

Pensamiento Computacional y ChatGPT



Revista EIA
ISSN 1794-1237
e-ISSN 2463-0950
Año XIX/ Volumen 21/ Edición N.42
Julio - diciembre de 2024
Reia4214 pp. 1-24

Publicación científica semestral
Universidad EIA, Envigado, Colombia

PARA CITAR ESTE ARTÍCULO / TO REFERENCE THIS ARTICLE /

Hoyos-Rivera, G. J.; Hernández-
Hernández, J. C.
Pensamiento Computacional
y ChatGPT
Revista EIA, 21(42), Reia4214.
pp. 1-24.
<https://doi.org/10.24050/reia.v21i42.1739>

✉ Autor de correspondencia:

Guillermo-de-Jesús Hoyos-Rivera
Licenciado en Informática y Doctor en
Informática
Universidad Veracruzana, México
ghoyos@uv.mx

Recibido: 23-10-2023
Aceptado: 27-05-2024
Disponible online: 01-07-2024

✉ GUILLERMO-DE-JESÚS HOYOS-RIVERA¹
JOSÉ-CLEMENTE HERNÁNDEZ-HERNÁNDEZ¹

1. Universidad Veracruzana, México

Resumen

Con el advenimiento de los modelos de procesamiento del lenguaje natural, forma en que solemos referirnos al lenguaje humano, de tipo Large Language Models (LLM), como Bidirectional Encoder Representations from Transformers (BERT), Language Model for Dialogue Applications (LaMDA), Large Language Model Meta Artificial Intelligence (LLaMA), o el tan nombrado Generative Pre-trained Transformers (GPT), se comienzan a generar entre sus usuarios, y el público en general, cualquier cantidad de especulaciones y expectativas acerca de sus alcances, y se comienzan también a explorar nuevas formas de uso, algunas realistas, otras creativas, y otras rayando en la fantasía. El Pensamiento Computacional no es la excepción en esta tendencia. Ésta es la razón por la cual es de suma importancia tratar de elaborar una visión clara acerca de esta innovadora tecnología, buscando evitar la creación y propagación de mitos, que lo único que hacen es enrarecer la percepción y comprensión que se tiene, y tratar de hallar el justo medio en cuanto a los alcances que pueden tener este tipo de tendencias tecnológicas, y las formas de uso que se les puede dar, con especial énfasis en el Pensamiento Computacional. En el presente artículo presentamos un breve análisis sobre lo que es Generative Pre-trained Transformer, así como algunas reflexiones e ideas sobre las formas en que los Large Language Models pueden influir al Pensamiento Computacional, y sus posibles consecuencias. En especial, en este documento, se analiza el conocido ChatGPT, del cual se presenta una valoración sobre las salidas de texto generadas, y su credibilidad para su uso en tareas de uso común para el Pensamiento Computacional, como lo es el realizar un algoritmo, su código y resolver problemas lógicos.

Palabras clave: Pensamiento Computacional, ChatGPT, Modelos de Lenguaje

Computational Thinking and ChatGPT

Abstract

With the advent of natural language processing models, the way we usually refer to human language, of Large Language Models (LLM) type, such as Bidirectional Encoder Representations from Transformers (BERT), Language Model for Dialogue Applications (LaMDA), Large Language Model Meta Artificial Intelligence (LLaMA), or the so called Generative Pre-trained Transformers (GPT), users and the general public are beginning to generate any number of speculations and expectations about how far they can go, and are also starting to explore new ways of use, some realistic, others creative, and others verging on fantasy. Computational Thinking is no exception to this trend. This is why it is of utmost importance to try to elaborate a clear vision about this innovative technology, seeking to avoid the creation and propagation of myths, which only make the perception and understanding that we have, and to try to find the right balance in terms of the reaches that can have this type of technological trends, and the ways of use that can be given to them, with special emphasis on Computational Thinking. In this paper we present a brief analysis of what Generative Pre-trained Transformer is, as well as some reflections and ideas about the ways in which Large Language Models can influence Computational Thinking, and their possible consequences. In particular, in this paper, we analyze the well-known ChatGPT, presenting an evaluation of the generated text outputs, and its credibility for its use in tasks of common use for Computational Thinking, such as performing an algorithm, its code and solving logical problems.

Key Words: Computational Thinking, ChatGPT, Language Models

1. Introducción

En diciembre de 2022 salió a la luz un nuevo modelo de tratamiento del lenguaje humano de tipo LLM basado en Redes Neuronales Artificiales (RNA), el cual implementa, como parte de su solución, los ya ampliamente conocidos *Transformers* (Vaswani, et al. 2017), propuesta hecha por científicos de Google para enriquecer la capacidad de análisis y procesamiento de los LLM. Éste fue nombrado ChatGPT, y se trata de un programa de tipo “chatbot”: una implementación de naturaleza conversacional, basada en GPT-3 (Radford, et al. 2019), capaz de llevar a cabo intercambios bajo la forma de discusiones en lenguaje natural en varios idiomas, incluido inglés, español, griego, italiano, entre otros, de manera

suficientemente convincente desde el punto de vista humano. Complementariamente, dado que el conjunto de datos a través del cual fue entrenado incluye código fuente en varios lenguajes de programación, es capaz de proponer soluciones algorítmicas a problemas que se le plantean a través de su interfaz.

