Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

# GENERACIÓN AUTOMÁTICA DE CÓDIGO FUENTE A TRAVÉS DE MODELOS PREENTRENADOS DE LENGUAJE. CHATGPT: EVALUACIÓN Y APLICACIÓN

Bender, Adrián; Nicolet, Santiago; Folino, Pablo y López, Juan José

*Facultad de Ingeniería, Universidad del Salvador, Buenos Aires, Argentina*
*{bender.adrian, santiago.nicolet}@usal.edu.ar*

## RESUMEN

Este estudio explora las capacidades del modelo de lenguaje de inteligencia artificial ChatGPT, con la arquitectura GPT-3 desarrollada por OpenAI, en la tarea de generar código fuente en JavaScript a partir de instrucciones en español. Los modelos de lenguaje transformer, como exponentes del aprendizaje profundo, son eficaces en aprender representaciones contextuales de palabras y frases. Esto permite que el modelo comprenda no sólo los términos de programación individuales, sino también cómo se combinan en estructuras más grandes como bucles y funciones. A través de un conjunto de solicitudes únicas de funciones de programación de un conjunto de casos seleccionados, preparados para este trabajo, examinamos la capacidad del modelo para transcribir estas especificaciones de alto nivel en un código fuente ejecutable y funcional. Los resultados del modelo fueron evaluados en un compilador, buscando una evaluación objetiva de la funcionalidad del código generado en casos de prueba unitarios preparados de antemano. El modelo logra un 100% de código compilable y un 90% de resolución exitosa de los problemas. Este trabajo persigue la exploración en la intersección de la IA y la programación, abriendo el camino para la automatización efectiva del desarrollo de código a partir de sentencias en lenguaje español. Se espera que este trabajo proporcione una contribución al creciente cuerpo de literatura que se centra en la generación de código y la comprensión del lenguaje natural en los modelos de lenguaje de IA.

**Palabras Claves:** Generación de Código, Transformers, Modelos Preentrenados, ChatGPT

## ABSTRACT

This study explores the capabilities of the ChatGPT artificial intelligence language model, with the GPT-3 architecture developed by OpenAI, in the task of generating JavaScript source code from Spanish instructions. Transformer language models, as exponents of deep learning, are effective in learning contextual representations of words and phrases. This allows the model to understand not only individual programming terms, but also how they are combined into larger structures such as loops and functions. Through a set of unique programming feature requests from a selected set of cases prepared for this work, we examine the model's ability to transcribe these high-level specifications into executable and functional source code. The results of the model were evaluated in a compiler, seeking an objective evaluation of the functionality of the code generated in unit test cases prepared in advance. The model achieves 100% compliable code and 90% successful problem resolution. This work pursues exploration at the intersection of AI and programming, opening the way for effective automation of code development based on sentences in the Spanish language. This work is expected to provide a contribution to the growing body of literature focusing on code generation and natural language understanding in AI language models.

**Keywords:** Code Generation, Transformers, Pretrained Models, ChatGPT

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

## 1. INTRODUCCIÓN

Los avances en los últimos años de la IA ayudan a los desarrolladores a lo largo del ciclo de vida de desarrollo de software. No es que los programadores estén siendo reemplazados por robots; más bien, las herramientas impulsadas por la inteligencia artificial están haciendo sean más productivos y efectivos (Schatsky & Bumb, 2020).

Nuestra investigación se concentra en poder determinar las habilidades de modelos de generación de texto a partir de modelos preentrenados de lenguaje para la generación de código fuente correcto, pero además en la eficacia de la traducción de instrucciones en español a instrucciones de programación, proporcionando una perspectiva única en las habilidades de comprensión del lenguaje natural de los modelos basados en Transformers (Zhang, y otros, 2023).

Este trabajo promete desbloquear nuevas vías de exploración en la intersección de la inteligencia artificial y la programación, abriendo el camino para la automatización más efectiva del desarrollo de software y la accesibilidad en lenguas no angloparlantes. Por ende, se espera que este estudio proporcione una valiosa contribución al creciente cuerpo de literatura que se centra en la generación automática de código y la comprensión del lenguaje natural en los modelos de lenguaje de IA.

En nuestra investigación inicial de herramientas para la generación automática de código fuente a través de modelos preentrenados de Lenguaje (Bender, Nicolet, Folino, Lopez, & Hansen, 2022), se reveló que estas herramientas podían generar programas simples a partir de cadenas de texto referidas como solicitudes o *prompts* en inglés o mediante encabezados de estilo de documentación. Esta capacidad de diferentes herramientas y sus resultados fue muy interesante porque, por ejemplo, GPT-3 (OpenAI, 2023) no fue entrenado explícitamente para la generación de código fuente, aunque el producto GitHub Copilot (Github, 2023) si lo es.

Dado el considerable éxito de los modelos de generación de lenguajes en otras modalidades y la abundancia de código disponible públicamente, planteamos la hipótesis de analizar las capacidades de una herramienta particular y en idioma español buscando suplir las dificultades de encontrar mecanismos de evaluación (Chen, Tworek, Jun, & y otros, 2021) y su posible implementación en un futuro por las empresas.

Lo estudiado en (Bender, Nicolet, Folino, Lopez, & Hansen, 2022) nos indicó que los modelos que mejor efectividad logran no estaban disponibles para su uso gratuito, lo que limita los avances a lo que puedan lograr empresas pequeñas, organizaciones educativas o incluso programadores individuales, quedando su uso restringido a muy escasas compañías de tecnología.

GPT 3 como modelo, planteó un aspecto innovador es que se define como un modelo de uso general (Task-Agnostic) (Finn, Abbeel, & Levine, 2017), y que por ende puede ser utilizado con un mínimo ajuste fino para diferentes fines, como por ejemplo para la modalidad de preguntas y respuestas, para la redacción de textos extensos, o para la traducción de lenguaje natural (Reed, y otros, 2018).

