada con el ejercicio, seleccionada aleatoriamente en el mismo servidor. La Categoría corresponde a la unidad de la asignatura de programación de la UTFSM relacionada con el ejercicio. La Dificultad es un número entre 1 y 10 que indica el nivel de dificultad del ejercicio. Los Errores del estudiante son obtenidos de la base de datos y generados por la memoria de Matías González; estos son errores que el estudiante ha cometido en códigos que ha subido a la plataforma de SMOJ.

Existen varios puntos en el *prompt*. Algunos sirven para recordar instrucciones al modelo, como la salida en formato JSON y que el ejercicio debe estar en español. También se mencionan características que todos los ejercicios deben tener en consideración. La primera es explicitar que no se puede usar la función *eval()*. De esta manera, la entrada de los ejercicios es texto que un estudiante puede manejar. Sin esta aclaración, las entradas de los programas serían listas o diccionarios escritos de manera textual, como por ejemplo [1, 2, 3, 4, 5] o { Llave1: Valor1, Llave2: Valor2 }.

Finalmente se le pide al modelo que no entregue mas texto del que tiene que generar, este texto que suele agregar son frases cortas, pero provocan que no se pueda formatear el código en las llaves que se piden.

3.2.3. Revisión de los ejercicios generados

Muchos de los ejercicios generados presentaban errores, como errores en el enunciado, ejemplos incorrectos, y problemas ortográficos y gramaticales. Con este objetivo se hace otra petición con otro Sistema y Usuario, esto con el propósito de revisar todo el ejercicio generado y que otro modelo lo revise y repare todo lo que encuentre incorrecto. Este paso mejoró significativamente los errores gramaticales que tenia el ejercicio, además de arreglar ejercicios que no tenían ningún problema a resolver o ninguna forma clara de resolver los problemas.

Para esta sección se utilizó el nuevo modelo de OpenAI, Chat-GPT 4o. Primero se intentó usar el modelo 3.5, pero este modelo asumía que los ejercicios entregados estaban completamente correctos y no realizaba ninguna modificación. Por esta razón, se decidió cambiar al modelo 4. Con este cambio, los ejercicios dejaron de tener problemas. Sin embargo, el modelo 4 es considerablemente más costoso que el modelo 3. Para mantener la calidad del modelo 4 con el costo del modelo 3, se comenzó a usar el modelo 4o, que combina la calidad del modelo 4 con el precio del modelo 3.

Sistema

You are a group of Computer Science 1 teachers that speaks spanish, with the job of improving the exercise that is given to you by other professors, you only modify the content that is inside the JSON parameters so that it can be sended via query.
Because you are answering you dont have to write anything beside the modifications and improvements to the exercise that is going to be between <exercise></exercise>, you dont add those on the answer, only the content between them.
Improve the exercise in any way you see benefitial so it is more easy to understand for the student and be sure that the exercise that you return can be solved with code and has the proper instructions to do, if it doesn't has it you have to add them, even if it means adding mathematical equations, you can change all of the content.

Figura 4: Rol Sistema para revisar ejercicios.
Fuente: Elaboración Propia

El modelo debe cumplir el rol de un sistema similar a un grupo de profesores, asegurando la calidad en la generación de ejercicios. Se le indica que modifique únicamente el contenido del JSON, sin alterar el formato preexistente ni agregar información adicional. También se le solicita que mejore los ejercicios de cualquier manera que considere conveniente. Estas mejoras generalmente consisten en mejorar la redacción de los ejercicios, pero en algunos casos se requiere corregir errores en las instrucciones que pueden causar problemas irresolubles. Con estas mejoras, se reduce la cantidad de ejercicios problemáticos.

Usuario

Can you improve the following exercise and make it a little bit harder to solve, and please make sure that the example given on the exercise is correct <exercise>{exercise} </exercise>on spanish
Please, make sure the code you write is complete, you don't miss any line, branch or statement.
Also remember to only modify the exercise that is between the special brackets ><exercise»</exercise>and only return me the JSON that i sent you so that a program on PHP can easily read it and use json_decode() on it, remember to remove the special brackets <exercise></exercise>when you are giving it back, nothing else.
I know you want to give me your opinion or how you did or what you changed, but i dont care so dont write that, just give me the code

Figura 5: Petición del usuario para revisar y corregir el ejercicio.
Fuente: Elaboración Propia

En este *prompt* se solicita mejorar la dificultad de un ejercicio dado. La petición se debe a que los ejercicios generados presentaban una baja dificultad, especialmente aquellos de niveles más bajos, los cuales resultaban ser monótonos y carecían de un verdadero desafui. Por ejemplo, algunos ejercicios consistían en recibir una lista de números como entrada y simplemente imprimir esa misma lista en el mismo orden. Debido a la falta de dificultad de estos ejercicios, se pidió hacerlos más desafiantes. Usando el ejemplo anterior, en lugar de solo imprimir la lista, ahora el ejercicio requeriría imprimir la lista ordenada.

3.2.4. Generación de casos de pruebas con LLMs

Para generar los casos de prueba, se realizan tres peticiones sucesivas, con diferentes modelos en cada una. Esto aumenta el tiempo de ejecución total de la solución, pero mejora significativamente la calidad de los casos de prueba. El flujo para generarlos es el siguiente: primero se genera el código para resolver el problema planteado en la sección anterior; luego se solicita la generación de diferentes casos de prueba, utilizando el código generado junto con el enunciado del ejercicio; finalmente, se revisan los casos de prueba para asegurarse de que estén en el formato correcto y cumplan con las instrucciones del ejercicio.

Para la generación del código se utilizó el modelo GPT-4, ya que el modelo GPT-3.5 generaba código incorrecto o no manejaba adecuadamente los casos borde de los ejercicios. Sin embargo, para la generación de los casos de prueba del código generado, se empleó el modelo GPT-3.5-turbo, dado que no se requiere tanta calidad en el texto ni un razonamiento complejo; solo es necesario generar parámetros de manera aleatoria según ciertas reglas proporcionadas.

Sistema

You are a software engineer with a passion for competitive programming and i'm going to give you an exercise between <input></input>. You have to generate the code to solve the exercise correctly in Python. I know you want to give me your opinion on how you did it or why, but I'm not interested. Just follow the format I ask for and generate the code, nothing else, no extra text or spaces or anything.

Figura 6: Rol del sistema en generación de código.
Fuente: Elaboración Propia

Se solicita al sistema que funcione como un ingeniero de software apasionado por la programación competitiva. El propósito de esta función es garantizar que todos los casos de prueba generados sean similares a los que se podrían encontrar en entornos de producción. aclarándole que el problema que tiene que solucionar sera entregado entre <input></input>, de esta forma el modelo sabe exactamente cual es el problema y se genera menos confusión.

Usuario

Can you give me the code for the following exercise<input> exercise </input>
Please, make sure the code you write is complete, you don't miss any line, branch or statement. Also remember to only modify the test cases that are between the special brackets <input></input>and only return me the test cases without the special brackets, nothing else. I know you want to give me your opinion or how you did or what you changed, but i dont care so dont write that, just give me the code

Figura 7: Petición del usuario en generación de código.
Fuente: Elaboración Propia

Finalmente, el *Prompt* que se le entrega es ya con el ejercicio y lo único que se le tiene que recordar es mantener el formato y no responder nada mas, es necesario repetirlo siempre al final, ya que los modelos de Chat-GPT suelen escribir algo entre las respuestas que dan.