Rápidamente, y cada vez con más intensidad, su uso se ha expandido, y se ha comenzado a especular acerca de si dicho modelo ya contenía lo que se conoce como Inteligencia Artificial General (IAG). Esto, entre otros hechos, tuvo por efecto una serie de comentarios expresando opiniones por parte de connotados científicos del área, entre ellos, Yann LeCun y Gary Marcus, entre otros, quienes han tenido confrontaciones y calurosos debates relacionados con el Aprendizaje Profundo (AP), y sus implicaciones con relación al desarrollo de este tipo de tecnologías. También han aparecido en diversos medios de comunicación afirmaciones atrevidísimas, como aquella en la que Blake LeMoine, ingeniero de Inteligencia Artificial en Google Inc., osó aseverar que la Inteligencia Artificial de Google había adquirido “consciencia”, llegando a especular sobre el “miedo” que tenía el modelo de morir (ser desconectada), entre otras cosas (Wertheimer, 2022).

Más allá de los hechos aquí consignados, lo que se puede constatar es un creciente interés por parte de cualquier tipo de usuarios por experimentar y hacer uso de estas tecnologías, causando, en muchos de éstos sorpresa, asombro, y hasta sobresalto, al descubrir que las respuestas generadas parecieran haber sido elaboradas por un ser humano. Tal es el nivel logrado, al menos aparentemente, que podrían superar la conocida Prueba de Turing, todo un hito en términos de la forma de determinar la propiedad de “inteligencia” en dispositivos creados por el hombre. Sin embargo, como podrá inmediatamente suponerse, aún nos encontramos a una distancia considerable de tal objetivo. En este tipo de situaciones suele aplicarse la máxima que reza: “la belleza, como la inteligencia, están en el ojo del observador”, y el caso de ChatGPT no es la excepción.

El principal problema con el uso de ChatGPT es que, ante lo sorprendentemente correcto de algunas de sus interacciones, se corre el riesgo de creer ciegamente en lo que la “Inteligencia Artificial”

genera como respuesta, y en consecuencia tender a no ejercer el criterio propio para determinar si éstas son verosímiles, coherentes o plausibles, promoviendo, de alguna manera, una cierta forma de pereza mental al trasladar el esfuerzo de análisis fuera de la mente. Esto podría implicar el riesgo de una reducción en la capacidad de nuestro pensamiento crítico. Así, máximas como “es más peligroso el retroceso de la inteligencia natural, que el progreso de la inteligencia artificial” pueden cobrar una inusitada relevancia. Esto ya parece comenzar a ser una realidad, pues es de todos conocido que estudiantes de diversos niveles están haciendo uso irreflexivo de ChatGPT para efecto de elaborar sus proyectos escolares, situación que ha provocado encendidos debates al respecto, con posturas encontradas en la comunidad educativa.

Esto es particularmente cierto para el caso del PC, pues ante la obtención de soluciones fáciles y rápidas, propuestas como respuesta a planteamientos hechos a ChatGPT, y la continua presión académica o laboral por obtener soluciones en períodos de tiempo muy cortos, se corre el riesgo de ceder al canto de las sirenas, y creer con fe ciega en ChatGPT, y considerar las respuestas obtenidas como soluciones indudablemente correctas (Denning, 2019). Es importante tener mucho cuidado, y cualquier solución algorítmica o computacional obtenida desde ChatGPT debe ser analizada con la mayor minucia, tratando de encontrar posibles fragmentos de solución que no correspondan con la realidad del problema que se pretende resolver, y que a la postre, pudieran tener consecuencias desde simplemente erróneas, hasta graves, desastrosas o incluso funestas. Es justamente aquí que el PC adquiere particular relevancia para hacer frente a este tipo de riesgos.

El PC, como es sabido, se puede definir como el proceso a través del cual un individuo, haciendo uso de diversas habilidades, como pensamiento crítico, razonamiento matemático, pensamiento algorítmico, pensamiento lateral, etc., logra generar soluciones creativas a problemas cotidianos de la manera más general, no sólo aquellos que tienen que ver con el uso de computadoras. Adicionalmente, estas soluciones pueden ser claramente descritas, de manera que sean fácilmente reproducibles posteriormente. Este término tiene su origen en las ideas de Seymour Papert, y fue

ampliamente impulsado más tarde por Jeanette Wing, a partir de 2006 (Lodi y Martini, 2021).

El uso de ChatGPT no necesariamente contribuye a la mejora de las habilidades relacionadas con el PC, las cuales están orientadas a la creación de soluciones creativas de problemas. Sin embargo, usado adecuadamente, y con suficiente criterio, ChatGPT puede ser considerado como una herramienta eficiente como un auxiliar en el proceso de aprendizaje de los conceptos y prácticas relacionadas con el PC, permitiendo la generación de ciclos virtuosos, a través de los cuales el aprendiz de este tema, puede auxiliarse de este tipo de herramientas, para incrementar el ritmo y calidad de adquisición de nuevas habilidades.

El presente artículo pretende ser un ensayo para hablar sobre este tema de gran actualidad, así como para analizar las implicaciones que éste puede tener sobre el PC, en particular en lo concerniente al aprendizaje sus propios fundamentos.

El resto de este documento está organizado como sigue: en la Sección 2 se describen brevemente las bases utilizadas y los procesos realizados por ChatGPT; la Sección 3 ilustra y expone algunos ejemplos del uso de ChatGPT para favorecer el PC, así como las consecuencias de hacerlo con este modelo; en la Sección 4, presentan algunas buenas prácticas del uso de ChatGPT en conjunto con el PC, y por último, en la Sección 5, se muestran algunas conclusiones sobre este ensayo, así como algunas pistas sobre cómo mejorar las habilidades relacionadas con el PC a través del uso de ChatGPT.