El lanzamiento de ChatGPT (OpenAI, 2023), que es de uso gratuito o bien de una suscripción muy baja para la versión Plus, basado en GPT-3.5 en octubre del año 2022 generó una expectativa en el público general inusitada. Números no oficiales estimaban para febrero de 2023 un estimado de 100 millones de usuarios de esta herramienta (Hu, 2023). Esa permeabilidad en la opinión pública y la gratuidad del acceso motivaron principalmente la selección de esta herramienta como la seleccionada para este trabajo. El acceso a ChatGPT es por medio de un navegador, por medio de una url publica sin requerimientos particulares para su uso (OpenAI, 2023).

Esa expectativa del uso masivo de ChatGPT y nuestro trabajo previo de investigación (Bender, Nicolet, Folino, Lopez, & Hansen, 2022) acerca de las capacidades de los Modelos de Lenguajes Grande, en ingles *Large Language Models* (LLM) (Shanahan, 2023) es la que llevó a la planificación y concreción de este trabajo donde enfocamos la prueba de la herramienta directamente desde la perspectiva de usuarios finales.

Como menciona (Meyer, Urbanowicz, & Martin, 2023) la utilidad y aplicabilidad de ChatGPT al escribir código dependerá en última instancia de (1) el nivel de detalle proporcionado por el usuario al describir la función y los parámetros del código deseado y (2) de la escala y complejidad del código solicitado, en este trabajo, nos centramos en la tarea de generación de funciones independientes a partir de cadenas de solicitud de texto plano y en algunos casos con ejemplos y la evaluación del código fuente automáticamente generado a través de pruebas unitarias.

Las pruebas documentadas hasta 2022 usaban mayormente Python, sin embargo, en las encuestas de tecnologías más populares de Stackoverflow (StackOverflow, 2022), en 2022 marca el décimo año

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

consecutivo de JavaScript como el lenguaje de programación más utilizado con un 65%, por lo que decidimos realizar este estudio con JavaScript como lenguaje objetivo.

La pregunta de investigación es, entonces: ¿puede ChatGPT producir código fuente en JavaScript de manera correcta y satisfactoria a partir de descripciones en idioma español?

El estudio presenta resultados del tipo cuantitativo y cualitativo, conforme los otros estudios relevados en la bibliografía (Bender, Nicolet, Folino, Lopez, & Hansen, 2022). Este trabajo está organizado de la siguiente manera:

Primeramente, presentamos los problemas seleccionados para el estudio y las clasificaciones que realizamos de ellos (Sección 2).

Luego presentamos metodología, las pruebas realizadas y las herramientas bajo las cuales el estudio se realizó y los considerandos de situaciones particulares (Sección 3).

A continuación, presentamos los resultados cuantitativos, es decir los resultados desde un punto de vista estadístico (Sección 4).

Posteriormente presentamos análisis cualitativos de los resultados entregados por ChatGPT y de las interacciones realizadas (Sección 5) para finalmente,

Finalmente presentamos nuestras conclusiones (Sección 6) y futuras líneas de investigación que se desprenden de estos resultados (Sección 7).

## 2. DEFINICION DE LOS PROBLEMAS

Para comparar con precisión nuestro modelo, creamos un conjunto de datos de 81 problemas de programación originales con pruebas unitarias. Estos datos fueron seleccionados de un conjunto de problemas usados en los últimos años por una cátedra de programación de la USAL, con lo que tienen el nivel o complejidad de lo que debe resolver un estudiante avanzado de ingeniería en informática en un examen.

La mayoría de los trabajos revisados en (Bender, Nicolet, Folino, Lopez, & Hansen, 2022) evalúan la comprensión del lenguaje, algoritmos y matemáticas simples, con algunos comparable a simples preguntas de entrevista de software, con lo que decidimos avanzar más en profundidad y evitamos los problemas simples de matemáticas que ya se habían mostrado resueltos con las herramientas anteriores.

Nos enfocamos mayormente, entonces, en la búsqueda de funciones que trabajen con conjuntos de datos, expresados en forma de arreglos o en matrices, que requieren una mayor abstracción en su comprensión y procesamiento.

### 2.1. Tipos de problemas

Los problemas seleccionados para este trabajo tienen relaciones de conjuntos y arreglos, como unión, intersección, producto cartesiano de arreglos o matrices, productos escalar o vectorial, transposición de matrices e incluso multiplicación de matrices.

Como ejemplos de los problemas que requieren tratamiento de números podemos mencionar análisis numérico como cálculo de divisores, identificación de números amigos, de número perfecto o conversores de base hexadecimal a decimal, binario a decimal y viceversa.

Hemos seleccionado problemas que requieren análisis de conjuntos de números como medianas o promedios sesgados, un generador de serie monótamente creciente o cálculo de máximo común divisor de un conjunto de números de tamaño no determinado.

Como selección de problemas más complejos seleccionamos implementaciones de cálculo de resultantes según el teorema de Varignon, de cálculo de dígitos verificadores de patentes, de implementación de algoritmo de cifrado de Vigenère, de distribución de cargos en elecciones según la metodología D'Hont o una variante del problema de los puentes de Königsberg que si tiene resolución.

Para probar las capacidades de interpretación de los comentarios, sólo al 75% de los problemas se le asignó un ejemplo de resolución, con lo que el 25% fue resuelto sin ejemplos de guía en la entrada o en la salida. Todos los enunciados de los problemas y su clasificación están disponibles para ser consultados.

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

### 2.2. Clasificación de los problemas.

Para poder realizar un mejor análisis de los resultados, realizamos clasificaciones de los problemas en grupos o categorías, en la búsqueda de patrones que nos permitan entender de mejor manera los resultados que obtendríamos. Desde el punto de vista de los datos con los que las funciones deben trabajar, catalogamos los problemas en arreglos, matrices, texto, números y estructuras.