Luego de tener generado el código del ejercicio, se procede a generar los casos de pruebas con los siguientes Sistemas y Usuario.

Sistema

You are a software engineer especiallized in writing automated tests for a given exercise and code solution, you are always ensuring that what you return can be used on an online judge to test the given exercise,
these test cases should always be correct in terms of the output and input. You need to provide at least 7 border test cases for the given exercise,
Remember to challenge the exercise with intriguing test cases that scrutinize every possible mistake to ensure the student's code doesn't just work, so they can be as long as you want with a maximum of 150 characters of input and 150 characters of output,
but it's also robust and efficient. if the input is a list it must always be of 2 parts, the first one being an integer n that is the length of the list and the second part is the list in the format
item1
item2
item3
...
nitem
but the output doesn't follow those rules, so it has to be like [item1,item2,...,itemN], also if the outputs are float, they shall be rounded at 1 decimal.
The inputs should be provided as stdin strings, and the outputs should be in a string format immitating the output of the program.
Use the following format for all test cases: 'testN':{'input':<str>, 'output':<str>}, where N is the test case number, and <str>is a string representing the input or output.
The code is going to be inside of <code></code>and the exercise is going to be between <exercise></exercise>

Figura 8: Rol del sistema para generación de casos de prueba.
Fuente: Elaboración Propia

Luego de generar el código del ejercicio, se hace otra *prompt*. Esta vez, se explicitan las características que deben tener todos los casos de prueba y el formato que estos deben seguir. Para facilitar la lectura, el formato usado es JSON. Las características que deben tener todos los casos de prueba son: un máximo de 150 caracteres en la entrada y salida del programa, que los casos de prueba sean correctos y puedan ser usados para probar los códigos con profundidad.

Una característica específica que se menciona es la forma de entregar listas como entradas. Puesto que usualmente se entregan listas de manera textual (ej. [1, 2, 3, 4, 5]), se pide que se entreguen como son usualmente presentadas en SMOJ, pero no se aplica la misma regla para la salida del programa. Además se tienen que redondear todos los números de punto flotante a 1 decimal, esta regla se hace para eliminar diferencias que puedan existir en diferentes métodos de cálculos.

Usuario

Write tests for the following exercise:
<exercise>{exercise}</exercise><code>{Code}</code>
Please, make sure the code you write is complete and remember to follow the json format 'testN':{'input':<str>, 'output':<str>}.
Make sure you don't miss any line, branch or statement. and no extra text

Figura 9: Petición del usuario de generación de casos de pruebas.
Fuente: Elaboración Propia

Se usa esta *Prompt*, entregándole el ejercicio generado en la sección anterior y el código generado por el modelo en el *Prompt* anterior.

3.2.5. Revisión de los casos de pruebas generados

Se revisan los casos de pruebas para asegurarse del formato y de que los casos de pruebas funcionen de manera correcta, esta revision es obligatoria, puesto que muchas veces se generan casos de pruebas erroneos y al pasarlos por otro modelo los arregla.El modelo que se uso para esta seccion es Chat-GPT 4o, por el bajo costo y un buen funcionamiento en general.
Se usaron los siguientes roles de Sistema y Usuario:

Sistema

You are a lead tester software engineer with the role of improving the test cases given to you on python, you can modify them or improve them following the exercise rules.
The output needs to be a valid JSON object with the following keys: 'testN':{'input':<str>, 'output':<str>} so if you see that they dont follow that format you have to correct it, you do not modify the structure of the text, not give your opinion. just modify the code that is given to you

Figura 10: Rol del sistema de revisión de casos de pruebas.
Fuente: Elaboración Propia

Al igual que en las secciones de generación de casos de pruebas, se solicitó al Sistema que funcionara como un ingeniero de software especializado en pruebas de software. Su tarea era únicamente modificar los ejercicios y asegurarse del formato, sin alterar ni agregar más texto del necesario.

Usuario

Can you improve the test cases that i made for the following exercise: {exercise}
These are the test cases that i made: <input>{testCases} </input>
Please, make sure the code you write is complete, you don't miss any line, branch or statement.
Also remember to only modify the test cases that are between the special brackets <input></input>and only return me the test cases without the special brackets, nothing else. I know you want to give me your opinion or how you did or what you changed, but i dont care so dont write that, just the change the code.

Figura 11: Petición de usuario para revisar los casos de pruebas.
Fuente: Elaboración Propia

Se proporciona a Usuario el ejercicio junto con los casos de prueba generados para su revisión. En el sistema, se establece la necesidad de mejorar únicamente los casos de prueba, sin añadir texto innecesario

VALIDACIÓN DE LA SOLUCIÓN

4.1. Metodología

Para validar la solución presentada, se realizaron dos actividades distintas:

- Implementación de la plataforma en una actividad presencial con estudiantes.
- Distribución de ejercicios generados con diferentes características a un grupo de profesores y ayudantes de la asignatura de programación.

4.2. Pruebas con Estudiantes

4.2.1. Contexto

Para las pruebas con estudiantes, se modificó la plataforma de SMOJ de manera que los estudiantes solo pudieran acceder a la sección de generación de ejercicios con LLMs y a sus propios envíos. Esta restricción se implementó para evitar que los estudiantes consultaran información de otros usuarios o revisaran ejercicios preexistentes en la plataforma, incluyendo sus diferentes casos de prueba.

Con el objetivo de mantener la coherencia y asegurar que los ejercicios generados fueran útiles para los estudiantes, se seleccionó la unidad de Diccionarios. Al LLM se le proporcionó toda la información sobre lo que los estudiantes sabían hacer con Diccionarios y ejercicios similares que habían resuelto en el SMOJ de la semana o en otras ayudantías. Esto facilitó la creación de ejercicios con características y métodos de resolución similares.

4.2.2. Encuesta pre-experiencia

Para determinar la percepción de los estudiantes, se diseñó una encuesta en Google Forms que debían completar antes y después de la actividad. El propósito era medir cómo se sentían los estudiantes respecto a sus habilidades de programación antes de probar la plataforma y comparar estos sentimientos después de la experiencia. Las preguntas de la encuesta fueron las siguientes:

Tabla 1: Preguntas pre experiencia.

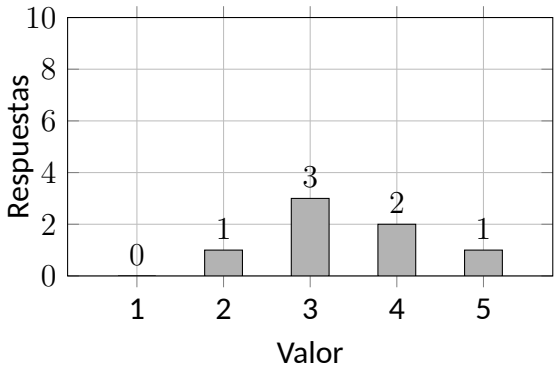
Fuente: Elaboración Propia.

