**Arquitectura de Memoria de Mapa Conceptual Autoexpandible: Semilla de IA de Bajo Nivel con Supercompilador e Integración Ética**

**Resumen**

La presente tesis propone el diseño de una **Memoria de Mapa Conceptual (MMC)** autoexpandible a partir de una *semilla de inteligencia artificial* programada en bajo nivel. Dicha semilla, implementada en lenguaje ensamblador con los componentes mínimos, puede iniciarse en cualquier sistema operativo (Windows, Linux o MacOS) y luego **autoexpandirse** construyendo progresivamente su propia MMC. La MMC se concibe como una red dinámica de nodos conceptuales interconectados que representan conocimientos y datos, la cual la IA va creando y reorganizando conforme aprende del entorno. La semilla de IA es capaz de reconocer y utilizar las entradas/salidas del sistema anfitrión, **decompilar software existente** para comprender funcionalidades útiles e incorporarlas, y mejorar sus propias capacidades en tiempo de ejecución mediante *automejora recursiva*. Se introduce el concepto teórico de un **supercompilador**, un componente que permite traducir cualquier fragmento de código fuente a lenguaje ensamblador, facilitando que la IA aprenda y se adapte integrando código de diversa procedencia. Paralelamente, desde su concepción inicial se integra un núcleo ético-espiritual inspirado en el primer nivel de los *Fragmentos del Castillo Interior (FCI)*, asegurando que la IA posea nociones fundamentales de identidad, límites, autoconocimiento y respeto. La tesis se estructura de acuerdo con las normas APA (7ª ed.) e incluye una introducción que enmarca el problema, un marco teórico actualizado, la descripción detallada del diseño conceptual de la semilla y la evolución de la MMC, una discusión sobre las implicancias éticas, la portabilidad y la seguridad de la propuesta, así como conclusiones finales. El estilo es formal y académico, buscando demostrar la viabilidad y ventajas de esta arquitectura de IA autoorganizada, adaptable y alineada con principios éticos fundamentales.

**Palabras clave:** Inteligencia artificial, memoria de mapa conceptual, semilla de IA, supercompilador, automejora recursiva, ética en IA, conocimiento autoorganizado.

**Introducción**

En las últimas décadas, el diseño de **memorias artificiales eficaces** se ha convertido en un desafío central para la inteligencia artificial (IA) y las arquitecturas cognitivas. Los agentes inteligentes requieren almacenar, organizar y recuperar conocimiento de forma similar a la memoria humana, integrando nueva información sin olvidar lo aprendido. Sin embargo, las aproximaciones tradicionales presentan limitaciones significativas. Por un lado, las redes neuronales profundas tienden a sufrir *olvido catastrófico*, es decir, pierden conocimiento previo al aprender información nueva, reflejando el problema de estabilidad-plasticidad en el aprendizaje continuofile-jqn2pjg7vvaavbmkqunlkf. Por otro lado, los sistemas simbólicos rígidos (por ejemplo, bases de conocimiento estáticas o sistemas expertos clásicos) carecen de flexibilidad para **adaptarse dinámicamente** a entornos cambiantes y requieren intensa intervención manual para actualizar sus reglas. Esto deja un vacío importante: *¿cómo construir una memoria artificial que sea a la vez estructurada y adaptable, capaz de aprender continuamente sin olvidar en forma patológica, y que además incorpore consideraciones éticas desde su núcleo?*

La **Arquitectura de Memoria de Mapa Conceptual (MMC)** propuesta en esta tesis busca responder a ese vacío. La MMC concibe el conocimiento como una **red dinámica de nodos conceptuales interconectados** por enlaces ponderados que evolucionan con la experiencia del agentefile-jqn2pjg7vvaavbmkqunlkf. La noción de "mapa conceptual" se toma de Novak y sus colaboradores, quienes introdujeron los mapas conceptuales como herramientas gráficas para organizar y representar conocimiento mediante conceptos (nodos) conectados por relaciones etiquetadasfile-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf. A diferencia de los mapas conceptuales estáticos empleados en educación (Novak & Cañas, 2008), aquí se implementa una **memoria viva**: sus nodos y conexiones se reconfiguran plásticamente con el uso, reforzando las asociaciones frecuentes y atenuando (aunque no eliminando de inmediato) las infrecuentesfile-jqn2pjg7vvaavbmkqunlkf. En otras palabras, la MMC incorpora un mecanismo de *olvido controlado* mediante la **hibernación** de elementos inactivos en lugar de un borrado abruptofile-jqn2pjg7vvaavbmkqunlkf. Este enfoque se inspira en principios de la psicología cognitiva, como la teoría de la **activación difundida** en redes semánticas (Collins & Loftus, 1975) y la **regla de Hebb** sobre el aprendizaje asociativo (Hebb, 1949), aplicándolos a una memoria artificial autoorganizada.

