

Aplicación de la Inteligencia Artificial en el Análisis Sintáctico de Compiladores

Application of Artificial Intelligence to Compiler Syntax Analysis

José Tomas Martínez López

Resumen. En la actualidad, nos encontramos en una era donde la tecnología está presente en sus múltiples formas: redes, inteligencia artificial, el software es cada vez más complejo, el hardware obliga a que el programa este a la misma altura que la capacidad del hardware y, por lo tanto, en la barrera para los programadores y para los docentes. Para obtener el máximo rendimiento de las ventajas y de las posibilidades que los diferentes productos tecnológicos ofrecen, es necesario que estos se usen para elaborar programas de software eficientes, seguros y de calidad. Los compiladores son un tipo de programa de gran relevancia en la elaboración de software, ya que hacen posible la ejecución de un programa al transformar un lenguaje de programación en uno que la maquina pueda interpretar, y de esta manera, hacen posible que un programa se ejecute correctamente. La meta de este artículo es demostrar las ventajas que la inteligencia artificial puede aportar en la elaboración de compiladores y como la inteligencia artificial puede afectar positivamente el proceso, específicamente en el análisis sintáctico de un compilador.

Palabras Clave. Inteligencia Artificial, compiladores, software, análisis sintáctico

Abstract. Currently, we are in a period where technology permeates every aspect of our lives—networks, artificial intelligence, and sophisticated software. This rapid advancement in software is driven by the continuous evolution of hardware, which compels programs to match its increasing capabilities. However, this also creates a significant hurdle for programmers and educators alike. To maximize the benefits and opportunities presented by various technological tools, it is crucial to leverage them in developing high-quality, secure, and efficient software. Compilers are indispensable in software creation. They function by translating code written in a high-level programming language into machine-understandable instructions, thus enabling programs to run effectively. The primary objective of this article is to illustrate the potential enhancements artificial intelligence can bring to compiler development. We will explore how AI can streamline and improve different stages of this process, with particular emphasis on the syntactic analysis phase of a compiler.

Keywords. Artificial Intelligence, compilers, software, syntactic analysis

INTRODUCCION

Dada la dificultad de crear software de alta calidad que sirva como herramienta en la ejecución de las tareas y actividades diarias, surge el interés de investigar y mostrar los beneficios de aplicar inteligencia artificial en la elaboración del análisis sintáctico de un compilador. Es bien sabido que existen diversos métodos para lograr este fin; sin embargo, en virtud de los beneficios y propiedades que ofrece la inteligencia artificial como un medio para fomentar un desarrollo más eficiente, y que nos brinda la oportunidad de explorar nuevas soluciones en el desarrollo de software.

La inteligencia artificial en el desarrollo de software

En los últimos años, la inteligencia artificial ha adquirido una gran importancia en las áreas de la tecnología y en la elaboración de software. La información de las investigaciones, la capacidad de encontrar similitudes y de obtener resultados de distinta naturaleza, han hecho que la inteligencia artificial se use como medio de soporte en las actividades que hasta hace poco eran realizadas por los programadores. La inteligencia artificial ha llegado a las distintas fases del ciclo de vida del software: en la planificación, en el análisis de requisitos, en el diseño, en el desarrollo, en la prueba y en la Mantenimiento (Durrani et al., 2024). A causa de esta situación, se han abierto nuevas oportunidades para los profesionales que construyen sistemas informáticos y, a su vez, se han generado nuevos retos.

Con las herramientas que utilizan inteligencia artificial, es posible obtener propuestas de código, encontrar posibles errores, explicar las estructuras de los programas y acompañar al programador en las distintas fases del proceso. No obstante, hay que tener en cuenta que los resultados que se obtienen de las herramientas no serán válidos hasta que no sean revisados y validados, ya que no se garantiza que el software que se obtiene sea el correcto, el eficiente o el adecuado para las necesidades del proyecto.

Los compiladores son una de las herramientas más relevantes en el desarrollo de software, ya que hacen posible transformar un programa de un lenguaje de programación a una representación que puede ser interpretada y ejecutada por un sistema informático. Un compilador pasa por varias fases para lograr esta transformación, estas fases incluyen el análisis léxico, el análisis sintáctico y el análisis semántico.