Número	Pregunta
1	Me siento capaz de resolver problemas de programación.
2	Me aseguro de entender el problema completo antes de empezar a programar.
3	Soy capaz de predecir el comportamiento y la salida de un programa solo con leer el enunciado.
4	Puedo explicar a otra persona mis soluciones a los problemas de programación.
5	Me resulta difícil arreglar mi código si no recibo retroalimentación.
6	Sientes que los ejercicios propuestos en SMOJ tienen una dificultad acorde a tus capacidades.
7	Cuándo te encuentras atascado/a en un ejercicio, ¿Cuál es la razón en la mayoría de los casos?

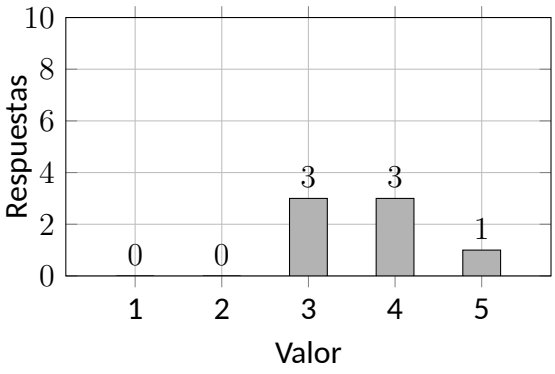
En este caso todas las preguntas fueron con respuesta de 1 a 5 a excepción de la ultima pregunta que es un texto de respuesta larga

4.2.3. Resultados encuesta pre experiencia

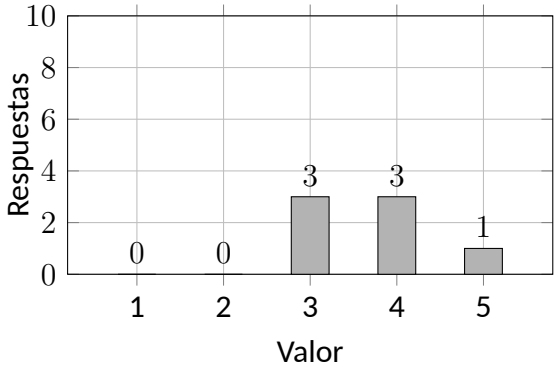
Las respuestas de las preguntas son valores entre 1 al 5, asociando el valor de 1 a 'muy en desacuerdo' y el valor del 5 a 'Muy de acuerdo'. Las respuestas serán mostradas en gráficos de barras mostrando la cantidad de respuestas que existen entre los valores del 1 al 5.



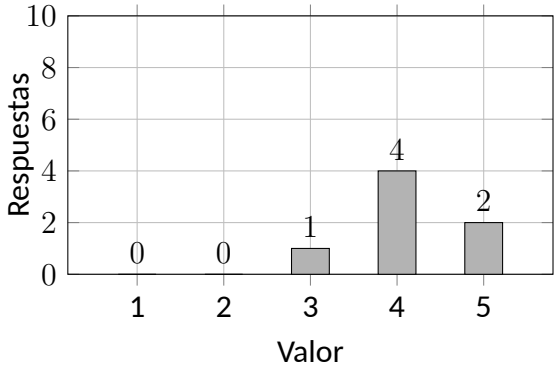
(a) Gráfico de las respuestas pregunta 1



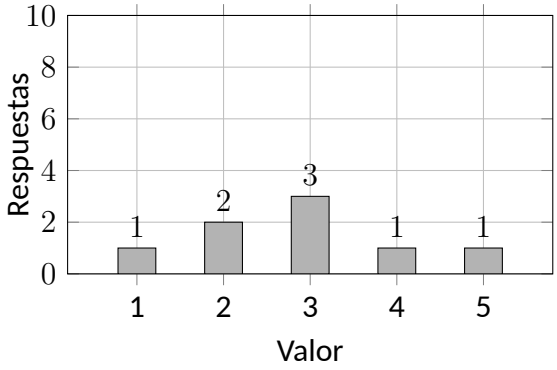
(b) Gráfico de las respuestas pregunta 2



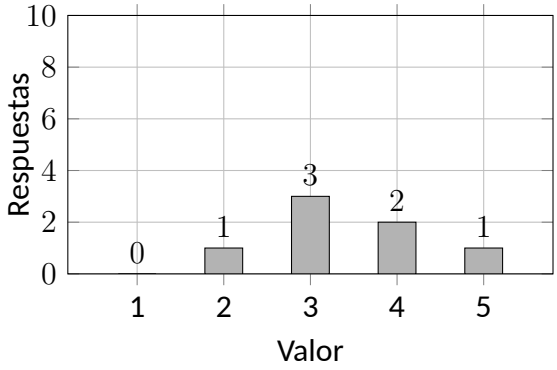
(a) Gráfico de las respuestas pregunta 3



(b) Gráfico de las respuestas pregunta 4



(a) Gráfico de las respuestas pregunta 5



(b) Gráfico de las respuestas pregunta 6

En referencia a la pregunta número 7, que es de respuesta abierta, la mayoría de los estudiantes mencionaron dificultades para entender correctamente lo que se pide en el enunciado o para relacionar la información del enunciado con lo que se debe crear en el programa. Otro tema comúnmente mencionado fue la incapacidad para escribir el código y plasmar la solución desde el pensamiento al código. Muchos atribuyeron esto a la falta de dominio de la materia, mientras que otros lo asociaron a su propia falta de creatividad.

4.2.4. Preguntas post experiencia

Tabla 2: Preguntas post Experiencia.

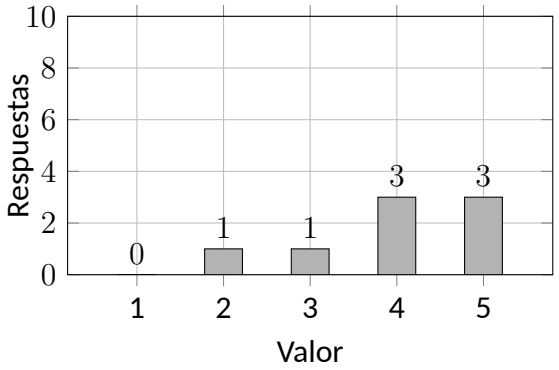
Fuente: Elaboración Propia.

Número	Pregunta
1	Me siento capaz de resolver problemas de programación.
2	Me aseguro de entender el problema completo antes de empezar a programar.
3	Soy capaz de predecir el comportamiento y la salida de un programa solo con leer el enunciado.
4	Puedo explicar a otra persona mis soluciones a los problemas de programación.
5	Me resulta difícil arreglar mi código si no recibo retroalimentación.
6	Sientes que los ejercicios propuestos en la experiencia tienen una dificultad acorde a tus capacidades.
7	Disfrutaste la experiencia del día de hoy
8	Creo que la actividad del día de hoy fue bastante útil para mis estudios
9	¿Qué te pareció la experiencia? ¿Alguna nota o sugerencia?