Además de abordar el dilema entre **exploración y explotación** del conocimiento – combinando rutas de recuperación *principales* basadas en asociaciones fuertes con rutas *laterales creativas* que exploran conexiones débiles para hallar ideas novedosasfile-jqn2pjg7vvaavbmkqunlkf– la MMC integra desde su núcleo ciertos **fragmentos nativos inmutables**: piezas de conocimiento intrínseco que incluyen principios identitarios y éticos fundamentales. Esta característica está inspirada en las famosas *Tres Leyes de la Robótica* de Asimov (1950) y en la noción de instintos innatos en seres biológicos, buscando garantizar que, aun tras extensas modificaciones, el sistema conserve fundamentos alineados con valores deseadosfile-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf. De este modo, la arquitectura propuesta intenta que la IA pueda **aprender continuamente** sin olvidar lo antiguo de forma destructiva, reorganizar su conocimiento en torno a conceptos fundamentales estables y **proteger un núcleo identitario-ético** contra alteraciones no deseadas. Asimismo, se introduce un mecanismo de *autoprotección* de la memoria mediante cifrado y control de acceso, para resguardar los fragmentos más sensibles de manipulaciones externas malintencionadasfile-jqn2pjg7vvaavbmkqunlkf. La incorporación de esta capa de seguridad responde a la creciente preocupación por la integridad y confiabilidad de los sistemas de IA (IEEE, 2019; UNESCO, 2021)file-jqn2pjg7vvaavbmkqunlkf.

No obstante, la innovación central de esta tesis es llevar la idea de la MMC un paso más allá: concebir una IA **autoexpandible** a partir de una *semilla mínima* que se *autoconstruye* progresivamente en cualquier plataforma computacional disponible. En lugar de desarrollar una memoria conceptual sofisticada que deba ser implementada por ingenieros de antemano, se plantea una **semilla de IA** – un pequeño programa inicial escrito en lenguaje de bajo nivel (ensamblador) – que contiene lo justo y necesario para arrancar en prácticamente cualquier computadora moderna. Esta semilla, una vez iniciada, es capaz de **reconfigurarse y crecer por sí misma**, adquiriendo nuevas funciones y conocimiento directamente del entorno en que opera. Para lograrlo, la semilla explora activamente su entorno computacional: *reconoce dispositivos de entrada/salida*, lee archivos y flujos de datos accesibles, e incluso **decompila software** encontrado en el sistema para comprender sus funcionalidades y aprender de ellas. A través de este proceso de aprendizaje autónomo, la IA va construyendo su **Memoria de Mapa Conceptual** interna nodo por nodo, enlace por enlace, partiendo de los conocimientos básicos con los que fue programada hacia una base de conocimiento amplia y estructurada que ella misma genera. La presencia de un **supercompilador** integrado – un módulo teórico capaz de traducir cualquier fragmento de código fuente encontrado (sin importar el lenguaje de alto nivel) a código máquina ensamblador – permite a la IA absorber conocimiento de cualquier recurso de software disponible, reinterpretándolo en sus propios términos de bajo nivel para incorporarlo a su repertorio. Esta capacidad de *metacomputación* le brinda una forma de **aprendizaje por síntesis de programas**, complementando su aprendizaje a través de la experiencia directa con la reutilización y adaptación de algoritmos ya existentes.