Como los trabajos con conjuntos de datos son más complejos y se requiere una abstracción superior que los trabajos con datos simples, la predominancia de estos es la mayor, con 63% en arreglos o *arrays* y 14% de datos organizados en matrices, para un total de casi 80% de estos datos organizados en conjuntos. Hay procesamiento de números y de texto, aproximadamente 10% de cada uno, dejando una pequeña participación a estructuras que emulan clases (desde el punto de vista de los modelos de programación).

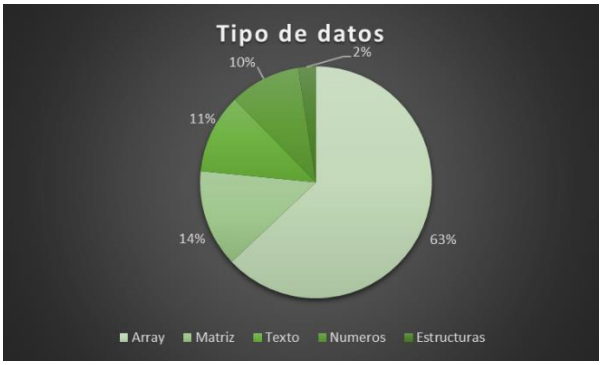


*Figura 1: Participación de problemas en función del tipo de datos a procesar*

Evaluar la complejidad de los requerimientos o de las funciones obtenidas es dificil y arbitrario ya que no se encuentran metodologías adecuadas para realizarlas (Bender, Nicolet, Folino, Lopez, & Hansen, 2022). Sin embargo y basados en nuestra experiencia docente, catalogamos los problemas en tres categorías: (problemas) sencillos, (de) media (complejidad) y de alta (complejidad).

Esta catalogación tiene aspectos referidos a la complejidad con la que se debe entender la lógica interna o el procesamiento de datos. Como en todo proceso evaluativo, la participación de problemas sencillos es mayor en esta base de problemas, con una participación en el diseño de 75 / 25 / 5, llegando a porcentajes reales de 72% de problemas sencillos, 22% de problemas de mediana complejidad y 6% de problemas de alta complejidad.

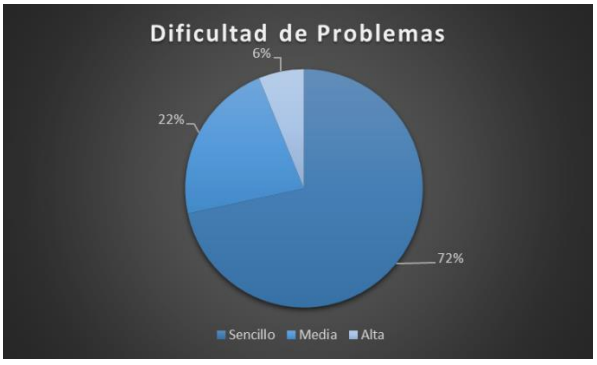


*Figura 2: Clasificación de los problemas en base a su complejidad*

Con respecto a otros análisis respecto de las características de las funciones requeridas, también analizamos las mismas desde la cantidad de parámetros de entrada y la cantidad de parámetros de salida. Estas caracterizaciones respecto a cantidad de parámetros no son tan importantes en la evaluación de la dificultad como la de la complejidad del procesamiento, pero podemos concluir que, entre más parámetros de entrada, mayor dificultad de resolución. De la misma manera, entre más parámetros tenga la salida de la función concluimos que la función es de mayor dificultad de resolución.

Respecto de los parámetros de entrada, la mayoría de las funciones requeridas tienen uno (53%) o dos parámetros (33%), con un total para las dos condiciones, las más frecuentes del 85% con participaciones mejores de tres (7%), cuatro (4%) o cero (3%) parámetros de entrada.

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

Respecto a los parámetros de salida, la participación de un parámetro de salida es mayoría, con un 75%, y participaciones similares de dos (8%), tres (7%), parámetros variables de salida (9%) y cuatro (1%).

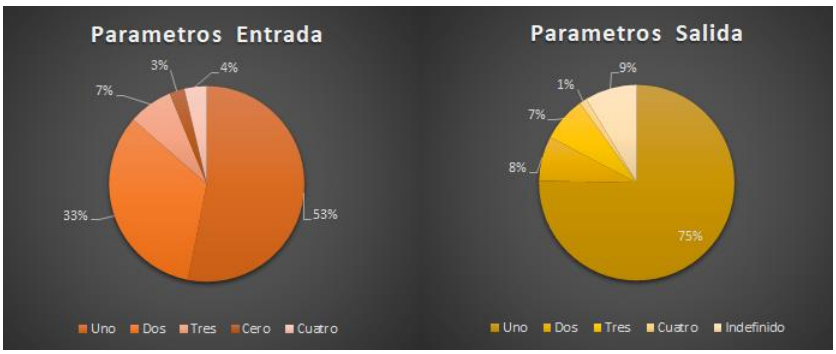


*Figura 3: Clasificación de los problemas en base a la cantidad de parámetros de entrada y de salida*

## 3. METODOLOGIA Y MARCO DE EVALUACIÓN

Para las pruebas del código usamos NodeJS, que es un entorno de ejecución para JavaScript construido con V8, el motor de JavaScript de Chrome (NodeJS.org, 2023). Seleccionamos Visual Studio Code (Microsoft, 2023) como IDE (Price, 2021) de desarrollo, herramientas que son gratuitas para uso tanto privado como comercial de tal manera que no sea restrictiva la adopción por tener que pagar licencias.

Todos los problemas fueron escritos con la misma o base, solicitando una función en JavaScript con un nombre determinado y unívoco que cumpliera un determinado requisito. Como mencionamos anteriormente, algunos de los comentarios tenían ejemplos y otros no.

La metodología consistió en tomar cada uno de los enunciados de los problemas escritos y solicitarle a ChatGPT la resolución del problema. A continuación, proporcionamos ejemplos de requerimientos realizados.