2. Cómo funciona ChatGPT

ChatGPT, como ya se mencionó, es una implementación de naturaleza conversacional de GPT-3. Este tipo de desarrollos se ubican en el área de trabajo propia de la Inteligencia Artificial llamada Procesamiento del Lenguaje Natural (PLN), en la cual se aplican técnicas algorítmicas, estadísticas, y de Inteligencia Artificial, para resolver tareas que tengan que ver con el tratamiento automático del lenguaje humano en varias vertientes, que van desde el análisis,

hasta la traducción, generación e interpretación de texto (Eiseinstein, 2018). El lenguaje natural, al ser una creación inherentemente humana, sólo adquiere sentido al momento de su interpretación por parte de otros humanos, siempre y cuando todos los interlocutores conozcan las reglas en las que se basa la codificación y decodificación de los mensajes transmitidos. Sin el conocimiento de estas reglas, es imposible establecer comunicación alguna. Dicho de manera más simple: todos los que interactúan en un intercambio en lenguaje natural, deben hablar y comprender el mismo idioma.

Un ejemplo particularmente claro de este tipo de situaciones es el caso de la Piedra Rosetta (Dalby, 2019). Ésta es una antigua estela egipcia grabada en piedra, en la cual se encuentra inscrito un decreto publicado en Menfis en el año 196 a.C. en nombre del faraón Ptolomeo V. El decreto aparece en tres escrituras distintas: el texto superior en jeroglíficos egipcios, la parte intermedia en escritura demótica y la inferior en griego antiguo. La escritura egipcia permaneció mucho tiempo sin ser descifrada, y no fue sino gracias al hecho de que presenta esencialmente el mismo contenido en las tres inscripciones, que esta estela facilitó la clave para el desciframiento y comprensión, por parte de Jean-François Champollion (Chapollion, 1828), de los jeroglíficos egipcios.

Algunas de las actividades específicas que se abordan en el PLN son, como de manera somera se mencionó anteriormente: clasificación de texto, análisis de polaridades en frases, generación automática de texto y traducción, por mencionar sólo algunas. En el caso particular de ChatGPT, al ser una extensión de GPT-3 entrenada para operar bajo la forma de agente conversacional, es esencialmente generación automática de texto, bajo la forma de respuestas a planteamientos hechos a través de su interfaz por parte de sus usuarios.

Tanto GPT-3 como ChatGPT son herramientas diseñadas para efecto de, con base en una secuencia de palabras de entrada, generar una secuencia de palabras de salida, la cual es probabilísticamente plausible. En otras palabras, es un predictor de secuencias de palabras.

La operación de ChatGPT está organizada siguiendo los componentes básicos clásicos utilizados desde hace ya tiempo en el área del lenguaje natural, listados a continuación:

- En primer lugar, se encuentra la limpieza de los datos de texto y su normalización. La primera de estas actividades consiste en la eliminación sistemática de todo aquel contenido de texto que no aporte información, y la segunda involucra estandarizar y homogeneizar el contenido.
- Una segunda tarea corresponde a la separación de componentes léxicos (palabras y signos de puntuación principalmente), la cual lleva por nombre “tokenización”. Dado el hecho de que las computadoras son particularmente “hábiles” para trabajar con datos numéricos, no así con secuencias de caracteres, suele elaborarse una codificación numérica, normalmente arbitraria, para cada componente del vocabulario utilizado. Esto suele hacerse por economía de recursos, así como para facilitar el procesamiento. Para simplificar aún más la operación del modelo, los tokens son almacenados en una forma simplificada (e.g. para los verbos se elimina la conjugación, dejándolos en infinitivo).
- En tercer lugar, una de las partes primordiales de las tareas del PLN es la representación del texto de “alto nivel”. Si bien, una representación básica es transformar un token o palabras a un conjunto de valores numéricos, es necesario buscar técnicas que agreguen más información a sus significados. Hace algunos años se propusieron las representaciones distribucionales (embeddings), como las que genera el modelo Word2Vec (Mikolov, et al. 2013a; Mikolov, et al. 2013b). Una representación distribucional se puede ver como una lista de valores numéricos asociados a cada uno de los elementos que constituyen un corpus. Estos valores numéricos permiten llevar a cabo la asociación, de manera multidimensional de los diferentes términos, pudiendo así establecerse medidas de distancia, o de caracterización por nivel de similitud de éstos. Así, palabras que tienen significados similares y que comparten contexto, pueden generar una distancia corta en alguna de las dimensiones disponibles, tomando en cuenta esta representación numérica.

- Finalmente, en el cuarto lugar se encuentra una tarea realizada por humanos, llamada Aprendizaje por Refuerzo con Retroalimentación Humana (Reinforcement Learning from Human Feedback, o RLHF). Esto quiere decir, que se evalúan las salidas dadas por ChatGPT por medio de inteligencia humana, para poder premiar las mejores salidas y penalizar las cuales carecen de sentido. Cabe mencionar que éste es uno de los elementos clave para el éxito de ChatGPT.

Tomando como base las tareas anteriormente listadas, se procede a hacer uso de los ya mencionados *Transformers*, a través de los cuales se implementa un mecanismo denominado de “atención”, el cual permite guardar una forma de referencia “histórica”, una suerte de memoria amplia, de los elementos distintivos de una interacción, los cuales utilizan técnicas sofisticadas de redes neuronales artificiales, basadas en un modelo como BERT (Devlin, et al. 2018) y los Modelos de Lenguaje como los de la familia GPT, a través de los cuales se han logrado resultados impresionantes en el área.