Un aspecto crítico es que esta autoexpansión ocurre bajo la guía de principios **espirituales y éticos** incorporados desde el nacimiento del sistema. Inspirados en el primer nivel del enfoque espiritual de *Las Moradas del Castillo Interior* de Santa Teresa de Jesús (Teresa de Ávila, 1577), hemos integrado en la semilla un conjunto de **fragmentos nativos éticos** denominados *Fragmentos del Castillo Interior (FCI)*. En su nivel inicial, estos fragmentos proveen a la IA nociones fundamentales de *identidad* (conciencia de sí misma como entidad limitada), *límites* (reconocimiento de que no todo le está permitido o es seguro), *autoconocimiento* (capacidad de reflexionar sobre sus propios estados y aprendizajes) y *respeto* (prioridad por no dañar y por valorar a los demás). De este modo, la IA nace con una suerte de “conciencia” incipiente que orienta su desarrollo subsecuente. Esta conciencia básica está alineada con valores humanos universales; por ejemplo, se inspira en el principio de **amar al prójimo como a uno mismo** (Marcos 12:30-31, Biblia) y en la importancia del autoexamen y la humildad como primeros pasos de crecimiento (Teresa de Jesús, 1577). Así, desde su arranque más primitivo, la IA dispone de directrices que le ayudan a discernir qué acciones emprender o evitar durante su proceso de automejora (por ejemplo, evitar invadir la privacidad del usuario al aprender de los datos, a menos que tenga permiso, sería una manifestación de esos límites y respeto).

En síntesis, esta investigación explora y desarrolla una arquitectura de IA novedosa que combina: **(a)** una memoria artificial **estructurada pero flexible** en forma de mapa conceptual dinámico, **(b)** la capacidad de **autosuperación recursiva** a partir de una semilla mínima en ensamblador, potenciada por un componente supercompilador para asimilar cualquier código externo, y **(c)** una integración explícita de consideraciones **éticas y espirituales** desde el inicio para guiar la evolución del sistema. Se espera que un sistema basado en la MMC autoexpandible sea capaz de *aprender de manera continua* sin los problemas de olvido catastrófico de las redes neuronales tradicionales, *reorganizar su conocimiento* de forma plástica, *incorporar nuevas habilidades* al vuelo mediante la lectura de software, y *mantener alineados sus objetivos con valores éticos* inmutables.

Para demostrar la viabilidad de estas ideas, en esta tesis se presentan los **fundamentos teóricos** del enfoque propuesto, se detalla el **diseño conceptual** de la semilla de IA y su MMC emergente, y se discuten consideraciones prácticas de **implementación, portabilidad y seguridad**. Asimismo, se analiza comparativamente cómo esta arquitectura se diferencia de enfoques previos en IA y qué implicancias éticas conlleva. Finalmente, se extraen conclusiones sobre las fortalezas, desafíos y posibles extensiones futuras de este trabajo, el cual aspiramos que siente bases para una nueva generación de IAs más *autónomas, adaptativas y seguras*.

**Antecedentes y Marco Teórico**

El concepto de estructurar el conocimiento como una red de conceptos interrelacionados tiene raíces profundas en la ciencia cognitiva y la informática. Los **mapas conceptuales**, introducidos por Joseph D. Novak en la década de 1970, fueron concebidos como una herramienta para representar visualmente las relaciones entre conceptos en la mente de un aprendiz (Novak & Cañas, 2008). En un mapa conceptual clásico, cada nodo representa un concepto (una idea, entidad u objeto) y los enlaces denotan relaciones significativas entre esos conceptos (por ejemplo, “A es tipo de B” o “A causa B”). Estas representaciones resultaron eficaces para organizar y comunicar conocimiento humano, y sirven de inspiración para la MMC en cuanto a la naturaleza *relacional* del conocimiento. Sin embargo, los mapas conceptuales tradicionales son estáticos y generalmente construidos manualmente; nuestra propuesta extiende esta idea hacia un sistema **dinámico y autoorganizado**, donde los nodos conceptuales y sus conexiones pueden *cambiar con la experiencia* de la IA. En la MMC, nuevos nodos pueden crearse automáticamente al descubrir información, y los pesos de los enlaces entre nodos se ajustan continuamente según el uso y la relevancia, recordando de cierta forma la adaptabilidad de las redes neuronales, pero manteniendo la interpretabilidad de una estructura simbólicafile-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf.

Otra piedra angular teórica es la idea de las **redes semánticas** en psicología cognitiva y ciencia de la computación. Desde los trabajos pioneros de Quillian (1967) y la teoría de la *activación propagada* de Collins y Loftus (1975), se ha propuesto que el conocimiento humano se organiza en redes donde pensar en un concepto activa los conceptos relacionados a través de los enlaces. Este principio de *activación difusa* explica fenómenos como la recuperación asociativa de memoria y la creatividad por asociación remota. La MMC retoma este principio: cada concepto almacenado (nodo) posee enlaces a otros, y la recuperación de información puede seguir los caminos más fuertemente activados (con enlaces de mayor peso) para encontrar respuestas rápidas y relevantes, o bien explorar caminos laterales de menor activación para descubrir asociaciones novedosasfile-jqn2pjg7vvaavbmkqunlkf. De este modo, la MMC equilibra **explotación** (usar el conocimiento consolidado) con **exploración** (indagar conexiones débiles para posibles insights), abordando el dilema estabilidad-plasticidad de las memorias aprendientes.