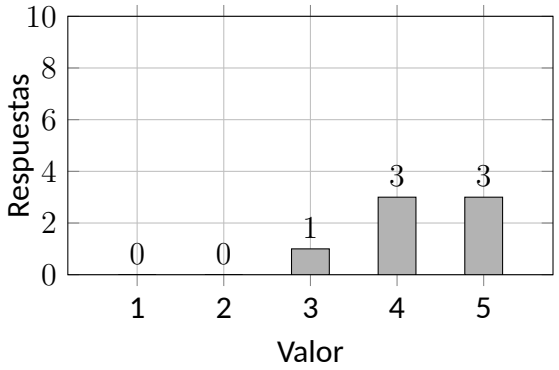
En este caso todas las preguntas fueron con respuesta de 1 a 5 a excepción de la ultima pregunta que es de respuesta abierta.

4.2.5. Resultados encuesta post experiencia

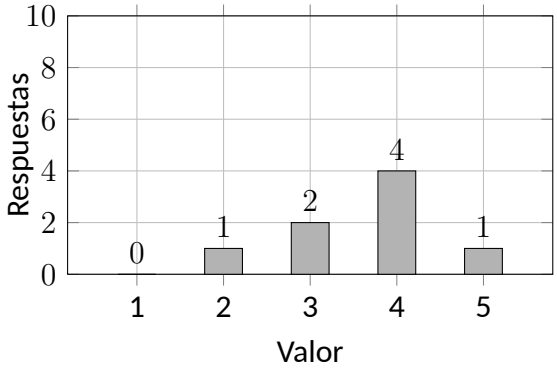
Las respuestas de las preguntas son valores entre 1 al 5, asociando el valor de 1 a 'muy en desacuerdo' y el valor del 5 a 'Muy de acuerdo'. Las respuestas serán mostradas en gráficos de barras mostrando la cantidad de respuestas que existen entre los valores del 1 al 5.



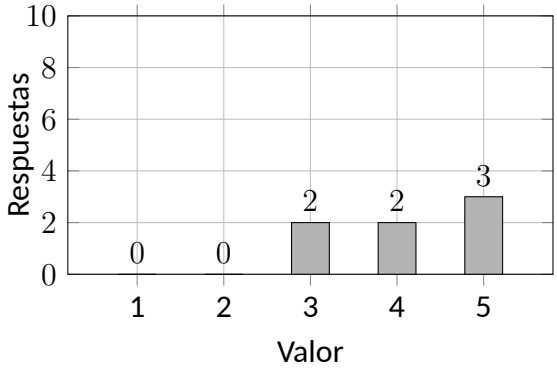
(a) Gráfico de las respuestas pregunta 1



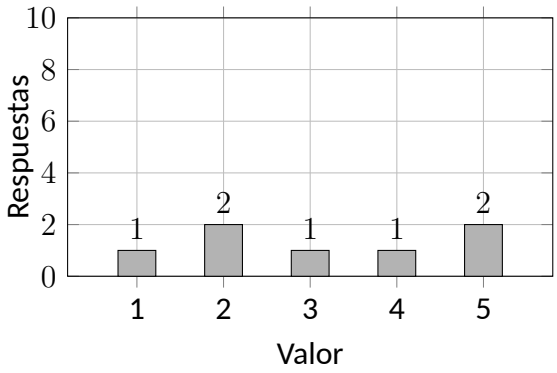
(b) Gráfico de las respuestas pregunta 2



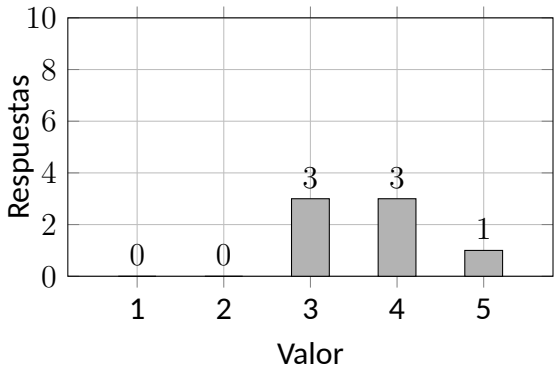
(a) Gráfico de las respuestas pregunta 3



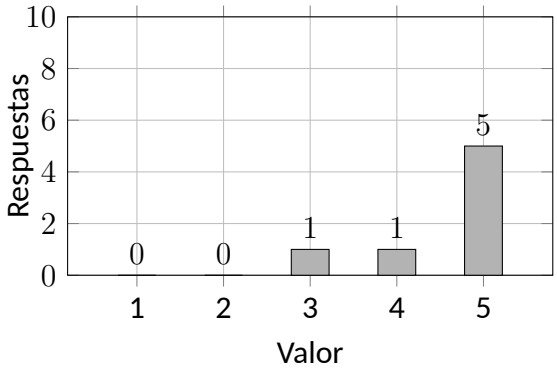
(b) Gráfico de las respuestas pregunta 4



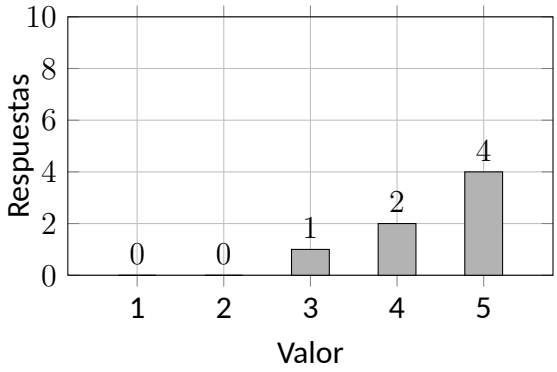
(a) Gráfico de las respuestas pregunta 5



(b) Gráfico de las respuestas pregunta 6



(a) Gráfico de las respuestas pregunta 7



(b) Gráfico de las respuestas pregunta 8

Mencionando las respuestas sobre la experiencia, todos los estudiantes encontraron la plataforma y la experiencia muy interesante. Les encantó la capacidad de obtener ejercicios con temáticas de la cultura pop y con niveles de dificultad adecuados a su comodidad. También apreciaron la posibilidad de estudiar desde casa con una gran cantidad de ejercicios disponibles.

Sin embargo, también mencionaron algunos problemas que encontraron tanto en los ejercicios como en la plataforma. Uno de los problemas era que la plataforma generaba un tiempo de espera y no redirigía a la página del ejercicio. Otro problema que los estudiantes señalaron fue la forma en que se redactaban los enunciados. Consideraron que las instrucciones no eran muy claras o que tenían ciertos vacíos.

En general, se puede observar que todos los gráficos tienden a mostrar resultados más cercanos a 5 en comparación con la pre-experiencia. Esto demuestra que, después de completar los ejercicios y la ayudantía, los estudiantes se sintieron más confiados en sus respuestas y en los códigos que escribían, mostrando un aumento en su autoeficacia y en la percepción de sus propias habilidades para comprender y resolver problemas.

4.3. Encuesta de Docentes

4.3.1. Metodología

Para determinar la calidad de los ejercicios generados por la plataforma, se tomó la opinión de tres profesores y 1 ayudante del curso IWI-131. El propósito fue evaluar si los ejercicios generados tenían las dificultades solicitadas o si existían discrepancias. Además, se pidió la opinión de los profesores sobre cada ejercicio, para conocer sus impresiones.

Para este fin, se generaron tres ejercicios con diferentes características, evitando errores comunes de los estudiantes para asegurar que fueran ejercicios nuevos generados por la plataforma. Los ejercicios tenían niveles de dificultad de 3, 6 y 9, recordando que la dificultad máxima es de 10.