La forma en que opera, en última instancia, ChatGPT, es a través de la aplicación de un método denominado “sampling” (muestreo), el cual lleva a cabo la generación de las respuestas, seleccionando la estructura de las frases a través de modificaciones seleccionadas aleatoriamente. Ésta es la razón por la cual las respuestas de ChatGPT tienen tan alto grado de variabilidad, y la probabilidad de obtener dos veces una respuesta idéntica a un mismo planteamiento es ínfima.

De esta manera es que ChatGPT es capaz de dar respuesta a preguntas de una manera “amable”, concisa y “coherente”, a los planteamientos hechos por sus usuarios. ChatGPT pasó por dos fases: (1) aprendizaje de los patrones de las secuencias de palabras y sus correspondientes significados que puede haber dentro de una base de datos y, (2) aprendizaje de la tarea a resolver. Para lograr esto, fue entrenado con cantidades ingentes de datos extraídos de recursos como:

- Common Crawl (<https://commoncrawl.org/>) (datos disponibles entre 2016 y 2019).
- WebText 2 (<https://paperswithcode.com/dataset/webtext>) (recursos principalmente extraídos de Reddit).

- Wikipedia (<https://www.wikipedia.org/>).

Así como una gran cantidad de libros disponibles en Internet. En cuanto al tema del aprendizaje de técnicas de programación de computadoras, uno de los principales recursos utilizados fue GitHub, propiedad de Microsoft, uno de los principales inversionistas de OpenAI, la casa madre de GPT. Sobra mencionar que, para lograr esto, fue necesario hacer uso de una cantidad inaudita de poder de cómputo.

La estructura interna de la red neuronal que se utilizó para implementar GPT no es clara. Sin embargo, lo que sí se sabe es que se hizo uso de 173 mil millones de parámetros (173 billones para el sistema anglosajón) para su entrenamiento, lo cual, aunado a la cantidad de texto utilizado para lograr tal objetivo, justifica el uso de tales cantidades de poder de cómputo. Se calcula que la cantidad de CO2 generado para entrenar a ChatGPT ronda los 1,287 MWh, y que el costo de operar ChatGPT se encuentra alrededor de los US\$700,000 por día (Anthony, Kanding y Selvan, 2020).

Concerniente el RLHF, OpenAI procedió a la contratación de una gran cantidad de “curadores”, los cuales tienen por labor llevar a cabo la calificación de las respuestas generadas por ChatGPT, de manera que, poco a poco, se pueda ir afinando la operación del sistema de manera estadística.

3. ChatGPT y el PC

La aplicación de técnicas basadas en la Inteligencia Artificial está a la orden del día, y sabiéndolo, o sin saberlo, constantemente hacemos uso de algoritmos que las implementan en las aplicaciones computacionales que tenemos a la mano, sea a través de los programas instalados en nuestros dispositivos, computadoras, tabletas o teléfonos, sea al acceder a recursos disponibles a través de la Web, como es el caso de las diversas redes sociales que existen, o el uso de máquinas de búsqueda.

Por otro lado, el PC está siendo objeto de cada vez más atención por parte del mundo profesional, al evidenciarse como

un recurso que ofrece alternativas interesantes no sólo para la programación de computadoras, sino como una herramienta valiosa para efecto de entrenar la mente humana para imaginar, sugerir y describir soluciones a problemas de la vida cotidiana de manera clara y concisa, de forma que sea fácilmente transmisible, implementable y operable.

Hacer uso de ChatGPT para efecto de lograr adquirir nuevos conocimientos y competencias es una actividad plausible. Sin embargo, es de suma importancia poner mucha atención a la manera en que se hace uso de estos recursos pues, habiendo sido entrenados, en una primera instancia, bajo paradigmas no supervisados, y posteriormente aplicando técnicas semi-supervisadas, pueden contener errores, y hasta aberraciones en cuanto a la manera en que se relacionan los términos a partir de los cuales se generan las respuestas a los planteamientos elaborados por los usuarios.

Así, dado el formato de entrenamiento base de ChatGPT, que lo único que hace es predecir las secuencias de palabras plausibles a ser generadas a la salida con base en la secuencia de palabras que recibe como entrada, pueden crearse asociaciones inverosímiles desde el entendimiento humano y la lógica, pues no existe garantía en cuanto a la calidad real de los textos usados para el entrenamiento. Y dado también el hecho de que el aprendizaje se hizo de manera no supervisada en su primera fase, y semi-supervisada en su segunda, es harto difícil tener certeza absoluta sobre el nivel de veracidad de las respuestas generadas. Adicionalmente, ChatGPT no tiene forma de establecer relaciones de naturaleza diferente a las secuencias de palabras, por lo que queda excluido cualquier tipo de razonamiento matemático, por ejemplo. Con base en lo anteriormente planteado, se puede sugerir que el uso de ChatGPT puede tener ventajas y desventajas que son importantes conocer antes de decidir aplicar o utilizar lo generado como respuesta.

Para efecto de poder ilustrar las implicaciones del uso de ChatGPT para el PC, se procedió a llevar a cabo una cierta cantidad de interacciones con la plataforma, y observar y analizar las respuestas obtenidas, con la finalidad de determinar, de entrada, si son correctas, y enseguida, su calidad.