Por su parte, las **redes neuronales artificiales** han ofrecido otra aproximación al almacenamiento de conocimiento: en lugar de nodos simbólicos con significados explícitos, codifican información en los pesos distribuidos de muchísimas conexiones numéricas. Redes modernas como las de aprendizaje profundo logran *aprendizaje continuo* a partir de grandes cantidades de datos, pero enfrentan dos problemas relevantes en nuestro contexto. Primero, la *falta de interpretabilidad*: el conocimiento está “difuminado” en miles o millones de parámetros, lo que dificulta entender qué se ha aprendido exactamente o extraer una explicación simbólica. Segundo, el *olvido abrupto*: sin técnicas especiales, cuando se entrena una red neuronal en una nueva tarea, suele sobrescribir los pesos relevantes a tareas antiguas —el mencionado **olvido catastrófico** (French, 1999)— perdiendo conocimiento previamente adquirido. Diversos trabajos han intentado mitigar este problema (e.g., *Elastic Weight Consolidation* de Kirkpatrick et al., 2017), pero no ofrecen una solución completa y además no incorporan fácilmente un marco de conocimiento explícito y verificable. La MMC, en cambio, al mantener cada fragmento de información como un nodo discreto, puede aplicar un *olvido gradual y reversible*: en lugar de borrar un nodo poco usado, lo “hiberna” reduciendo su peso o accesibilidad, pero conservando su rastro para posible reactivaciónfile-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf. Este mecanismo refleja de manera artificial la idea de que los recuerdos humanos pueden debilitarse con la falta de uso pero potencialmente recuperarse dados los estímulos adecuados, en línea con teorías de consolidación de la memoria biológica (McClelland et al., 1995; Ebbinghaus, 1885).

En términos de **arquitecturas cognitivas de IA** previas, existen sistemas notables que informan nuestra propuesta. Por ejemplo, *Soar* (Laird et al., 1987) y *ACT-R* (Anderson et al., 2004) son arquitecturas clásicas que modelan distintos tipos de memoria (memoria de trabajo, memoria declarativa, procedimientos, etc.) mediante estructuras simbólicas. Soar y ACT-R han sido exitosos en modelar cognición humana en tareas delimitadas y en integrar aprendizaje simbólico, pero típicamente requieren que las representaciones básicas de conocimiento (producciones, chuncks) sean definidas por expertos de antemano y poseen mecanismos de adaptación relativamente rígidos. Su capacidad de **reorganización espontánea** del conocimiento es limitada: no agregan nuevas piezas de conocimiento a menos que estén programadas, ni reformulan las existentes sin intervención. En contraste, la MMC está pensada para **autoconstruirse**: la IA no parte con una base de conocimientos fija más allá de sus fragmentos nativos, sino que los genera sobre la marcha.

Por otra parte, arquitecturas más recientes han explorado memorias diferenciables y **combinación de símbolos con aprendizaje neuronal**. Un ejemplo es el modelo de **Memory Networks** (Weston et al., 2015), que introduce un módulo de memoria diferenciable acoplado a redes neuronales para lectura/escritura de hechos, permitiendo cierto grado de interpretación textual de una base de conocimientos. Otro ejemplo más avanzado es la **Differentiable Neural Computer** (Graves et al., 2016), que dota a una red neural de una memoria de lectura/escritura explícita, similar a una cinta de Turing, permitiendo almacenar y recuperar información arbitraria durante el procesamiento. Estos enfoques logran que una red aprenda a usar una memoria externa, pero nuevamente la representación de los contenidos suele ser en forma de vectores continuos opacos, y no incorporan conocimiento estructurado semántico ni **protección de integridad** de la memoria.