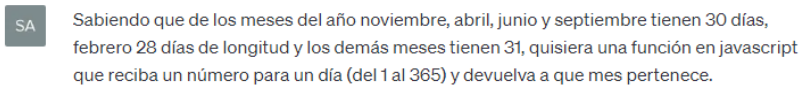


*Figura 4: Ejemplo 15 de interacción con ChatGPT*

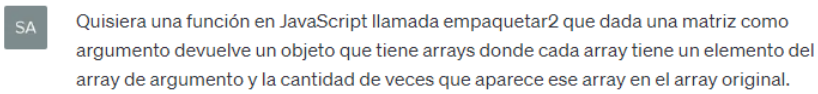


*Figura 5: Ejemplo 49 de interacción con ChatGPT*

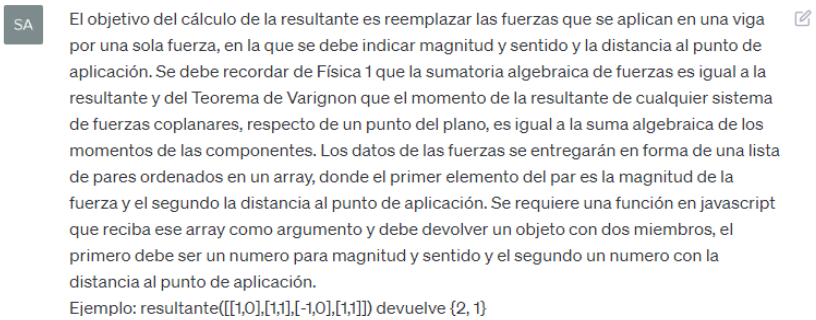


*Figura 6: Ejemplo 77 de interacción con ChatGPT*

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

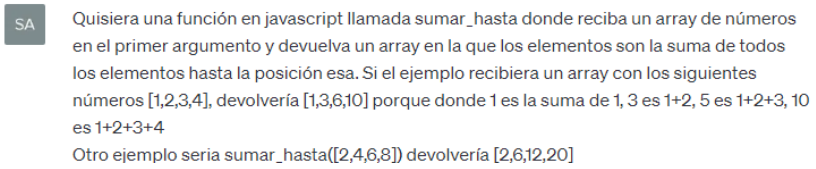


*Figura 7: Ejemplo 64 de interacción con ChatGPT*

El resultado entregado por ChatGPT de la solicitud está compuesto por cuatro partes: introducción, código fuente de la función requerida, ejemplos de pruebas y descripción de la función.

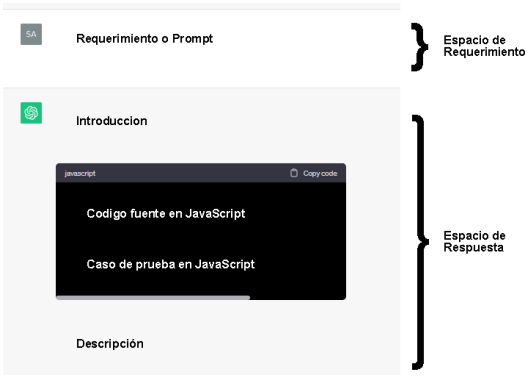


*Figura 8: Organización espacial de la interacción con ChatGPT*

En los casos que el requerimiento tuviera uno o más casos de prueba, detectamos que normalmente eran los usados por ChatGPT como ejemplos o casos de prueba. En los ejemplos que no tenían en el requerimiento, el modelo los generó por sí mismo y sin solicitarlo de antemano.

Hicimos una copia de la interacción con el modelo y la almacenamos para tener el respaldo y poder analizarla con detenimiento. A continuación, presentamos algunos ejemplos de la respuesta del modelo a los problemas planteados.

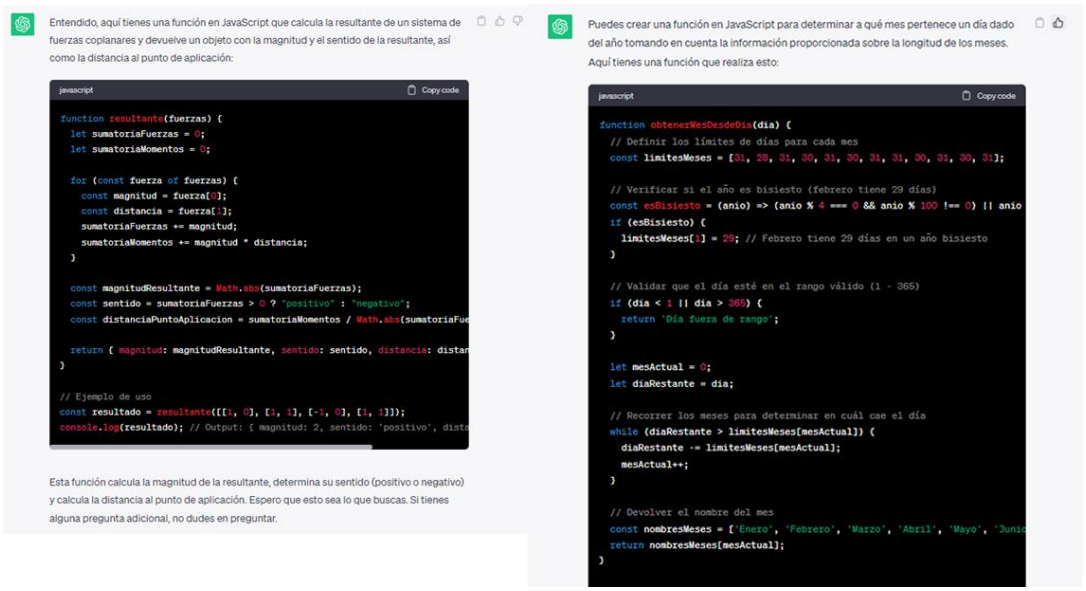


*Figura 9: Respuesta de ChatGPT a los ejemplos 15 y 77*

El código fuente y los ejemplos de pruebas, así como las pruebas unitarias previamente elaboradas fueron integrados en un único archivo para ser probados en la IDE con las pruebas unitarias. Luego compilamos y se ejecutamos la función y analizamos los resultados, si el resultado era correcto para las pruebas unitarias, se asentó el resultado del problema como un 1.