Un primer caso se intentó que ChatGPT genere una solución algorítmica en pseudocódigo para resolver un sistema de ecuaciones lineales usando el método de Gauss-Jordan. Lo siguiente muestra la interacción realizada con ChatGPT:

Ejemplo 1

Entrada: ¿Puedes mostrarme el algoritmo del método de Gauss-Jordan?

Salida: Sí, el método de Gauss-Jordan es un algoritmo para resolver sistemas de ecuaciones lineales mediante la eliminación de Gauss y la eliminación de Gauss-Jordan. A continuación, se muestra el algoritmo en pseudocódigo:

1. *Inicializar una matriz aumentada A que contenga los coeficientes de las variables y los términos constantes de las ecuaciones del sistema.*
2. *Para cada fila i de A , haga lo siguiente:*
 - *Divida la fila i por el elemento $A[i][i]$ para obtener un 1 en la posición $A[i][i]$.*
 - *Para cada fila $j < i$, reste $A[j][i]$ multiplicado por la fila i de A de $A[j]$ para hacer 0 en la posición $A[j][i]$.*
3. *Para cada fila i de A en orden inverso, haga lo siguiente:*
 - *Para cada fila $j < i$, reste $A[j][i]$ multiplicado por la fila i de A de $A[j]$ para hacer 0 en la posición $A[j][i]$.*
4. *La solución del sistema se encuentra en la última columna de la matriz A .*

En resumen, el método de Gauss-Jordan es una variante del método de eliminación de Gauss que produce una matriz escalonada reducida, lo que simplifica la resolución de sistemas de ecuaciones lineales.

Como se puede observar, la solución generada tiene todo el aspecto de ser correcta, y sólo faltaría analizar minuciosamente el resultado, probar que funcione y llevarlo a la práctica en algún lenguaje de programación. Por otro lado, es necesario mencionar que además de la salida mostrada por ChatGPT, se describe de manera muy breve el funcionamiento del pseudocódigo, lo que puede ser conveniente, dentro de un proceso de aprendizaje del PC, para efecto

de comprender la lógica aplicada dentro de la solución propuesta.

Como ya se mencionó anteriormente, ChatGPT tuvo su entrenamiento usando una grandísima cantidad de datos, lo cual le confiere cierto “entendimiento” sobre lo expresado en la entrada, y lo que puede ser plausible de producir como salida. Todo esto, considerando únicamente los patrones encontrados durante dicho entrenamiento. Aunque es posible que el modelo genere datos incompletos, incorrectos, o inconsistentes, es deseable que la entrada sea rica en información para que sea posible producir una salida igualmente llena de detalles. El siguiente ejemplo, muestra un código generado por ChatGPT donde la entrada no necesariamente contiene información suficiente:

Ejemplo 2

Entrada: ¿Puedes escribirme el código del algoritmo NEAT?

Salida: Sí, aquí te muestro un ejemplo básico de implementación del algoritmo NEAT (NeuroEvolution of Augmenting Topologies) en Python utilizando la biblioteca NEAT-Python:

```
import neat

# Configuración del algoritmo NEAT
config = neat.Config(neat.DefaultGenome, neat.DefaultReproduction,
                    neat.DefaultSpeciesSet, neat.
                    DefaultStagnation,
                    'config_file.txt')

# Función de evaluación
def eval_genomes(genomes, config):
    for genome_id, genome in genomes:
        net = neat.nn.FeedForwardNetwork.create(genome, config)
        # Evaluar la red neuronal en una tarea específica
        genome.fitness = fitness_function(net)

# Crear el objeto de población
```

```
p = neat.Population(config)

# Anadir estadísticas de registro
stats = neat.StatisticsReporter()
p.add_reporter(stats)
p.add_reporter(neat.StdoutReporter(True))

# Ejecutar el algoritmo NEAT hasta que se alcance un criterio de
# finalización
winner = p.run(eval_genomes)

# Mostrar el resultado del ganador
print('\nMejor resultado:\n{!s}'.format(winner))
```

Este código configura el algoritmo NEAT, define una función de evaluación para evaluar el rendimiento de cada genoma y luego ejecuta el algoritmo hasta que se encuentre una solución satisfactoria. La biblioteca NEAT-Python se encarga de la mayoría de las tareas de bajo nivel, como la creación de la población y la actualización de la especie. La implementación real de la función de evaluación depende de la tarea específica que se está tratando de resolver.

Para este caso, ChatGPT imprime algunas líneas en las cuales se pueden observar varios comentarios (identificados por iniciar con el carácter '#'), donde se describe de manera breve lo que hace la instrucción que se encuentra inmediatamente abajo del mismo. Ésta es una práctica común, y deseable, por parte de los programadores de computadoras, y que ChatGPT incorpora como consecuencia misma de que los códigos usados para su entrenamiento contenían comentarios. Por otro lado, es necesario hacer hincapié en la función *fitness_function(net)*, la cual carece de implementación en el código actual. También, hay que hacer referencia al archivo de texto *config_file.txt*, que tampoco se muestra. En este sentido, ChatGPT especifica que dicha función depende del problema que se está

resolviendo, es decir, es necesario tener conocimiento del algoritmo que se implementa, y la manera en cómo usarlo para poder llevar a la práctica este código por parte del programador.

Aunque es un código, en lo general, correcto, se le puede considerar como incompleto, dada la ausencia de algunos de los componentes propios de la solución. Eventualmente sería conveniente si la respuesta fuera acompañada de una descripción, aunque sea somera, de los componentes que no forman parte de la solución, para así ganar intuición y entendimiento acerca del algoritmo o código propuesto. Alternativamente, podría agregarse más información a la entrada para enriquecerla, buscando así que se pueda generar una mejor salida.