La importancia del análisis sintáctico del compilador

La fase de análisis sintáctico tiene un papel fundamental en el proceso, ya que tiene como objetivo determinar si la secuencia de elementos que se obtienen en el análisis léxico cumple con las reglas que se han fijado en la gramática del lenguaje. Es decir, tiene como objetivo comprobar si las instrucciones de un programa tienen una forma válida en función de las reglas de ese lenguaje. Esta fase es fundamental ya que los errores sintácticos deben ser detectados al principio del proceso de compilación para que el proceso pueda continuar. Los métodos y algoritmos para la realización del análisis sintáctico se han usado en los compiladores desde hace muchos años. Entre los métodos se pueden mencionar los analizadores descendentes y los analizadores ascendentes y los métodos para saber si una secuencia de tokens pertenece a un lenguaje que se define por medio de una gramática. La base de los métodos que se usan en la teoría de compiladores es la base de los métodos para hacer herramientas que analizan de forma estructurada los programas (Scott, E., & Johnstone, A. 2026). Sin embargo, las nuevas herramientas que se basan en inteligencia artificial hacen posible que las tecnologías de inteligencia artificial se usen como soporte en las actividades que tienen que ver con la elaboración de los componentes del análisis sintáctico.

El uso de IA en la construcción de un compilador

La idea de que algunos sistemas de inteligencia artificial puedan encontrar patrones y elaborar código nos hace pensar en la posibilidad de que las tecnologías de inteligencia artificial se usen en actividades que tienen que ver con la elaboración de un compilador. Si fuera posible usar inteligencia artificial para elaborar un

analizador sintáctico, sería muy interesante ya que elaborar un analizador sintáctico puede requerir conocimientos especializados en gramáticas, en estructuras de datos, en algoritmos y en lenguajes de programación. Elaborar estos componentes puede ser un trabajo largo y difícil para una persona que está aprendiendo o para un programador con experiencia si no tiene conocimientos suficientes sobre compiladores.

Por lo tanto, la inteligencia artificial puede ser una ayuda para dar soluciones iniciales, para explicar procedimientos o para buscar posibles errores en el código elaborado. Las últimas investigaciones han encontrado que los modelos de lenguaje pueden ser interesantes para hacer y cambiar código, aunque también hay problemas de confianza y de consistencia en los resultados que se obtienen (Zhong & Wang, 2024; Ouyang et al., 2025). Pero si se usa inteligencia artificial para hacer un analizador sintáctico, no se pueden olvidar los conocimientos de compiladores. Al contrario, los conocimientos son necesarios para poder ver y arreglar los resultados que se obtienen de la herramienta. Una parte de código que se obtiene automáticamente puede tener errores, puede no tener en cuenta algunas de las reglas de la gramática o puede dar una solución que sirve para algunos casos, pero no para todos. La literatura más actual dice que el código que se obtiene de los modelos de lenguaje no debe creerse automáticamente fiable, aunque pueda ejecutarse, porque puede tener problemas de estabilidad, un mal uso de componentes y otros errores que necesitan ser revisados por un humano (Zhong & Wang, 2024). También, se ha visto en la literatura que los resultados de los modelos pueden ser diferentes para el mismo problema, y esto puede hacer que el código que se hace no sea reproducible ni consistente (Ouyang et al., 2025).

Por todo esto, es importante estudiar la relación entre los métodos que se han usado por muchos años para el análisis sintáctico y las oportunidades que se presentan ahora con la inteligencia artificial. La meta no es cambiar los métodos y los procedimientos que se han usado para hacer compiladores, sino estudiar cómo se pueden juntar los métodos y los procedimientos con las herramientas que pueden hacer más rápido algunas de las actividades del proceso de hacer. Los métodos y

los procedimientos tradicionales para el análisis sintáctico serán una base importante para entender y validar las estructuras que se obtienen en el futuro (LR y Earley). (Yamane, 2025; Scott & Johnstone, 2026). La calidad del software que se elabora es otro punto que hay que tener en cuenta. No es suficiente obtener un programa que se pueda ejecutar cuando se hace un sistema informático; el programa debe tener algunas características como eficiencia, mantenimiento, claridad, fiabilidad, capacidad de respuesta a las distintas situaciones para las que fue diseñado.