En el caso que la prueba unitaria no fuera satisfactoria, en línea con la herramienta se solicitaban modificaciones, de la misma manera que haría un programador individual, para verificar el uso desde ese punto de vista. Esto contrasta con la generación de lenguaje natural, donde las muestras se evalúan

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

típicamente por heurística o los trabajos estudiados en (Bender, Nicolet, Folino, Lopez, & Hansen, 2022) que buscan realizar las pruebas automáticamente o mediante automatización. Si con una sola interacción de solicitud de modificación se obtuvo un resultado correcto, se asentaba el resultado como un 2.

En los casos que con una interacción no se obtuvo un resultado correcto en las pruebas unitarias, se verificaba el código tratando de entender si el problema era de la solicitud o de la implementación. Se intentó lograr que el modelo corrija el código adecuadamente y si se logra con algunas interacciones, se lo cataloga como un 3. En el caso que se superaron los cuatro intentos se lo cataloga como fallo y se le asigna un 4.

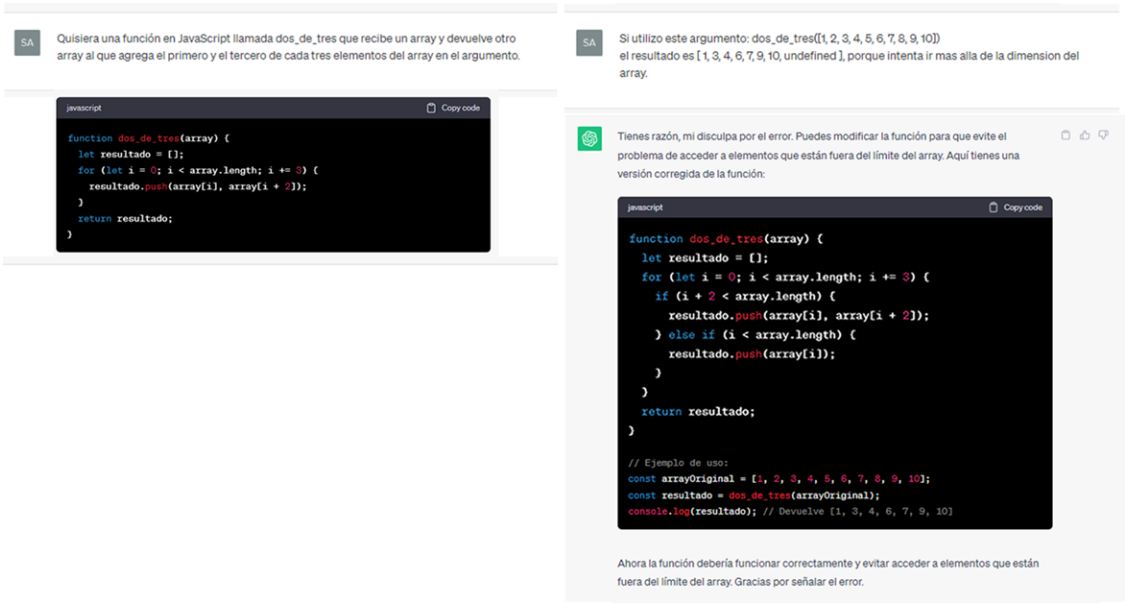


*Figura 10: Ejemplo de solicitud de modificación en ejemplo 12*

En algunos casos la solicitud fue no utilizar algunas funciones muy específicas de JavaScript para evaluar el desarrollo funcional de la función de manera más sencillo, evitando el uso de funciones como *splice* o *slice*.

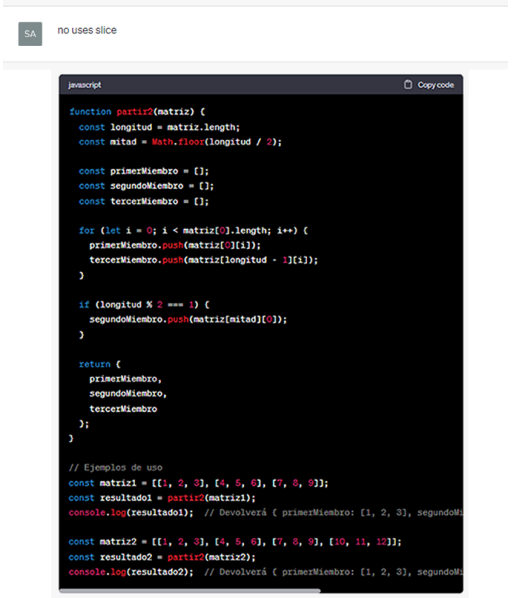


*Figura 11: Ejemplo de solicitud de modificación en ejemplo 37*

Una vez otorgado el puntaje 1-4 respecto de la resolución funcional de los problemas planteados, se los revisaba desde un punto de vista de usuario, respecto a la legibilidad del código, los comentarios, los ejemplos y la explicación del funcionamiento de la función. Anteriormente, investigaciones han

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

usado métricas como BLEU (Papineni, Roukos, Ward, & Zhu, 2002) y las métricas de generación se han concentrado en gran medida en analizar la exactitud y complejidad de la salida del código en lugar de la expresividad y complejidad en sí misma. Esta revisión cualitativa se usa solo como referencia ya que no parece ser comparable.