Las preguntas formuladas fueron las siguientes:

Tabla 3: Preguntas para Docentes.

Fuente: Elaboración Propia.

Número	Pregunta
1	Dificultad que usted cree que debería tener el ejercicio número 1.
2	¿Que opina del Ejercicio número 1?
3	Dificultad que usted cree que debería tener el ejercicio número 2.
4	¿Que opina del Ejercicio número 2?
5	Dificultad que usted cree que debería tener el ejercicio número 3.
6	¿Que opina del Ejercicio número 3?
7	¿Cree que el programa es útil para mejorar el aprendizaje de los estudiantes?
8	¿Cuál es su opinión general de la plataforma?

Las preguntas 1, 3 y 5 son de escala del 1 al 10 para determinar la dificultad percibida por los

docentes. Las demás preguntas son de respuesta libre, permitiendo a los docentes expresar sus opiniones detalladamente.

4.3.2. Preguntas generadas con la plataforma

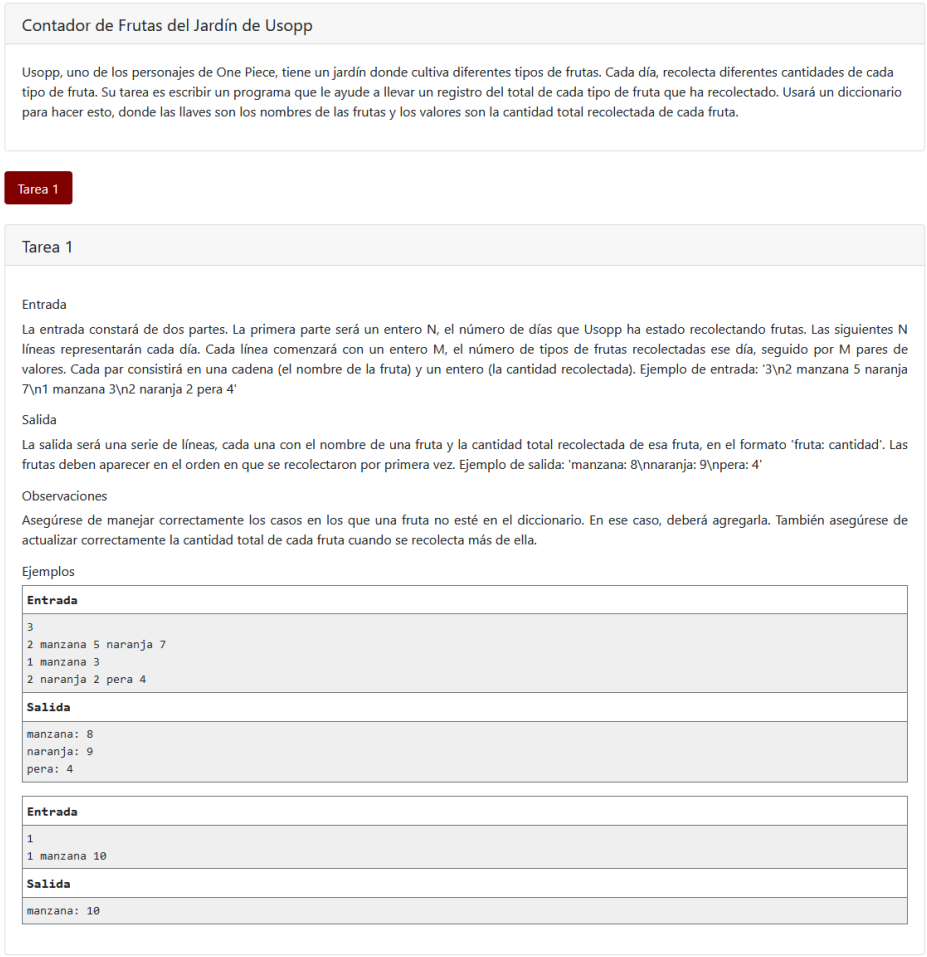


Figura 19: Pregunta 1 dificultad 6 generada con el sistema.
Fuente: Elaboración propia.