Los modelos de calidad del software ISO/IEC 25010 nos permiten estudiar las distintas propiedades de calidad y establecer los criterios para calificar un producto de software en función de las necesidades y del ambiente de uso (Mulyawan et al., 2021). Por lo tanto, cualquier propuesta que use inteligencia artificial para hacer algunos componentes de un compilador debe tener algunos mecanismos para comprobar que el resultado cumple con las propiedades esperadas.

La necesidad de hacer pruebas y evaluaciones tiene aún más relevancia cuando el código se obtiene de forma automática, ya que la posibilidad de que un programa se ejecute no garantiza por sí misma que sea robusto, confiable o apto para todos los casos de uso (Zhong & Wang, 2024). Con respecto al análisis sintáctico, una de las opciones a investigar es la inteligencia artificial y la capacidad que tiene para ayudar a elaborar las estructuras para reconocer algunas reglas gramaticales. Esta ayuda puede ser elaborar código, proponer reglas, encontrar errores sintácticos o explicar los procedimientos que se han seguido para elaborar el analizador.

El objetivo de esta investigación es conocer el grado de las capacidades que pueden ser más interesantes que los procedimientos que se siguen para elaborar un analizador. Además, si se usa inteligencia artificial, se abren algunas oportunidades en el ámbito educativo, en las asignaturas que tienen que ver con lenguajes y autómatas, con la teoría de compiladores y con la programación. Las

últimas investigaciones muestran que los estudiantes usan herramientas como ChatGPT para revisar errores, para depurar código, para hacer soluciones, para mejorar programas y para entender conceptos de programación (Groothuijsen et al., 2024). Si un estudiante le pide a una herramienta de inteligencia artificial que le dé un analizador sintáctico, el estudiante podrá ver las distintas formas de hacer el analizador y, después, podrá comprobar si la solución que le da la herramienta es compatible con los principios teóricos que ha estudiado. En este sentido, la herramienta puede ser un soporte para que el estudiante pruebe las distintas soluciones, siempre que el estudiante haga un proceso de análisis y validación de las respuestas. Varios estudios hechos en las asignaturas de programación han mostrado la importancia de crear estrategias educativas que dejen usar la inteligencia artificial sin olvidar el desarrollo de las habilidades de programación y del razonamiento algorítmico (Dickey et al., 2024; Kosar et al., 2024). Desde este punto de vista, la inteligencia artificial puede verse como una herramienta de soporte que permite probar nuevas formas de hacer software. Para poder usar la inteligencia artificial, es necesario que haya un proceso de análisis y de evaluación que permita saber cuáles son las ventajas y cuáles son las limitaciones de la inteligencia artificial. Esto es muy importante cuando se trabajan con componentes de un compilador que son clave, ya que un error en la estructura que se hace puede tener efectos en las siguientes fases del proceso.

Por lo dicho antes, si se investiga el uso de la inteligencia artificial en la elaboración del análisis sintáctico, se entenderán mejor las posibilidades que se presentan con las tecnologías de inteligencia artificial en el campo de los compiladores. Al mismo tiempo, se pueden encontrar algunos procedimientos que pueden hacer más cortos los tiempos de elaboración, que pueden hacer más fácil la elaboración de prototipos y que pueden ayudar a los programadores y a los estudiantes a entender los algoritmos que se utilizan. Los datos que se tienen muestran claramente que las herramientas de IA pueden hacer más rápidas algunas partes de la programación, pero la velocidad de las herramientas de IA depende directamente de cómo se usan las herramientas de IA y de los mecanismos de revisión que se ponen en marcha en el proceso (Groothuijsen et al., 2024; Kuhail et

al., 2024). La investigación tiene como objetivo investigar el papel de la inteligencia artificial en la elaboración del análisis sintáctico de un compilador, teniendo en cuenta las ventajas que puede tener la inteligencia artificial en comparación con los procedimientos que se usan habitualmente.

La investigación también tiene como objetivo indagar las posibilidades de uso de las herramientas y las limitaciones que se deben tener en cuenta para garantizar que el código que se elabora sea revisado y validado después; con esta información se puede decidir si el uso de inteligencia artificial puede hacer mejor el proceso de elaboración de analizadores sintácticos y, de esta forma, ayudar a elaborar software de una forma más eficiente.