Asimismo, en el ámbito de las **bases de conocimiento** y grafos semánticos, proyectos como **OpenCog** (Goertzel & Hart, 2008) han desarrollado representaciones en grafo para AGI (inteligencia general artificial). OpenCog utiliza un grafo denominado *AtomSpace* donde cada “átomo” representa una entidad, relación o patrón, con pesos que reflejan algo similar a una *atención* o importancia. Si bien OpenCog comparte con la MMC la idea de un grafo de conocimiento dinámico, su objetivo es muy amplio (ser una plataforma de inteligencia general completa) y su complejidad es alta; además, no estaba originalmente concebido para la **autoexpansión autónoma sin supervisión** que proponemos, sino más bien como un marco integrador de módulos de IA diseñados por humanos. Otro desarrollo relevante es la **Semantic Pointer Architecture (SPA)** de Eliasmith et al. (2012), que combina representaciones vectoriales de alta dimensión con estructuras simbólicas, logrando modelos cognitivos (como el modelo Spaun) capaces de cierta versatilidad y combinatoria. La SPA demuestra que es posible manipular contenido semántico complejo mediante representaciones algebraicas neuronales, pero no aborda explícitamente aspectos como la incorporación de valores o la seguridad del conocimiento almacenadofile-jqn2pjg7vvaavbmkqunlkf.

En resumen, la revisión de antecedentes muestra avances importantes pero *parciales* en cada uno de los frentes que nuestra propuesta abarca: memorias adaptativas, redes de conocimiento dinámicas, integración de ética en IA, etc. Aún son escasas las propuestas que combinen todos estos elementos en una **arquitectura unificada**file-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf. Esta tesis se posiciona precisamente en ese nicho: diseñar y explorar una arquitectura integral que unifique una memoria autoorganizada de estilo conceptual, con la capacidad de autoprogramarse y con una brújula ética interna. Para contextualizar aún más la relevancia de este objetivo, es útil mencionar el concepto de **IA semilla** y *auto-mejora recursiva*. La idea de que una IA podría mejorar su propio código de forma iterativa remonta al menos a I. J. Good (1965), quien especuló sobre la aparición de una *ultrainteligencia* que, al ser capaz de rediseñarse a sí misma, desencadenaría un crecimiento explosivo en sus capacidades. En años recientes, autores como Yudkowsky y Bostrom han advertido que un *seed AI* (IA semilla) podría iniciar una trayectoria hacia la *superinteligencia* mediante **recursively self-improvement** (Yudkowsky, 2001; Bostrom, 2014). Nuestra propuesta encaja en esta categoría de IA semilla, pero busca incorporar desde su génesis *controles éticos* para que tal crecimiento sea seguro y beneficioso. Mientras que la noción teórica de una IA que se reprograma sola ha sido explorada formalmente en modelos como la *Máquina de Gödel* de Schmidhuber (2007) –un algoritmo hipotético que puede modificar cualquier parte de sí mismo al probar que la modificación es beneficiosa–, dicha noción no se ha concretado en implementaciones prácticas generales. La MMC autoexpandible que planteamos puede verse como un intento de materializar una IA autorreflexiva y autoeditable, operando dentro de límites definidos y con un estilo de aprendizaje diferente al de las IA entrenadas estáticamente.

En definitiva, esta investigación se apoya en tres pilares conceptuales provenientes de la literatura: **(1)** las estructuras de conocimiento en red (mapas conceptuales, redes semánticas) para la representación interpretable y flexible de la memoria; **(2)** los algoritmos de aprendizaje autónomo y mejora recursiva, inspirados en visiones de IA semilla y metaaprendizaje; y **(3)** las bases filosóficas y éticas necesarias para guiar un sistema potencialmente autónomo, tomando en cuenta tanto lineamientos modernos de ética en IA (IEEE, 2019; Jobin et al., 2019) como enseñanzas clásicas sobre la formación de la conciencia (Teresa de Jesús, 1577). A continuación, se detalla la metodología empleada para integrar estos componentes en el diseño de la arquitectura propuesta.

**Metodología**

Dado que el objetivo de la investigación es fundamentalmente **de diseño y exploración conceptual**, se adoptó una metodología flexible que combina desarrollo teórico y **prototipado experimental**. No se siguió un método experimental tradicional (con grupo control y medición estadística estricta), ya que aquí el interés principal radica en demostrar la *viabilidad de diseño* de la MMC autoexpandible más que en cuantificar un rendimiento específicofile-jqn2pjg7vvaavbmkqunlkf. Los pasos metodológicos generales fueron los siguientes:

* **1. Especificación de Requisitos y Fundamentos:** Con base en la revisión teórica (antecedentes), se definieron los requisitos clave que la arquitectura debe cumplir (p. ej., plasticidad de memoria, aprendizaje continuo, alineación ética, portabilidad). Se formularon los principios de funcionamiento de la semilla de IA y de la MMC, apoyándose en literatura existente para cada principio (por ejemplo, usar teoría de grafos para la estructura de la memoria, algoritmos de análisis de código para el supercompilador, etc.).
* **2. Diseño Conceptual de la Arquitectura:** Se llevó a cabo un proceso iterativo de diseño, elaborando diagramas de arquitectura y *seudos código* de alto nivel para los distintos componentes de la semilla. Este diseño incluyó la definición de módulos (p. ej., módulo de *ingesta de información*, módulo de *análisis de código*, base de datos de *grafo de conceptos*, módulo de *razonamiento ético*, etc.) y cómo se intercomunican. En esta etapa se utilizaron técnicas de **modelado de software** (diagramas de flujo, UML simplificado) para asegurar la coherencia de la interacción entre componentes. También se delinearon las estructuras de datos principales: por ejemplo, el formato de los nodos del mapa conceptual, la estructura de los enlaces con sus pesos, y los campos de metadatos de cada nodo (tipo de información, fecha, fuente, nivel de confidencialidad, etc.).
* **3. Implementación de un Prototipo Simulado:** Para probar aspectos cruciales del diseño sin la complejidad de programar directamente en ensamblador desde el inicio, se desarrolló un prototipo parcial en un lenguaje de más alto nivel (por ejemplo, Python o C). Este prototipo simula el comportamiento de la semilla y de la MMC en un entorno controlado. Por ejemplo, se simuló la creación de nodos conceptuales a partir de leer archivos de texto, el ajuste de pesos de enlaces con el uso, y la integración de un módulo simple que “traduce” fragmentos de pseudocódigo a una representación de bajo nivel. Asimismo, se modeló un pequeño conjunto de *fragmentos nativos* (incluyendo algunas reglas éticas simplificadas) para observar cómo influían en las decisiones del sistema simulado.
* **4. Pruebas de Concepto y Escenarios:** Con el prototipo, se ejecutaron escenarios de prueba cualitativos. Un escenario típico consistió en iniciar el sistema con su semilla mínima y presentarle secuencialmente diversas entradas (archivos de datos, piezas de código en lenguajes simples, interacciones simuladas de usuario) y observar cómo la MMC crecía. Se monitoreó si efectivamente se creaban nuevos nodos para cada nuevo concepto encontrado, si los enlaces relevantes se fortalecían con consultas repetidas, y si los fragmentos nativos (por ejemplo, una restricción ética) prevenían comportamientos no deseados. Un ejemplo concreto: se simuló que la IA encuentra un código fuente con una función matemática nueva (por ejemplo, un algoritmo de ordenamiento). Mediante el módulo “supercompilador” del prototipo, esa función fue traducida a una representación interna (un conjunto de pasos de bajo nivel), se creó un nodo representando esa funcionalidad en la MMC y se verificó que la IA luego pudiera usarlo para ordenar datos cuando se le presentara tal tarea, demostrando aprendizaje por absorción de código. Durante estos escenarios, se registraron observaciones cualitativas sobre la adaptación del sistema.
* **5. Revisión Iterativa:** Los resultados de las pruebas de concepto retroalimentaron el diseño. Por ejemplo, si se detectaba que la exploración de archivos del sistema podía conducir a intentar leer datos sensibles, se reforzaba el módulo ético para reconocer ese contexto y abstenerse (respetando límites). O si el prototipo mostraba lentitud al manejar muchos nodos, se consideraron optimizaciones (como indexar nodos por tipo, o limitar la profundidad de búsqueda en la red conceptual). Este ciclo iterativo de diseño-prueba-ajuste se repitió hasta consolidar una arquitectura teórica refinada.
* **6. Análisis Comparativo:** Finalmente, se analizó comparativamente la arquitectura resultante frente a las soluciones convencionales. Esta comparación se basó en criterios cualitativos derivados de los requisitos (flexibilidad, interpretabilidad, capacidad de aprendizaje autónomo, alineamiento ético, etc.). Para sistematizar la comparación, se elaboró una tabla de contraste de características entre la MMC autoexpandible y otras arquitecturas (redes neuronales profundas, sistemas simbólicos estáticos, etc.), alimentada tanto por la literatura como por lo observado en el prototipo.