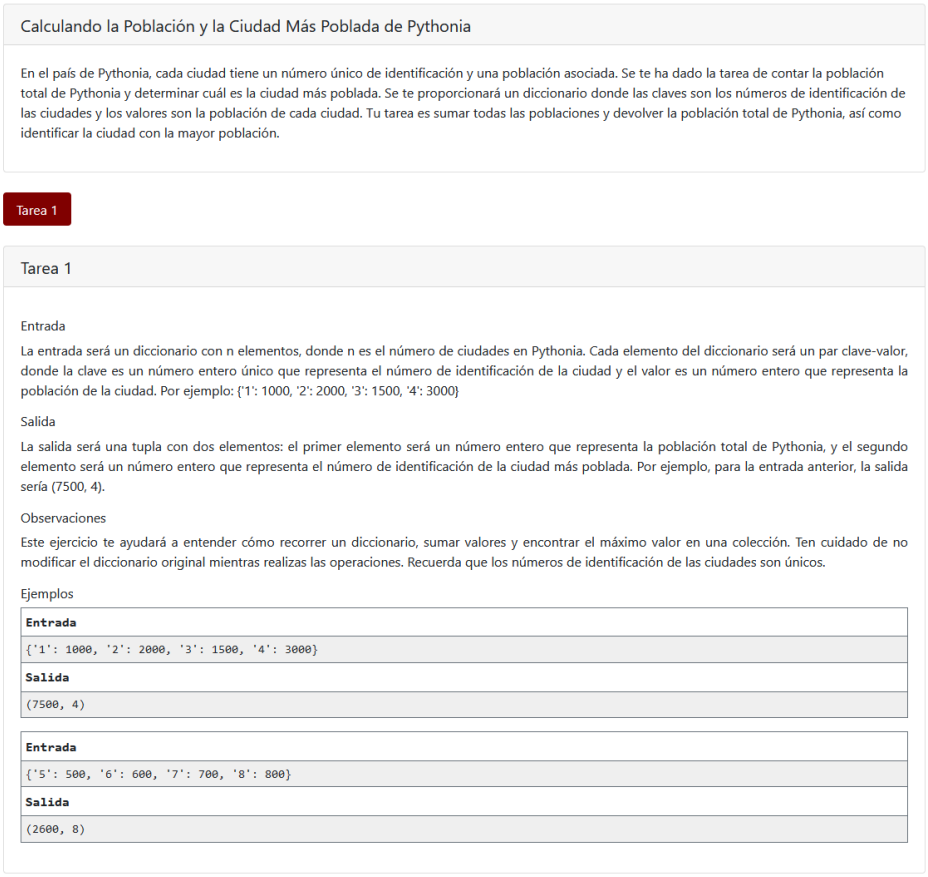
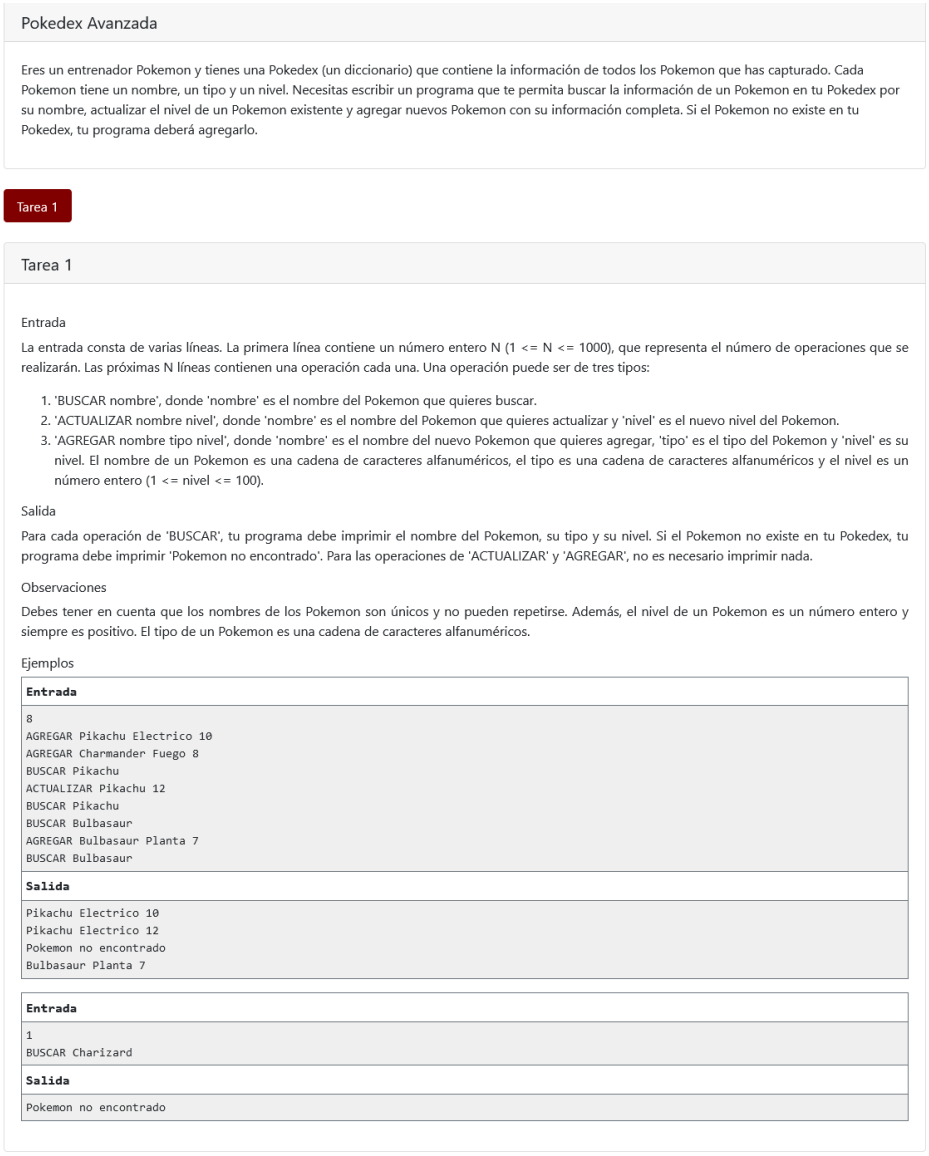


Figura 20: Pregunta 2 dificultad 3 generada con el sistema.
Fuente: Elaboración propia.



Pokedex Avanzada

Eres un entrenador Pokemon y tienes una Pokedex (un diccionario) que contiene la información de todos los Pokemon que has capturado. Cada Pokemon tiene un nombre, un tipo y un nivel. Necesitas escribir un programa que te permita buscar la información de un Pokemon en tu Pokedex por su nombre, actualizar el nivel de un Pokemon existente y agregar nuevos Pokemon con su información completa. Si el Pokemon no existe en tu Pokedex, tu programa deberá agregarlo.

Tarea 1

Tarea 1

Entrada

La entrada consta de varias líneas. La primera línea contiene un número entero N (1 <= N <= 1000), que representa el número de operaciones que se realizarán. Las próximas N líneas contienen una operación cada una. Una operación puede ser de tres tipos:

1. 'BUSCAR nombre', donde 'nombre' es el nombre del Pokemon que quieres buscar.
2. 'ACTUALIZAR nombre nivel', donde 'nombre' es el nombre del Pokemon que quieres actualizar y 'nivel' es el nuevo nivel del Pokemon.
3. 'AGREGAR nombre tipo nivel', donde 'nombre' es el nombre del nuevo Pokemon que quieres agregar, 'tipo' es el tipo del Pokemon y 'nivel' es su nivel. El nombre de un Pokemon es una cadena de caracteres alfanuméricos, el tipo es una cadena de caracteres alfanuméricos y el nivel es un número entero (1 <= nivel <= 100).

Salida

Para cada operación de 'BUSCAR', tu programa debe imprimir el nombre del Pokemon, su tipo y su nivel. Si el Pokemon no existe en tu Pokedex, tu programa debe imprimir 'Pokemon no encontrado'. Para las operaciones de 'ACTUALIZAR' y 'AGREGAR', no es necesario imprimir nada.

Observaciones

Debes tener en cuenta que los nombres de los Pokemon son únicos y no pueden repetirse. Además, el nivel de un Pokemon es un número entero y siempre es positivo. El tipo de un Pokemon es una cadena de caracteres alfanuméricos.

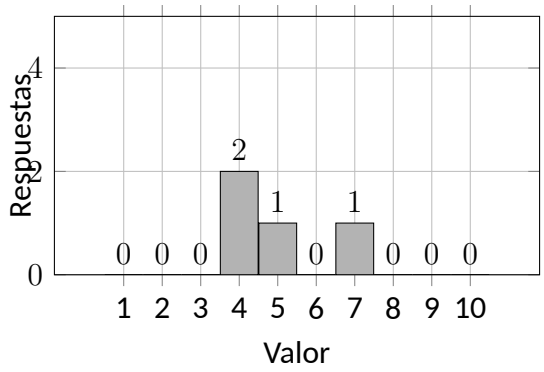
Ejemplos

Entrada
8 AGREGAR Pikachu Electrico 10 AGREGAR Charmander Fuego 8 BUSCAR Pikachu ACTUALIZAR Pikachu 12 BUSCAR Pikachu BUSCAR Bulbasaur AGREGAR Bulbasaur Planta 7 BUSCAR Bulbasaur
Salida
Pikachu Electrico 10 Pikachu Electrico 12 Pokemon no encontrado Bulbasaur Planta 7

Entrada
1 BUSCAR Charizard
Salida
Pokemon no encontrado

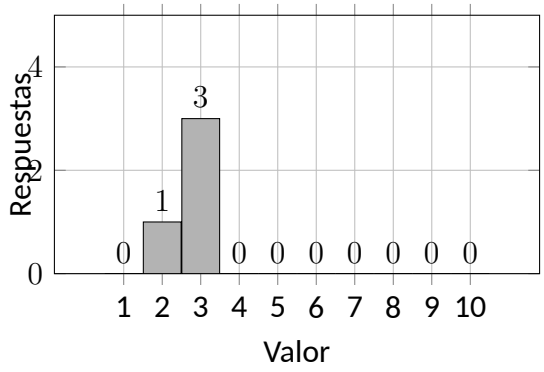
Figura 21: Pregunta 3 dificultad 9 generada con el sistema.
Fuente: Elaboración propia.