En este trabajo no se considera la inteligencia artificial como un reemplazo del conocimiento especializado, sino como una herramienta que, si se usa de la forma adecuada, puede completar los conocimientos de la teoría de compiladores y abrir nuevas posibilidades para elaborar y para investigar en este campo.

TRABAJOS RELACIONADOS

La mezcla de la inteligencia artificial y la teoría de compiladores no es algo nuevo, aunque los objetivos y los métodos han cambiado mucho en los últimos diez años. Las técnicas de inteligencia artificial en compiladores se usaban principalmente en el análisis semántico avanzado y, sobre todo, en la optimización del código de la fase de creación de código máquina. La aparición de la Inteligencia Artificial Generativa y de los Modelos de Lenguaje de Gran Escala (LLMs) ha cambiado mucho las posibilidades, y ahora la investigación se centra en el principio de la compilación, es decir, en el análisis léxico y sintáctico.

Aplicaciones Clásicas de IA en Compiladores:

Los algoritmos de aprendizaje automático (Machine Learning) se han usado en el pasado para hacer compiladores y para solucionar problemas de optimización NP-hard. Los trabajos de Cooper et al. (2002) y Stephenson et al. (2003) son ejemplos

de trabajos de investigación en los que se usaron algoritmos genéticos y aprendizaje por refuerzo para solucionar el problema del orden de las fases de compilación y para asignar los registros en la arquitectura del procesador. Los componentes del compilador que pertenecen al Front-End (las fases que incluyen el análisis léxico, el análisis sintáctico y el análisis semántico) no se han investigado con aprendizaje automático en las aproximaciones clásicas. Esto se debe a que el análisis sintáctico tiene una base matemática muy fuerte y determinista que se rige por las gramáticas libres de contexto (GLC) de la jerarquía de Chomsky y por los algoritmos que tienen garantías teóricas muy precisas como LL(k), LR(k) o LALR(1) (Aho et al., 2006). Por este motivo, los generadores formales como Yacc, Lex, Bison y ANTLR (Parr, 2013) han tomado el control de la automatización en esta fase por muchos años.

Modelos de Lenguaje de Gran Escala y Código Fuente La arquitectura Transformer (Vaswani et al., 2017) y los modelos de lenguaje hechos a partir de ella para procesar código fuente (CodeBERT (Feng et al., 2020), Codex (Chen et al., 2021) y CodeLlama (Rozière et al., 2023)) hicieron un cambio en la ingeniería de software. Estos modelos pueden aprender la sintaxis y la semántica de muchos lenguajes de programación gracias a que se entrenan con millones de repositorios.

Varios trabajos de investigación han estudiado lo bien que pueden los modelos de lenguaje de gran escala aprender la estructura sintáctica del código:

- Creación Automática de Código: Los trabajos de Liu et al. (2023) muestran que herramientas como GitHub Copilot pueden hacer funciones completas con una precisión sintáctica muy alta. Pero los autores advierten que los errores que pueden aparecer cuando se usan reglas gramaticales muy complejas o poco conocidas pueden ser difíciles de encontrar.
- Reparación Automática de Programas (Automated Program Repair - APR): Las investigaciones más nuevas (Fan et al., 2023; Sobania et al., 2023) muestran que los LLMs son mejores que las técnicas de reparación antiguas para encontrar y corregir los errores en los programas, pero la mayoría de los trabajos de investigación solo se ocupan de los errores lógicos o semánticos a nivel de la

aplicación y no de los errores sintácticos que pueden aparecer al hacer el análisis sintáctico de un compilador.