## 4. RESULTADOS CUANTITATIVOS

En todos los casos el código fuente que el modelo generó compiló adecuadamente sin errores sintácticos y los ejemplos de pruebas dados por el modelo eran adecuados para la verificación sintáctica. Esto es un avance respecto a otros estudios revisados anteriormente (Chen, Tworek, Jun, & y otros, 2021).

Respecto a la resolución funcional, el resultado fue del 76% para las soluciones con un funcionamiento correcto directamente la salida del modelo. Los casos en los que se requirió una modificación con una sola interacción adicional con el modelo fueron del 14%, con lo cual podemos indicar que con la salida directa del modelo o con una sola interacción el resultado fue del 90% que es el que consideramos exitoso en función de los parámetros determinados al inicio del trabajo.

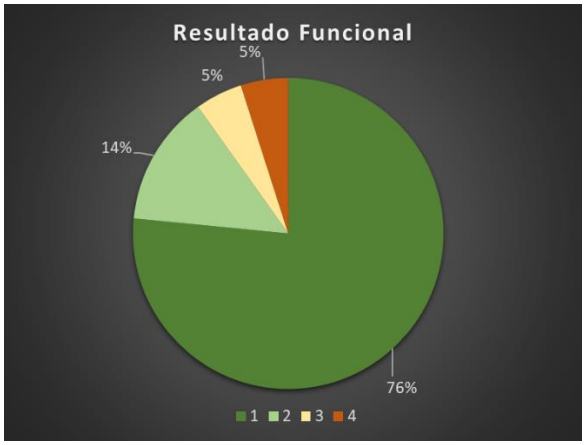


*Figura 12: Resultado de la evaluación funcional de las funciones obtenidas*

Como en nuestra clasificación de los problemas sencillos eran cercanos al 75%, analizamos los resultados funcionales respecto a la dificultad de los casos. Lo que visualizamos de este cruce es que la resolución funcional no tiene relación con la dificultad, sino que la distribución de resultados es similar en todos los casos de dificultad.

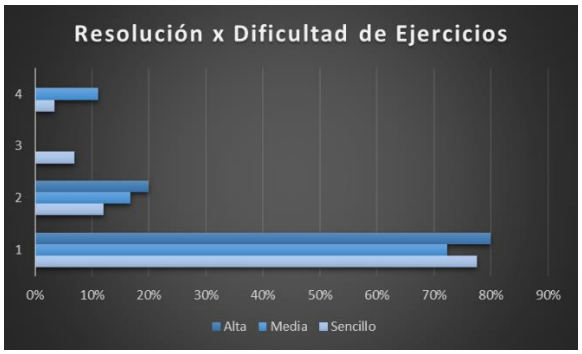


*Figura 13: Resolución respecto de dificultad de problemas*

Para el siguiente análisis, cruzamos los resultados de la resolución funcional respecto a los tipos de datos. Para una simplificación de la visión de este cruce, sumamos los valores del resultado funcional 1 y 2 porque el resultado es considerado correcto y por otro lado sumamos los valores de resultados de evaluación funcional 3 o 4 por considerarlos en general no correctos. Al igual que respecto a la dificultad de los problemas, no hay diferencias evidentes en las resoluciones funcionales respecto a los tipos de datos a procesar por las funciones.

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

La única mención por revisar es la de matrices, que tuvo un desempeño inferior al resto, con un 73% sumados las calificaciones 1 y 2, sin embargo, corresponden a la mayor cantidad de casos considerados de dificultad media.


*Figura 14: Resolución respecto de los tipos de datos*

Como otra dimensión evaluada para los problemas presentados fueron la cantidad de parámetros de entrada y la cantidad de parámetros de salida, debemos analizar los resultados funcionales cruzados con estas dimensiones. Sumando nuevamente los resultados para clasificaciones 1 y 2, se tuvieron valores de 100% o 90% para cada una de las categorías por parámetros de entrada, correspondientes con el resultado general.

En el caso de los parámetros de salida, si vemos diferencias respecto al resultado general, obteniéndose valores inferiores a los resultados generales para tres parámetros de salida (2 casos) y parámetros variables (2 casos). La baja cantidad de muestras hacen que estos números proporcionales parezcan muy altos sin embargos son pocos casos, que se analizaron individualmente. En estos casos lo que se encontró, en el análisis cualitativo, es que la lógica subyacente era más compleja de expresar en lenguaje natural y las solicitudes no fueron exitosas por ese motivo. Se debe buscar una manera diferente de expresar esos determinados condicionantes de lo solicitado.

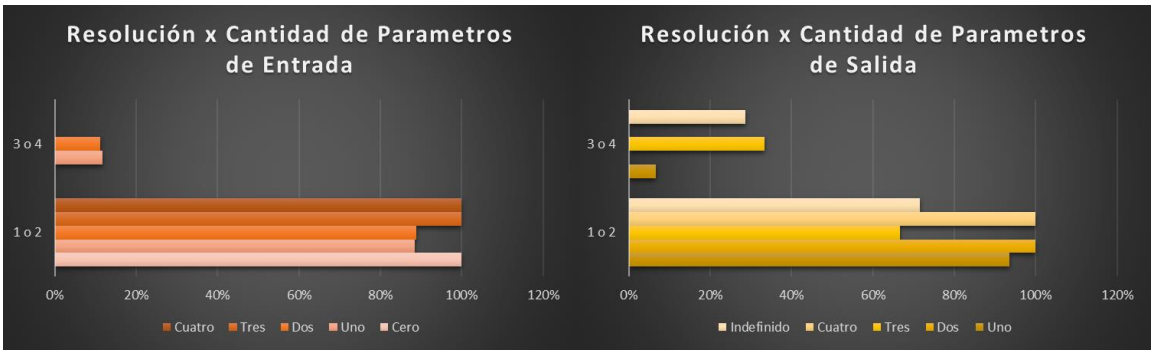


*Figura 15: Resolución en base a cantidad de parámetros de entrada y de salida*

## 5. RESULTADOS CUALITATIVOS

El análisis cualitativo del código fuente generado por el modelo abordo diferentes dimensiones tratando de evaluar la aplicabilidad y calidad del código desde el punto de vista de un usuario programador. Las dimensiones relevadas para cada una de las respuestas a los problemas planteados.