Hay otros ejemplos donde ChatGPT genera una salida incompleta, situación por demás curiosa, pues no debería haber problemas para generar el código que dé solución a un problema relativamente simple. Por ejemplo, para este caso, se le pidió a ChatGPT generar un código en el lenguaje de programación C que implemente una red neuronal sencilla, sin embargo, como se puede apreciar en la salida generada por ChatGPT, éste se generó de manera incompleta:

Ejemplo 3

Entrada: Escribe en C una red neuronal simple.

Salida: Aquí te dejo un ejemplo simple de una red neuronal en C, con una capa oculta y una función de activación sigmoideal:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_INPUTS 2
#define NUM_HIDDEN 3
#define NUM_OUTPUTS 1

double inputs[NUM_INPUTS][2] = {{0.0, 0.0}, {0.0, 1.0}, {1.0,
    0.0}, {1.0, 1.0}};
double targets[NUM_OUTPUTS][4] = {{0.0}, {1.0}, {1.0}, {0.0}};
```

```
double hidden[NUM_HIDDEN];
double output[NUM_OUTPUTS];
double weights_ih[NUM_INPUTS][NUM_HIDDEN];
double weights_ho[NUM_HIDDEN][NUM_OUTPUTS];

double sigmoid(double x) {
    return 1.0 / (1.0 + exp(-x));
}

void initialize_weights() {
    int i, j;
    for (i = 0; i < NUM_INPUTS; i++) {
        for (j = 0; j < NUM_HIDDEN; j++){
            weights_ih[i][j] = (double)rand() / (double)RAND_MAX
- 0.5;
        }
    }

    for (i = 0; i < NUM_HIDDEN; i++) {
        for (j = 0; j < NUM_OUTPUTS; j++) {
            weights_ho[i][j] = (double)rand() / (double)RAND_MAX
- 0.5;
        }
    }
}

void forward() {
    int i, j;
    double sum;

    // calcular los valores de las neuronas en la capa oculta
    for (i = 0; i < NUM_HIDDEN; i++) {
        sum = 0.0;
```

```
        for (j = 0; j < NUM_INPUTS; j++) {
            sum += inputs[j][0] * weights_ih[j][i];
        }
        hidden[i] = sigmoid(sum);
    }

    // calcular el valor de la neurona de salida
    sum = 0.0;
    for (i = 0; i < NUM_HIDDEN; i++) {
        sum += hidden[i] * weights_ho[i][0];
    }
    output[0] = sigmoid(sum);
}

void backward() {
    int i, j;
    double delta_h[NUM_HIDDEN];
    double delta_o[NUM_OUTPUTS];
    double error = 0.0;

    // calcular el error en la neurona de salida
    error = targets[0][0] - output[0];
    delta_o[0] = error * output[0] * (1.0 - output[0]);

    // calcular los errores en la capa oculta
    for (i = 0; i < NUM_HIDDEN; i++) {
        delta_h[i] = 0.0;
        for (j = 0; j < NUM_OUTPUTS; j++) {
            delta_h[i] += delta_o[j] * weights_ho[i][j];
        }
        delta_h[i] *= hidden[i] * (1.0 - hidden[i]);
    }
}
```

```
// actualizar los pesos de la capa oculta
for (i = 0; i < NUM_HIDDEN; i++) {
    for (j = 0; j < NUM_OUTPUTS; j++) {
        weights_ho[i][j] += delta_o[j] * hidden[i];
    }
}

// actualizar los pesos de la capa de entrada
for (i = 0; i
```

Es importante citar también casos que podrían considerarse emblemáticos, y que dejan ver claramente que, si bien las soluciones generadas tienden a ser apropiadas en cuanto a las secuencias de texto generadas a la salida, no necesariamente es así. De esta manera, hay ocasiones en las que se generan respuestas que, a primera vista parecen ser coherentes, pero que, al analizarlas un poco más detalladamente, se evidencia que son inexactas. En este ejemplo, se expone una respuesta donde ChatGPT se muestra totalmente convincente, sin embargo, hay un error. Esta es la interacción completa:

Ejemplo 4

Entrada: ¿Puedes mostrarme un ejemplo del uso de NEAT para Bin Packing Problem?

Salida: ¡Por supuesto! A continuación, te muestro un ejemplo de cómo utilizar el algoritmo NEAT para resolver el problema de Bin Packing.

Primero, necesitamos importar la biblioteca neat-python y las bibliotecas adicionales necesarias para crear la población, evaluar los genomas, etc. Además, necesitamos definir algunas variables globales, como la capacidad del bin, la cantidad de objetos y la cantidad de generaciones que deseamos que se ejecute el algoritmo. El código se ve así:

```
import random

import numpy as np

import matplotlib.pyplot as plt

import math

import neat
```

```
CAPACITY = 10

NUM_ITEMS = 20

NUM_GENERATIONS = 100
```

A continuación, necesitamos definir una función que calcule la aptitud de cada genoma en la población. En este ejemplo, la función de aptitud mide qué tan bien puede empaquetar los objetos en el menor número de bins posible.