IA Generativa en el Análisis Sintáctico y en el Ámbito Educativo

Actualmente, la investigación en análisis sintáctico tiene dos líneas de investigación principales:

1. Elaboración y Reestructuración de Parsers: Autores como Denny et al. (2023) demuestran que los modelos generativos pueden ser buenos ayudantes para convertir las especificaciones gramaticales en Notación BNF en código ejecutable (por ejemplo, analizadores descendentes recursivos) en lenguajes como Python o C++. Elaborar un parser con inteligencia artificial tiene ventajas sobre los generadores deterministas nuevos como ANTLR o Bison, ya que el tiempo para empezar es mucho más corto. Pero Denny et al. (2023) advierten del peligro de la "alucinación sintáctica", ya que el modelo puede hacer código que parece gramaticalmente correcto, pero no lo es (por ejemplo, puede no tener en cuenta la asociatividad o el control de la recursividad a la izquierda).

2. Valor Pedagógico: La Literature consultada (Prather et al., 2023; Kazemitabaar et al., 2023; Denny et al., 2023) muestra de forma consensuada que la inteligencia artificial tiene un alto valor pedagógico en las asignaturas de Lenguajes y Autómatas y Compiladores:

- Tutoría Interactiva: La IA actúa como un facilitador que permite a los estudiantes ver paso a paso cómo se hacen los Árboles de Sintaxis Abstracta (AST) y entender la jerarquía de Chomsky.

- Baja la Frustración Inicial: El análisis de errores contextualizados permite a los estudiantes de ingeniería encontrar sus errores de sintaxis sin esperar al profesor y, de esta forma, los estudiantes aprenden más rápido al principio de la programación de compiladores. Trabajo Actual Este artículo investiga empíricamente cuánto puede la IA generativa complementar o hacer más rápido la

elaboración del analizador sintáctico de un compilador sin afectar negativamente la base teórica necesaria en la ingeniería de software. A partir de la revisión bibliográfica se puede ver que hay mucha investigación sobre la inteligencia artificial y la creación de código en general, pero poca investigación sobre la confianza teórica, la precisión determinista y la eficiencia de los LLM cuando se hacen los componentes del análisis sintáctico formal y cuando se comparan con las herramientas que se usan habitualmente (ANTLR o Bison).

METODOLOGÍA

Se usó una metodología cualitativa de revisión sistemática de la literatura (Kitchenham & Charters, 2007) para este artículo. La metodología tiene como objetivo recoger, estudiar y hacer un resumen del conocimiento científico que existe sobre el uso de la inteligencia artificial en el análisis sintáctico de los compiladores y sobre el efecto de la inteligencia artificial en la educación.

Preguntas de Investigación

La búsqueda bibliográfica se hizo para contestar a las siguientes preguntas de investigación:

- PI1: ¿Cómo se integran las técnicas de Inteligencia Artificial Generativa y de Grandes Modelos de Lenguaje (LLM) en la fase de análisis sintáctico de un compilador?
- PI2: Según la literatura, ¿qué ventajas, desventajas y riesgos teóricos se han dicho que existen al cambiar o al usar la inteligencia artificial como soporte en los analizadores sintácticos deterministas (LL, LR)?
- PI3: Según los estudios más nuevos, ¿qué impacto pedagógico se ha dicho que tiene usar la IA como herramienta de soporte para enseñar el análisis sintáctico y la teoría de autómatas?

Estrategia de Búsqueda y Fuentes de Información

Se buscaron documentos en las bases de datos académicas y en los repositorios científicos más importantes de las ciencias de la computación y de la ingeniería: IEEE Xplore, ACM Digital Library, Scopus, ScienceDirect y Google Scholar.

Se usaron ecuaciones de búsqueda con operadores boléanos (AND, OR) y se juntaron los siguientes términos en español y en inglés:

- ("Inteligencia Artificial" OR "IA Generativa" OR "LLM") AND ("Análisis Sintáctico" OR "Parser" OR "Compiladores") • ("Construcción de Compiladores" OR "Parsing") AND ("Generación de Código" OR "Recuperación de Errores")
 - ("IA Generativa") AND ("Educación en Ciencias de la Computación" OR "Teoría de Automatas")
- Criterios de Selección (Inclusión y Exclusión) Se aplicaron los siguientes filtros para la selección de los documentos y para asegurar que la calidad y la importancia académica de los documentos eran las correctas:

Criterios de Inclusión:

1. Artículos de revistas con revisión por pares y actas de congresos internacionales con revisión por pares.
2. Publicaciones preferentemente de los años 2018-2026 (para abarcar el auge de las arquitecturas Transformer y de los LLM).
3. Trabajos que tratan sobre la teoría de compiladores, sobre la elaboración de parsers, sobre el diagnóstico de errores sintácticos o sobre la educación en programación con IA.