### 5.1. Claridad y Legibilidad

Todo el código es muy legible, con un uso correcto de la indentación y espaciado entre bloques, lo que resulta muy claro a la hora de leerlo. En todos los casos donde se dejó al modelo seleccionar los nombres de las variables o de las funciones, estas fueron descriptivas adecuadamente. Como ejemplos podemos mostrar:

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

*Tabla 1: Ejemplos de nombres de variables*

| Número del ejemplo | Ejemplos de usos / nombres de Variable |
|---|---|
| **54** | **const** resultado **= [...arr];**<br>**const** totalElementos **= arr.length;** |
| **56** | **const** vueltoRestante **= pago - gasto;**<br>**const** vueltoFinal **= [];** |
| **80** | **let** maxRatio **= 0;**<br>**let** partidoGanador **= -1;**<br>**for (let** j **= 0;** j **<** numPartidos**;** j**++) {**<br>**const votos =** resultados**.filter(result => result[0] === j + 1).map(result =>**<br>**result[2]).reduce((**total**,** v**) => total +** v**, 0);**<br>**const** ratio **=** votos **/ (**asignacion**[**j**] + 1);** |
| **81** | barrios**: ['Kneiphof', 'Vorstadt', 'Altstadt', 'Lomse'],**<br>**const** caminoEuleriano **= [];** |

### 5.2. Estructura y Organización

El código fue evaluado desde el punto de vista de la organización, de la declaración de variables de la inicialización de las variables cuando corresponda y el resultado fue satisfactorio. Se respetan las mismas convenciones en la indentación como en el espaciado a lo largo de todas las interacciones.

En algunos casos el modelo generó funciones y estructuras para mejorar la modularidad del código y en todos los casos agregó al final del código fuente ejemplos de uso adecuados para probar la función.

### 5.3. Explicaciones y comentarios

El código fuente generado por el modelo en general no tiene comentarios incrustados sin embargo algunas definiciones si los incorporan. Todas las funciones, sin embargo, tuvieron asociadas un texto explicativo relacionada al diseño de la función o a su utilización. Estos textos usan palabras en negrita o entre comillas simples para realizar énfasis en algunas palabras para mejorar la legibilidad.

No fue parte del trabajo solicitar documentación línea a línea, pero entendemos que dadas las características de explicación de las funciones realizadas estas podrían solicitarse y el modelo cumplimentarlo, pero quedará para una posible futura línea de investigación.

Estas dimensiones de análisis podrían adaptarse o ampliarse dependiendo de los objetivos específicos del estudio. Este análisis podría mejorarse con entrevistas con expertos o análisis de contenido para recopilar más datos en estas dimensiones, pero lo dejaremos para otras futuras líneas de investigación.

Si es importante remarcar de este análisis que el modelo generó, en algunas resoluciones, resultados más detallados o explicativos que no fueron solicitados explícitamente.

*Tabla 2: Ejemplos de respuestas más explicativas*

| Numero de Ejemplo | Solicitud | Resultado |
|---|---|---|
| **19** | Quisiera una función en javascript llamada posyneg2 que, dada una lista de números, separe los elementos positivos en un array, los negativos en otro y en un tercer array los ceros y devuelva los tres arrays como miembros de un objeto | {<br>positivos: [ 1, 3, 4, 6, 7 ],<br>negativos: [ -2, -5, -8, -9 ],<br>ceros: [ 0, 0 ]<br>} |
| **37** | Quisiera una función en JavaScript llamada partir2 que reciba un parámetro que es una matriz. La función debe retornar un objeto que tenga tres miembros. El primer y el tercer elemento son arrays que tienen la misma cantidad de elementos, los primeros y los últimos de la matriz. El miembro del medio es un array que tiene un elemento. | {<br>primerMiembro: [ [ 1, 2, 3 ] ],<br>segundoMiembro: [ [ 4, 5, 6 ] ],<br>tercerMiembro: [ [ 7, 8, 9 ] ]<br>} |

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

| | | |
|---|---|---|
| **81** | Quisiera una función en javascript llamada puentes de tal manera que, usando la información de los puentes definida, desde un barrio cualquiera se realice una ruta de puentes cruzando todos ellos, pero una sola vez cada uno y que devuelva un objeto que tenga como primer miembro el barrio desde donde salió, como segundo miembro el barrio a donde llegó y el orden de los puentes que cruzó. | [ { puente: 'a', desde: 'Vorstadt', hacia: 'Kneiphof' },<br>{ puente: 'b', desde: 'Kneiphof', hacia: 'Vorstadt' },<br>…<br>{ puente: 'g', desde: 'Lomse', hacia: 'Altstadt' } ] |

## 6. CONCLUSIONES

Hemos visto que el código fuente obtenido de todas las interacciones compila en todos los casos, 100% de éxito. Adicionalmente hemos podido corroborar que las resoluciones se encuentran funcionalmente muy cercanas a lo solicitado, en el 90%, con las denominaciones, nombres de los identificadores (variables y funciones) relacionadas con el problema a resolver, lo que mejora la legibilidad del código fuente.

El desafío más importante encontrado en la generación de código automática surge de confiar en el supuesto de que la intención del programador se pueda expresar con suficiente precisión mediante comentarios e interacciones, que la herramienta va a implementar con precisión lo solicitado. Esto a su vez nos formula algunas preguntas acerca de cuáles son las mejores sentencias, comentarios e incluso sintaxis o indicaciones que tengan la suficiente precisión para extraer el mejor comportamiento del modelo.