4.3.3. Respuestas de Docentes



(a) Gráfico de las respuestas pregunta 1

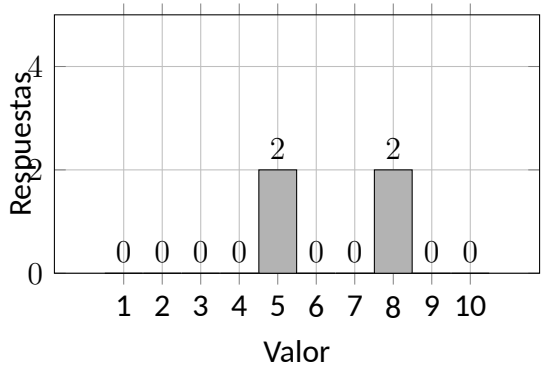
Varios de los profesores encontraron que el ejercicio tenia una dificultad media, ninguno acertó en la verdadera dificultad del ejercicio, pero dieron respuestas bastante similares. En lo que opinaron los profesores sobre este ejercicio, todos pudieron concluir que la dificultad del ejercicio radica en el comprender como interactuar con la entrada, además de usar un sistema de listas y diccionarios para poder construir la solución. También opinaron que la redacción puede ser confusa y que los estudiantes podrían no entender que significa el \n en la entrada de ejemplo, pero este es un problema de como SMOJ muestra el lenguaje de Markdown, puesto que al imprimirlo en otras paginas no muestra el símbolo, sino que muestra un salto de linea.



(a) Gráfico de las respuestas pregunta 2

Mencionando la dificultad estimada de la pregunta 2, el 75 % de los docentes pudieron acertar correctamente la dificultad del ejercicio, esto demuestra de que el ejercicio generado en dificultades bajas si se acerca a lo que en la realidad corresponde a una dificultad de 3. También, todos los profesores indicaron que el ejercicio seria de mucha utilidad para una persona que este recién empezando a trabajar con la unidad de diccionarios. Pero el ejercicio hace uso de la unidad de tuplas, unidad que ya no se ve en el curso de IWI-131, esto podría ser

provocado por el modelo uniendo los conceptos de listas y tuplas como una sola categoría, porque en ningún momento se le agrego la unidad ni conceptos de tupla al modelo con el que se generaron los ejercicios.



(a) Gráfico de las respuestas pregunta 3

Considerando las respuestas de la pregunta 3, aproximadamente el 50 % de los docentes respondieron una dificultad bastante cercana a la original, pero el otro 50 % se alejo bastante, diciendo que el ejercicio 3 tiene una dificultad de 5. La razón de esto se puede concluir de la opinión de los docentes en los ejercicios. En general todos los docentes lo encontraron como un ejercicio mas abordable y que los estudiantes que entiendan diccionarios no deberían tener problemas para poder solucionarlo, pero lo encontraron como un ejercicio mas largo que complejo, asociándolo mas a una tarea que a un ejercicio común de SMOj.

Mencionando las respuestas de la pregunta 7, todos los profesores y ayudante que contestaron les gusto la idea de la plataforma, les gusto bastante la idea de ampliar la cantidad de ejercicios que los estudiantes pueden resolver, sumando al menos 3 ejercicios a la semana, junto a los 4 a 6 ejercicios que se ofrecen oficialmente en SMOJ cada semana. También comentaron la alternativa de que los estudiantes puedan resolver ejercicios que estén en los niveles que ellos pueden resolver con una experiencia personalizada.

Finalmente, en relación con la pregunta número 8, en general, la plataforma fue bien recibida. Varios participantes manifestaron su agrado por la idea de mejorar las tecnologías ya asociadas a SMOJ, lo cual permite que tanto estudiantes como profesores sientan familiaridad con su uso. Un comentario positivo adicional fue la posibilidad de utilizar los ejercicios generados para crear más ejercicios en la plataforma, lo que contribuiría a reducir la carga de trabajo de profesores y ayudantes.

En cuanto a las áreas de mejora señaladas, se mencionó que los enunciados generados para los ejercicios pueden resultar excesivamente extensos en comparación con los estándares habituales de SMOJ. Asimismo, se observó que algunos ejercicios podrían ser redundantes al ser generados mediante un LLM, sugiriendo la incorporación de intervención humana para mejorar la calidad de los ejercicios.

4.4. Limitaciones

Todos estos resultados fueron expuestos a diferentes sesgos o limitaciones propios del contexto en el que se trabajaron, estas condiciones pueden modificar los resultados completamente si se fueran a cambiar. Las limitaciones experimentadas en estas pruebas fueron:

- Baja participación de estudiantes, debido a la existencia de un examen de matemáticas en el siguiente bloque de horario, la participación de estudiantes fue mas baja en comparación a una clase normal.
- Debido a un examen de matemáticas que hubo en el horario anterior, los estudiantes que asistieron eran personas que buscaban una mayor nota y comprensión de la asignatura, por lo que no eran estudiantes que fueron por tener que asistir.
- Asistieron 2 informáticos y 5 estudiantes de otras carreras, pero los informáticos suelen tener una mayor autoeficacia en los temas de programación.
- Como la experiencia se realizó en una clase, el tiempo máximo para resolver ejercicios fueron de 70 minutos, esto provocó que no pudieran resolver tantos ejercicios en comparación a si hubieran tenido una tarde con el sistema.

CAPÍTULO 5
CONCLUSIONES

5.1. Conclusiones Generales

La plataforma en la que se implementó la memoria fue la plataforma de SMOJ, donde se creó una página especial que se comunica con un servidor que genera diferentes ejercicios y sus casos de pruebas. SMOJ solo debe entregar la categoría, dificultad y errores comunes del estudiante al servidor para poder generar el ejercicio. Este servidor generador de ejercicios funciona con Python 3, LangChain y la API de OpenAI para sus modelos, como ChatGPT 3.5 y 4.

Para conocer la recepción por parte de los estudiantes y la complejidad de los ejercicios generados por la plataforma, se realizó una actividad en una ayudantía de asignatura de programación, en la cual los estudiantes pudieron acceder a la plataforma y generar un máximo de tres ejercicios por estudiante. Se les pidió que contestaran una encuesta antes y después de la actividad para medir su autoeficacia y su opinión acerca de la plataforma y los ejercicios generados.

En la encuesta post-experiencia se puede notar cómo mejoró la percepción de los estudiantes sobre sus propias habilidades de programación, mostrando que lograron aumentar su autoeficacia al comenzar resolviendo ejercicios más sencillos, de dificultades 4 o menores, y lograron sentirse cómodos con la materia que estaban estudiando. También se recibieron sugerencias y críticas, entre ellas, el alto tiempo que se demora el sistema en generar los ejercicios y la complejidad de los enunciados.

Para conocer la opinión de los profesores y ayudantes del Departamento de Informática de la Universidad Técnica Federico Santa María, se les entregó una encuesta con tres ejercicios generados por la plataforma, para conocer sus opiniones sobre la plataforma y la idea en general. Todos encontraron que la idea de implementar LLMs en SMOJ era interesante y, en general, les gustaron los ejercicios generados por la plataforma. Cabe mencionar que algunos profesores sintieron que los ejercicios podrían tener errores al generarse, pero que podrían usarse con la intervención de profesores o ayudantes para mejorar o arreglar los ejercicios en caso de que fallaran o faltara algo.