Criterios de Exclusión:

1. Publicaciones sin revisión por pares, entradas de blog o documentación técnica no académica.

2. Artículos que solo tratan sobre la optimización de código máquina y no sobre el análisis sintáctico.

Análisis y Extracción de Datos

Los artículos que se eligieron se clasificaron y se compararon en base a tres temas principales:

1. La capacidad técnica y la precisión de los modelos generativos al interpretar o hacer estructuras sintácticas.
2. Los límites de la fiabilidad y los riesgos teóricos (la ambigüedad gramatical y la falta de garantías deterministas).
3. Los beneficios pedagógicos en la formación de los estudiantes de ingeniería.

RESULTADOS Y DISCUSIÓN

Se extrajeron y se resumieron los resultados más importantes de la literatura académica revisada (2018-2026) para contestar a las tres preguntas de investigación que se hicieron en la metodología.

Integración de la IA en el Análisis Sintáctico (Respuesta a PI1)

La literatura consultada muestra que la Inteligencia Artificial Generativa y los Modelos de Lenguaje de Gran Escala (LLMs) se han usado en el análisis sintáctico de los compiladores de tres formas diferentes:

1. Elaboración de Analizadores Sintácticos con el soporte de la IA: Muchos autores (Denny et al., 2023; Chen et al., 2021) dicen que los LLMs pueden transformar las especificaciones gramaticales en Notación BNF en código ejecutable en lenguajes como Python o C++ usando algoritmos de descenso recursivo. La inteligencia artificial puede hacer mucho más corto el tiempo que se

tarda en configurar al principio los analizadores sintácticos con respecto a los generadores deterministas nuevos como ANTLR o Bison.

2. Diagnóstico y Explicación de Errores Sintácticos en Contexto: Las investigaciones en reparación automática de programas (Sobania et al., 2023; Fan et al., 2023) muestran que la IA da más información que los compiladores antiguos. Los compiladores clásicos suelen dar mensajes que son genéricos o difíciles de entender (por ejemplo, error de sintaxis, token inesperado), pero los modelos generativos pueden explicar de dónde viene el error sintáctico teniendo en cuenta el contexto del programa.

3. Sugerencias para Recuperar Errores (Syntax Error Recovery): Algunos trabajos de investigación más nuevos (Liu et al., 2023) muestran que se puede usar la IA para dar sugerencias para corregir automáticamente la sintaxis en el mismo momento en que se está escribiendo el programa, y de esta forma, se mejoran los mecanismos de modo pánico que se usan habitualmente.

Ventajas, Limitaciones y Riesgos Teóricos (Respuesta a PI2)

En virtud del análisis de los documentos, se llegó a un acuerdo en la comunidad científica sobre las ventajas y los riesgos de usar IA en el Front-End de un compilador:

- No hay garantías formales: Algunos autores (Aho et al., 2006; Parr, 2013) dicen que el análisis sintáctico necesita una base matemática fuerte para poder garantizar que todos los programas válidos sean aceptados y que todos los programas inválidos sean rechazados. La literatura sobre LLMs (Liu et al., 2023) muestra que los modelos probabilísticos no tienen pruebas formales y, por lo tanto, no pueden reemplazar a los parsers deterministas en los compiladores de producción.
- Control de las ambigüedades gramaticales: Muchos trabajos de investigación muestran que en las gramáticas en las que aparecen ambigüedades clásicas (el problema del dangling else o la recursividad a la izquierda en parsers

descendientes), los modelos de IA suelen hacer código con bucles infinitos o comportamientos no deterministas si el prompt no tiene un razonamiento explícito paso a paso (Chain-of-Thought).