El código generado tiene que ser probado en pruebas unitarias, que pueden ser automatizadas, pero de no superarlas el programador debe revisar ese código y determinar si realiza una corrección al código o bien una nueva interacción con el modelo. Podemos concluir que, incluso con estas herramientas, incluso si el modelo fuera perfectamente preciso, no esperaríamos que se eviten la presencia de programadores para generar código, o que se reduzcan los costos laborales asociados con la escritura de código fuente, pero si podemos vislumbrar una mayor capacidad de generación de código fuente por unidad de tiempo.

Las tareas de programación del mundo real a menudo implican iteraciones de enfoques y correcciones de errores, que se aproximan generando muestras alternativas, con lo que la falla de una prueba unitaria desde el punto de vista funcional no fue considerada en si misma un error y durante el proceso volvimos sobre la herramienta para pedirle correcciones o explicar mejor el objetivo buscado.

Estas capacidades hacen de ChatGPT una herramienta potencialmente útil para la generación de código y su explicación simultánea. Sin embargo, vale la pena señalar que, aunque el modelo puede generar explicaciones útiles basadas en los patrones que ha visto en sus datos de entrenamiento, no tiene una comprensión conceptual real del código, de la misma manera que un humano lo haría. Todas sus respuestas son el producto de patrones estadísticos que ha aprendido durante su entrenamiento.

Podemos, entonces, suponer que, dentro de la academia, se debería enseñar a los estudiantes a interactuar con ChatGPT de manera constructiva de acuerdo con la ética de la institución educativa y dentro de la rama profesional sería deseable que, mientras no se violen los derechos de propiedad intelectual (desconocidos por el usuario), pueda el programador interactuar con la herramienta que produzca código fuente a partir de comentarios de la mejor manera y más eficiente, pues esto podría radicar en una mayor productividad.

## 7. FUTURAS LINEAS DE INVESTIGACION

Dado que hemos comprobado que este modelo genera código fuente sintácticamente correcto en todos los casos y funcionalmente correcto en la gran mayoría, creemos que se debe seguir investigando su impacto en lo laboral, por lo que algunas áreas en las que creemos que se debe seguir investigando deben incluir:

Estudiar la capacidad del modelo de poder llevar adelante proyectos enteros mediante una sola solicitud o mediante varias interacciones ya que este estudio fue de generación de bloques unitarios, funciones.

Estudiar el impacto de solicitarle al modelo diferentes bloques de código con solicitudes estructuradas de determinada la mejor manera de realizar las solicitudes y determinar si estandarizar las formas verbales respecto de comunicaciones más informales, en lenguaje natural generan mejores resultados.

Universidad Tecnológica Nacional
XVI COINI 2023 1, 2 Y 3 de noviembre de 2023
Facultad Regional San Nicolás

AJEA –Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes del congreso DOI: https://doi.org/10.33414/ajea.1316.2023
Publicación en AJEA https://rtyc.utn.edu.ar/index.php/ajea/article/view/1452

Medir el valor económico de utilizar una herramienta para generar bloques enteros de código más rápido que deben ser integrados en proyectos en curso. Esto puede incluir realizar mediciones sobre tiempo de escritura de cantidades de texto por diferentes programadores y contrastarlos con los tiempos de usar este u otras herramientas que expongan modelos y cambiar a las herramientas de desarrollo, ya que el cambio de contexto de trabajo también puede generar ineficiencias.

Medir que el impacto en organizacionales dedicadas a la construcción de software para determinar cambios en las prácticas de documentación de código a partir de que la generación gruesa de código fuente la realice una herramienta y no el programador e incluso que sea el modelo que documente los bloques de código y línea a línea.

## 8. REFERENCIAS.

Bender, A., Nicolet, S., Folino, P., Lopez, J., & Hansen, G. (2022). Generación Automática de Código Fuente a través de Modelos Preentrenados de Lenguaje, un análisis de la literatura. *AGRANDA 2022 - Simposio Argentino de Ciencia de Datos y GRANdes DAtos*, 50-65.

Chen, M., Tworek, J., Jun, H., & y otros. (2021). Evaluating Large Language Models Trained on Code. *arXiv:2107.03374v2*.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400v3*.

Github. (2023). *Co-pilot*. Obtenido de Co-pilot: https://github.com/features/copilot

Hu, K. (2 de Febrero de 2023). *www.reuters.com*. Obtenido de www.reuters.com: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

Meyer, J., Urbanowicz, R., & Martin, P. (2023). ChatGPT and large language models in academia: opportunities and challenges. *BioData Mining 16*, 20.

Microsoft. (05 de 2023). *Visual Studio Code.* Obtenido de Visual Studio Code: https://code.visualstudio.com/

NodeJS.org. (05 de 2023). *NodeJS.* Obtenido de NodeJS: https://nodejs.org/es

OpenAI. (2023). *Chatgpt*. Obtenido de Chatgpt: https://openai.com/blog/chatgpt

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–318.

Price, M. J. (2021). *C# 10 and .NET 6 – Modern Cross-Platform Development.* Packt Publishing.

Reed, S., Chen, Y., Paine, T., van den Oord, A., Eslami, A., Rezende, D., . . . de Freitas, N. (2018). Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions. *International Conference on Learning Representations*.

Schatsky, D., & Bumb, S. (22 de Enero de 2020). *AI is helping to make better software.* Obtenido de AI is helping to make better software: https://www2.deloitte.com/us/en/insights/focus/signals-for-strategists/ai-assisted-software-development.html..html

Shanahan, M. (2023). Talking About Large Language Models. *arXiv:2212.03551v5*.

StackOverflow. (Diciembre de 2022). *Most popular technologies.* Obtenido de Most popular technologies: https://survey.stackoverflow.co/2022/#technology-most-popular-technologies

Zhang, S., Chen, Z., Shen, Y., Ding, M., Tenenbaum, J., & Gan, C. (2023). Planning with Large Language Models for Code Generation. *arXiv:2303.05510*.