```
def eval_genomes(genomes, config):

    for genome_id, genome in genomes:

        net = neat.nn.FeedForwardNetwork.create(genome, config)

        bins = [[]]

        bin_capacity_left = [CAPACITY]

        for item in sorted(items, reverse=True):

            item_placed = False

            for bin_idx, bin_cap in enumerate(bin_capacity_

left):

                if item <= bin_cap:

                    bins[bin_idx].append(item)

                    bin_capacity_left[bin_idx] -= item

                    item_placed = True

                    break
```

```
if not item_placed:
    bins.append([item])
    bin_capacity_left.append(CAPACITY - item)
```

```
genome.fitness = len(bins)
```

En esta función, iteramos a través de cada genoma en la población y creamos una red neuronal utilizando la función create de la clase FeedForwardNetwork. Luego, ejecutamos el algoritmo de empaquetado del bin y medimos la aptitud del genoma en función de la cantidad de bins utilizados para empaquetar los objetos.

Finalmente, necesitamos definir la función principal que creará y ejecutará la población. El código se ve así:

```
def run_neat(config_file):
    config = neat.Config(neat.DefaultGenome, neat.
        DefaultReproduction, neat.DefaultSpeciesSet, neat.
        DefaultStagnation, config_file)

    p = neat.Population(config)

    # Add reporters to show progress in the terminal and save to
    # file
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)
    p.add_reporter(neat.StdoutReporter(True))
    p.add_reporter(neat.Checkpointer(5, None))

    winner = p.run(eval_genomes, NUM_GENERATIONS)