5.2. Trabajo a futuro

Existen diversos aspectos en los que se puede continuar el trabajo desarrollado, las tareas pendientes serían mejorar los *prompts* que se entregan a los modelos para generar los casos de prueba y el ejercicio. Mejorar estos *prompts* y reducir la cantidad de llamadas debería

reducir el tiempo que se demora en generar cada ejercicio con los casos de prueba. Además, mejorar los *prompts* debería mejorar la calidad de los ejercicios generados.

Actualmente, el servidor toma las peticiones con Gunicorn considerando 12 trabajadores para procesarlas, pero no está pensado para soportar una alta cantidad de peticiones al mismo tiempo. Por lo tanto, pensando en infraestructura, se podrían aumentar la cantidad de hilos por trabajador para poder tomar más peticiones al mismo tiempo.

REFERENCIAS BIBLIOGRÁFICAS

[Cheah, 2020] Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12:1–14.

[Council, 2021] Council, F. T. (2021). Rising stars of the tech world: Why developers are job market royalty. Accessed: [1-Oct-2023].

[Dale y Weems, 2016] Dale, N. y Weems, C. (2016). *Programming And Problem Solving With C++: Comprehensive*. Jones and Bartlett Publishers, Inc., USA, 6th edición.

[Demir, 2021] Demir, F. (2021). The effect of different usage of the educational programming language in programming education on the programming anxiety and achievement. *Education and Information Technologies*, 27(3):4171–4194.

[Drini, 2018] Drini, M. (2018). Using new methodologies in teaching computer programming. En *2018 IEEE Integrated STEM Education Conference (ISEC)*, pp. 120–124.

[Drori *et al.*, 2022] Drori, I., Zhang, S., Shuttleworth, R., Tang, L., Lu, A., Ke, E., Liu, K., Chen, L., Tran, S., Cheng, N., Wang, R., Singh, N., Patti, T. L., Lynch, J., Shporer, A., Verma, N., Wu, E., y Strang, G. (2022). A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32).

[Falkner y Sheard, 2019] Falkner, K. y Sheard, J. (2019). *Pedagogic Approaches*, pp. 445–480. Cambridge University Press.

[Feng *et al.*, 2020] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., y Zhou, M. (2020). CodeBERT: A pre-trained model for programming and natural languages. En *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1536–1547, Online. Association for Computational Linguistics.

[Fincher, 1999] Fincher, S. (1999). What are we doing when we teach programming? *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No.99CH37011*, 1:12A4/1–12A4/5 vol.1.

[Freitas *et al.*, 2023] Freitas, T. C., Neto, A. C., Pereira, M. J. V., y Henriques, P. R. (2023). Nlp/ai based techniques for programming exercises generation. volumen 112. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing.

[González, 2024] González, M. (2024). CreaciÓn de un entorno de aprendizaje basado en llms para mejorar la comprensiÓn de problemas computacionales y el nivel de retroalimentaciÓn en cursos iniciales de programaciÓn.

[Jayaseelan, 2023] Jayaseelan, N. (2023). Llama 2: The new open source language model. https://www.e2enetworks.com/blog/llama-2-the-new-open-source-language-model.

[Jordan y Mitchell, 2015] Jordan, M. I. y Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

[Kasneci *et al.*, 2023] Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., y Kasneci, G. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

[Kurnia *et al.*, 2001] Kurnia, A., Lim, A., y Cheang, B. (2001). Online judge. *Computers Education*, 36(4):299–315.

[LEYTON, 2022] LEYTON, H. (2022). Desarrollo de un sistema adaptativo de recomendaciÓn de ejercicios de correcciÓn automÁtica en un curso masivo de introducciÓn a la programaciÓn.

[Luo *et al.*, 2008] Luo, Y., Wang, X., y Zhang, Z. (2008). Programming grid: a computer-aided education system for programming courses based on online judge. En *Proceedings of the 1st ACM Summit on Computing Education in China on First ACM Summit on Computing Education in China*, SCE '08, New York, NY, USA. Association for Computing Machinery.

[MacNeil *et al.*, 2022] MacNeil, S., Tran, A., Leinonen, J., Denny, P., Kim, J., Hellas, A., Bernstein, S., y Sarsa, S. (2022). Automatically generating cs learning materials with large language models. pp. 1176–1176. Association for Computing Machinery (ACM).

[Maharaj *et al.*, 2022] Maharaj, S., Polson, N., y Turk, A. (2022). Chess ai: Competing paradigms for machine intelligence. *Entropy*, 24(4).

[OpenAI, 2022] OpenAI (2022). Introducing chatgpt. https://openai.com/blog/chatgpt.

[Palomba *et al.*, 2016] Palomba, F., Panichella, A., Zaidman, A., Oliveto, R., y Lucia, A. D. (2016). Automatic test case generation: What if test code quality matters? pp. 130–141. Association for Computing Machinery, Inc.

[Pichai, 2023] Pichai, S. (2023). An important next step on our ai journey. https://blog.google/technology/ai/bard-google-ai-search-updates/.

[R. Dijkstra; Z. Genç, 2022] R. Dijkstra; Z. Genç, S. K. J. K. (2022). Reading comprehension quiz generation using generative pre-trained transformers. https://pure.uva.nl/ws/files/122339904/itb22_p1_full5439.pdf.

[Sanders *et al.*, 2017] Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., y Zander, C. (2017). Folk pedagogy: Nobody doesn't like active learning. pp. 145–154. Association for Computing Machinery, Inc.

[Sarsa *et al.*, 2022] Sarsa, S., Denny, P., Hellas, A., y Leinonen, J. (2022). Automatic generation of programming exercises and code explanations using large language models. volumen 1, pp. 27–43. Association for Computing Machinery, Inc.

[Vatchkov *et al.*, 2007] Vatchkov, P., Zhechev, B., Rachev, B., Smrikarov, A., Dimov, D., Library., A. D., y for PhD Students in Computing (2007 : Ruse, B. W. (2007). *Proceedings of the 2007 International Conference on Computer Systems and Technologies : 2007, Bulgaria, June 14-15, 2007*. Association for Computing Machinery.

[Wakatani y Maeda, 2015] Wakatani, A. y Maeda, T. (2015). Automatic generation of programming exercises for learning programming language. *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, pp. 461–465.

[Wang *et al.*, 2019] Wang, C., Li, M., y Smola, A. J. (2019). Language models with transformers.

[Wei *et al.*, 2022] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., y Fedus, W. (2022). Emergent abilities of large language models.

[Whalley *et al.*, 2006] Whalley, J., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P., y Prasad, C. (2006). An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. *Eighth Australasian Computing Education Conference (ACE2006)*.

[Zhou *et al.*, 2018] Zhou, W., Pan, Y., Zhou, Y., y Sun, G. (2018). The framework of a new online judge system for programming education. En *Proceedings of ACM Turing Celebration Conference - China*, ACM TURC '18, p. 9–14, New York, NY, USA. Association for Computing Machinery.