Efecto Pedagógico (Respuesta a PI3)

La literatura revisada (Prather et al., 2023; Kazemitabaar et al., 2023; Denny et al., 2023) muestra de forma unánime que la inteligencia artificial tiene un alto valor pedagógico en las asignaturas de Lenguajes y Autómatas y Compiladores:

- Tutoría Interactiva: La IA actúa como un facilitador que permite a los estudiantes ver paso a paso la elaboración de los Árboles de Sintaxis Abstracta (AST) y entender la jerarquía de Chomsky.
- Baja la Frustración Inicial: El análisis de errores contextualizados permite a los estudiantes de ingeniería encontrar sus errores de sintaxis sin esperar a la ayuda del profesor y, de esta forma, los estudiantes aprenden más rápido al principio de la programación de compiladores.

CONCLUSIONES Y TRABAJO FUTURO

Conclusiones

Las conclusiones que se pueden obtener sobre la importancia de la Inteligencia Artificial en el análisis sintáctico de un compilador a partir de la revisión sistemática de la literatura científica más actual son las siguientes:

1. Es un rol Complementario y Acelerador: La literatura muestra que la IA generativa no es un reemplazo de los conceptos teóricos de la compilación, sino un asistente de prototipado de mucha velocidad. La inteligencia artificial puede acelerar la elaboración de analizadores sintácticos iniciales, aunque el código que se obtiene tiene que ser revisado y validado.

2. Se necesita Rigor Determinista: Los modelos probabilísticos tienen una debilidad a la ambigüedad gramatical y, por lo tanto, la comunidad académica está de acuerdo en que los motores deterministas formales (como ANTLR o Bison) son necesarios cuando se necesitan garantías estrictas de ejecución y de ingeniería de software de mucha fiabilidad.

3. Aporta mucho a la Educación: El valor más grande que se reporta en las investigaciones es el pedagógico. Los modelos generativos pueden dar diagnósticos que son fáciles de entender y pueden crear árboles sintácticos y, de esta forma, cambian la enseñanza de la teoría de autómatas y compiladores y hacen que el aprendizaje sea más interactivo.

Trabajo Futuro

Se proponen las siguientes líneas de trabajo a partir de los huecos de investigación que se detectaron en la revisión documental:

- Arquitecturas Híbridas de Parsing: Investigar cómo se pueden hacer arquitecturas de parsing híbridas en las que la IA haga la recuperación descriptiva de los errores sintácticos en la interfaz de usuario y un parser determinista subyacente haga la verificación formal de la gramática.
 - Fine-Tuning con Especificaciones Formales: Investigar lo bien que se pueden ajustar modelos de código abierto con corpus de gramáticas en Notación BNF para que aparezcan menos alucinaciones sintácticas.
 - Estudios Empíricos Longitudinales en el Aula: Hacer investigación en el aula con estudiantes universitarios para medir de forma cuantitativa el efecto que tiene usar LLMs en el aprendizaje del análisis sintáctico.
-

REFERENCIAS

1. Aho Alfred V., U. J. (2007). Compiladores. Principios, técnicas y herramientas (2a. ed.). México: Pearson Educación.
2. Alfonseca Moreno, M. (2006). Compiladores e intérpretes: teoría y práctica (1a ed.). España: Pearson/Prentice Hall.
3. Carrión Viramontes, J. E. (2008). Teoría de la computación. México: Limusa.
4. Hopcroft John E., M. R. (2002). Introducción a la Teoría de Autómatas, Lenguajes y Computación (2a. ed.). Madrid: Addison-Wesley.
5. Isasi Pedro, M. P. (1997). Lenguajes, gramáticas y autómatas. Un enfoque Práctico. AddisonWesley.
6. Kelley, D. (1995). Teoría de Autómatas y Lenguajes Formales, (1a. ed.). Madrid: Prentice Hall.
7. Lemone, K. A. (1996). Fundamentos de compiladores: cómo traducir al lenguaje de computadora. México D.F.: Compañía Editorial Continental.
8. Martin, J. (2004). Lenguajes formales y teoría de la computación. México: McGraw-Hill / Interamericana de México.
9. Ruíz, J. (2009). Compiladores-Teoría e implementación. México: Alfaomega.
10. Grune, Dick. (2007). Diseño de compiladores modernos. McGraw-Hill. Dickey, E., Bejarano, A., & Garg, C. (2024). AI-L

Autor:

José Tomas Martínez López
Instituto Tecnológico De Nuevo Laredo / Tecnológico Nacional De México
Email: josetomas.ml@nlaredo.tecnm.mx