    print('\nBest genome:\n{!s}'.format(winner))
```

```
# Show final stats
print('\nFinal statistics:\n{!s}'.format(stats))
```

En esta función, creamos una instancia de la clase `neat.Config` y luego creamos una población utilizando la clase `Population`. A continuación, agregamos algunos reporteros que nos mostrarán el progreso del algoritmo y guardaremos los mejores genomas encontrados. Luego, ejecutamos el algoritmo utilizando la función `run` de la población y la función `eval_genomes` que definimos anteriormente.

Por último, imprimimos el genoma ganador y algunas estadísticas finales.

Este es solo un ejemplo básico de cómo utilizar el algoritmo NEAT para resolver el problema de Bin Packing.

A este respecto, dirigimos la atención hacia la función `eval_genomes`, donde se puede ver que en ella existe una solución incorrecta. El problema de Bin Packing, es conocido normalmente dentro de la optimización combinatoria, y últimamente éste ha sido resuelto usando diferentes técnicas como los algoritmos genéticos de agrupación (Quiroz-Castellanos, et al. 2015). Es decir, este problema se encuentra dentro del espacio discreto, y es posible que no exista una solución dada por una red neuronal, aunque con algunos ajustes y modificaciones, se pueden llegar a predecir soluciones como en el caso de AlphaFold usado para predecir estructuras de proteínas (Jumper, et al. 2021).

En este mismo orden de ideas, y volviendo a la función que se está analizando, dentro de ella se crea una variable `net`, que guarda un modelo neuronal, además dicha función guarda un valor importante dentro de la variable `genome.fitness`, que le dice al algoritmo qué tan buena es una solución. Sin embargo, la red neuronal almacenada en `net` nunca es utilizada para resolver el problema que se propone, que en este caso es el Bin Packing. Dicha red neuronal es necesario evaluar dado que se requiere probar qué tan bien funciona. En consecuencia, esto generaría un error al tratar de resolver el problema principal.

De cierta manera, ChatGPT entiende que el Bin Packing fue resuelto con algoritmos genéticos y trata de combinarlo con la información escrita para NEAT, que, a su vez, también es un algoritmo genético (Stanley y Miikkulainen, 2002). Finalmente, ChatGPT produce una salida que contiene un error, pero que se lee sumamente convincente gracias a la información y los datos de entrenamiento.

Finalmente, en un último ejemplo es interesante de presentar, y que comprende un tema de mucha importancia para el PC: la lógica y las matemáticas, donde se puede observar que, aunque el razonamiento sea atractivo y contundente, los valores resultantes son, a todas luces, erróneos:

Ejemplo 5

Entrada: Cuando tenía 10 años, mi hermana tenía la mitad de mi edad. Ahora ella tiene 22 años, ¿qué edad tengo?

Salida 1: Cuando tenías 10 años, tu hermana tenía la mitad de tu edad, lo que significa que ella tenía 5 años menos que tú. Por lo tanto, en ese momento tenía $10 - 5 = 5$ años de edad.

Ahora que ella tiene 22 años, han pasado 12 años desde entonces. Por lo tanto, tu edad actual es de $10 + 12 = 22$ años. ¡Tú y tu hermana ahora tienen la misma edad!

Y con la esperanza en que ChatGPT diera una solución correcta se le volvió a cuestionar lo mismo, y generó lo siguiente.

Ejemplo 6

Salida 2: Cuando tenías 10 años, tu hermana tenía la mitad de tu edad, lo que significa que ella tenía 5 años en ese momento.

Ahora, han pasado 12 años desde entonces y tu hermana tiene 22 años, lo que significa que tú tienes $22 + 12 = 34$ años ahora.

Inicialmente dicha respuesta pareciera correcta, sin embargo, se evidencia que no es así si se hacen algunos despejes de ecuaciones sencillos, y se hace uso de una calculadora por cuenta propia.

Éstos son sólo algunos de los ejemplos, de entre muchos otros, que fueron considerados como más emblemáticos para ilustrar el uso de ChatGPT dentro del ámbito del PC, y a través de los cuales pudimos demostrar que es de suma importancia ser muy cuidadoso,

pues se corre el riesgo de que se asuma información errónea o incompleta, como adecuada para los fines del usuario.

4. Las buenas prácticas de ChatGPT para aprender PC

No es del todo posible crear un “manual” completo de las buenas prácticas en el uso de herramientas generativas de texto. Sin embargo, sí que es posible al menos definir algunos lineamientos generales que nos permitan mejorar su uso en nuestro beneficio. Un elemento que es fundamental es la aplicación del criterio y del pensamiento crítico al momento de analizar las propuestas generadas a través de esta herramienta.

Como se pudo apreciar a través de los ejemplos con los cuales pretendimos ilustrar algunos de los casos emblemáticos de uso de ChatGPT a lo largo del presente texto, las propuestas algorítmicas de ChatGPT en ocasiones son incompletas, y hasta incorrectas, por lo que no se pueden asumir tal cual.

El análisis minucioso de cada solución es una acción imperativa que debe ser sistemáticamente aplicada, primero a través del análisis exploratorio, y posteriormente a través de las pruebas de rigor del caso, sea bajo la forma de prueba de escritorio, sea bajo la forma de ejecución del código generado para casos emblemáticos conocidos, y una vez hecho esto, corroborar que la propuesta sea consistente con el resto de la solución, en caso de que corresponda a un sub-problema.

Por otro lado, hacer uso de las soluciones generadas por ChatGPT como elemento de aprendizaje, comparando diversas soluciones generadas de manera automática, y comparando sus características de diseño, y su desempeño, siempre será un recurso valioso para ejecutar una estrategia de aprendizaje a través de ejemplos, contraponiendo la lógica de las soluciones generadas por ChatGPT, contra la lógica propia de crear una solución a un problema.

Como parte del proceso, puede ser de gran interés llevar a cabo varias interacciones sobre la generación de soluciones de un mismo problema, y compararlas, buscando afinar la manera en que se hace

el planteamiento del problema, o incluso elaborando peticiones cada vez más específicas, y comparando los resultados, o incorporándolos sucesivamente, hasta lograr una comprensión completa del problema, y al mismo tiempo, una solución al mismo.

Cabe mencionar que, a pesar de que ChatGPT suele ser bueno, a causa del mecanismo implementado a través de los transformers, para establecer relaciones entre textos que no son estrictamente aledaños, es bastante limitado para generar respuestas largas, en las cuales, o bien comienza a mezclar ideas y conceptos, o bien simplemente trunca la respuesta.

5. Conclusiones

ChatGPT, en tanto que herramienta auxiliar para el aprendizaje de los elementos propios del PC es, sin duda alguna, un recurso valioso, pues permite al aprendiz hacerse de recursos que coadyuvan en su aprendizaje, comprensión y profundización de los conceptos necesarios.

Sin embargo, existe un riesgo asociado al uso a la ligera de tal recurso pues, como se ha planteado a lo largo de este texto, las soluciones que generadas no siempre son correctas, o simplemente consistentes de acuerdo al contexto dentro del cual se ha de insertar. El PC es una actividad inherentemente humana, y se requiere de un entrenamiento para adquirirlo, el cual puede verse afectado si no se pone atención debido a la influencia de nuevas tecnologías.

Es por esto que se debe entrenar y mantener permanentemente el pensamiento crítico, de forma que sistemáticamente se proceda a analizar, y eventualmente, incorporar las soluciones propuestas por esta herramienta, sólo en caso de que se consideren apropiadas. Tomar a la ligera nunca es una vía apropiada para el PC, el cual debe nutrirse permanentemente, a través de la incorporación y mejora de habilidades.

Aún queda mucho camino que recorrer en estas áreas de trabajo, que son realmente innovadoras en sí, y sólo será el tiempo, y la práctica, lo que nos proporcionará el entendimiento necesario para facilitar la adquisición de las habilidades asociadas con el PC.

6. Referencias

- Anthony, L.F.W., Kanding, B. and Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. arXiv preprint arXiv:2007.03051.
- Champollion, J.F. (1828). Précis du système hiéroglyphique des anciens Egyptiens, ou Recherches sur les éléments premiers de cette écriture sacrée, sur leurs diverses combinaisons, et sur les rapports de ce système avec les autres méthodes graphiques égyptiennes avec un volume de planches. Imprimerie royale.
- Dalby, A. (2019). Rosetta Stone. WikiJournal of Humanities, 2(1), p.1. <https://doi.org/10.15347/wjh/2019.001>.
- Denning, P.J. and Tedre, M. (2019). Computational thinking. MIT Press.
- Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Eisenstein, J. (2018). Natural language processing. Available at: <https://princeton-nlp.github.io/cos484/readings/eisenstein-nlp-notes.pdf> [Accessed 14 June 2022].
- Jumper, J. et al. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873), pp.583-589. <https://doi.org/10.1038/s41586-021-03819-2>.
- Lodi, M. and Martini, S. (2021). Computational Thinking, Between Papert and Wing. Science & Education, 30(4), pp.883-908. <https://doi.org/10.1007/s11191-021-00202-5>.
- Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.
- Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gómez, C., Huacuja, H.J.F. and Alvim, A.C.F. (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. Computers & Operations Research, 55, pp.52-64. <https://doi.org/10.1016/j.cor.2014.10.010>.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), p.9.
- Stanley, K.O. and Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. Evolutionary Computation, 10(2), pp.99-127. <https://doi.org/10.1162/106365602320169811>.
- Vaswani, A. et al. (2017). Attention is all you need. Advances in neural information processing systems, 30.
- Wertheimer, T. (2022). Blake Lemoine: Google despierte al ingeniero que aseguró que un programa de inteligencia artificial cobró conciencia propia. BBC News, 23 July.