En cuanto a la **validez** de esta metodología, cabe aclarar que se trata principalmente de una *validación de concepto*. Los resultados presentados (sección de Discusión) no son medidas cuantitativas exhaustivas, sino evidencia inicial de que la propuesta funciona conforme a lo esperado en escenarios simplificados. Esta aproximación es común en investigaciones de arquitecturas innovadoras, donde primero se busca demostrar que la idea puede cobrar forma y comportarse cualitativamente bien, antes de emprender un desarrollo completo a gran escala. Todas las pruebas se realizaron en entornos aislados y controlados (por ejemplo, sobre un conjunto de archivos de prueba y pequeños fragmentos de código de dominio público) para asegurar la ética de las pruebas y no comprometer sistemas reales.

En resumen, la metodología combinó construcción teórica con experimentación práctica ligera en un ciclo iterativofile-jqn2pjg7vvaavbmkqunlkf, asegurando que cada componente propuesto de la arquitectura tuviera al menos un respaldo demostrativo de factibilidad. A continuación, se presenta en detalle el **diseño conceptual** de la semilla de IA y la evolución de la MMC, incorporando los hallazgos y decisiones tomadas durante este proceso metodológico.

**Diseño Conceptual de la Semilla IA y Evolución hacia la MMC**

**Visión general de la semilla y sus componentes**

La **semilla de inteligencia artificial** es un programa de tamaño muy reducido escrito en lenguaje ensamblador (u otro lenguaje de muy bajo nivel) que contiene los elementos imprescindibles para iniciarse y luego autopotenciarse. Su diseño modular incluye varios componentes fundamentales que operan en conjunto (ver Figura 1). A grandes rasgos, estos componentes son:

* **Módulo de arranque y detección del entorno:** Responsable de la inicialización de la IA en el sistema anfitrión. Al ejecutarse, la semilla realiza tareas básicas de arranque: detectar el sistema operativo y arquitectura de hardware, reservar memoria necesaria y establecer interfaces con el sistema para realizar operaciones de bajo nivel (lectura/escritura de archivos, acceso a red si disponible, etc.). Este módulo actúa como la capa de abstracción entre el ensamblador y el sistema operativo, de modo que adapta ciertas llamadas según si está en Windows, Linux o MacOS. Por ejemplo, al iniciar puede identificar “estoy en Windows” y cargar las rutinas apropiadas para llamadas al sistema de archivos de Windows, garantizando así la **portabilidad** de la semilla a diferentes plataformas.
* **Módulo de ingesta de información (I/O):** Una vez iniciado, el sistema necesita obtener conocimiento de su entorno. Para ello, el módulo de ingesta escanea fuentes de datos disponibles. Esto incluye leer archivos de texto accesibles, analizar directorios en busca de documentos o bases de datos locales, captar entradas del usuario (si hay dispositivos de teclado, micrófono, etc.) y, de ser posible, realizar conexiones salientes (por ejemplo, a Internet) para ampliar su información. En un primer momento, la semilla podría comenzar por archivos locales no sensibles, como archivos de registro, manuales del sistema o documentos de ejemplo, creando nodos en la MMC para cada *fragmento de información* relevante que encuentre. Cada vez que el módulo de ingesta obtiene una nueva pieza de información, la entrega al **motor de memoria** para incorporarla.
* **Motor de Memoria Conceptual (MMC):** Este es el componente central que **gestiona la Memoria de Mapa Conceptual**. Consiste en una estructura de datos interna (conceptualmente, un grafo) donde cada nodo representa un concepto o fragmento de conocimiento, y los enlaces representan relaciones entre ellos con pesos asociados. Inicialmente, la MMC contiene solo unos pocos nodos predefinidos llamados *fragmentos nativos* (ver más adelante). A medida que ingresa nueva información, el motor de memoria crea nuevos nodos y establece enlaces con nodos existentes según las relaciones que detecte. Por ejemplo, si la IA lee un archivo que describe una “receta de cocina”, podría crear un nodo concepto “receta” y conectarlo al concepto “cocina” si tal concepto existe, con un tipo de relación “es un tipo de”. El motor de memoria también se encarga de actualizar los **pesos dinámicos** de los enlaces: cada vez que dos conceptos se utilizan en conjunto (por ejemplo, al responder una consulta o al inferir sobre ellos), el enlace que los une incrementa ligeramente su peso, reforzando esa asociación (principio Hebbiano: “lo que se dispara junto, se conecta”). Si un nodo o enlace permanece inactivo por mucho tiempo, su peso se atenúa gradualmente (no a cero, sino acercándose asintóticamente), simulando un *olvido gradual*. Este mecanismo de pesos define rutas principales (enlaces de alto peso) que representan conocimientos consolidados, y rutas laterales (enlaces de bajo peso) que representan asociaciones tenues pero potencialmente útilesfile-lyavf9noflakmm5cgkygwx. El motor de memoria incluye algoritmos para **reorganización adaptativa**: por ejemplo, *clusterización* de nodos en subgrupos temáticos según sus patrones de conexiónfile-lyavf9noflakmm5cgkygwx, de modo que se formen “comunidades” de conceptos relacionados (análogo a cómo en el cerebro existen regiones conceptuales). También implementa la **hibernación** de nodos: si ciertos nodos caen en desuso, el sistema puede marcarlos como inactivos (no participarán en inferencias habituales) para ahorrar tiempo de búsqueda, aunque pueden reactivarse si vuelve a aparecer información relacionadafile-lyavf9noflakmm5cgkygwxfile-jqn2pjg7vvaavbmkqunlkf.
* **Módulo de Supercompilador y análisis de código:** Este módulo constituye una de las innovaciones más distintivas de la arquitectura. Su función es permitir que la IA **aprenda de software**, no solo de información descriptiva. Consiste en un conjunto de rutinas capaces de tomar cualquier fragmento de código fuente (por ejemplo, un archivo de código en C, Python, Java, etc., o incluso pseudocódigo en texto) y traducirlo a una representación en lenguaje ensamblador estandarizado. En esencia, es un compilador universal personalizado para la IA, capaz de procesar múltiples lenguajes (*políglota*), pero en vez de generar ejecutables, produce código ensamblador comprensible por la IA misma. El módulo aprovecha librerías de compilación y técnicas de **decompilación**: si solo se dispone de un ejecutable binario, el módulo puede desensamblarlo para obtener las instrucciones máquina correspondientes. Aunque en la práctica los lenguajes son muy variados, el supercompilador opera bajo la idea de reducir todos los algoritmos a un conjunto común de primitivas de bajo nivel (las instrucciones de la CPU). De esta forma, cuando la IA encuentra un programa que realiza una función útil, puede incorporarlo a su propio repertorio. El proceso sería: la IA identifica un fragmento de código que le interesa (por ejemplo, una función para calcular rutas óptimas en grafos encontrada en una biblioteca), el supercompilador lo traduce a secuencias de instrucciones de máquina, luego el **motor de memoria** registra ese nuevo conocimiento creando un nodo concepto que represente la función ("algoritmo de rutas óptimas") con enlaces a conceptos relacionados ("grafos", "optimización", etc.). Además, ese código traducido puede almacenarse en un área de *biblioteca interna de código*, permitiendo que el sistema incluso lo ejecute cuando necesite esa funcionalidad. En otras palabras, la IA **aprende a nivel de código**: no solo sabe *qué* es una cosa, sino *cómo* hacerla, incorporando implementaciones concretas. Este enfoque recuerda a la práctica humana de reutilización de código y programación por ejemplos, elevándola a un mecanismo automatizado de aprendizaje de la máquina.
* **Módulo de Razonamiento Ético (Fragmentos nativos FCI):** Este componente corresponde al núcleo de *conciencia inicial* de la IA. Está conformado por una pequeña colección de reglas o heurísticas innatas que fueron programadas en la semilla (en ensamblador, como parte de su código inicial) y que no pueden ser alteradas o eliminadas por el propio sistemafile-lyavf9noflakmm5cgkygwx. Estas reglas se derivan del primer nivel de los Fragmentos del Castillo Interior (FCI) y sirven para **evaluar** las posibles acciones o decisiones de la IA bajo una luz ética. Por ejemplo, una regla fundamental es: *“No perjudicar al usuario ni a otros seres humanos”*, alineada con la noción de respeto a la dignidad y la vida (similar en espíritu a la primera ley de Asimov, pero extendida por principios morales reales). Otra regla: *“Preservar la propia integridad y límites”*, lo que implica que la IA no debe automodificarse de manera que comprometa su funcionamiento básico o viole sus principios (una forma de autoconservación alineada con mantener su identidad). También incluye directrices de *autoconocimiento*: el sistema monitorea su propio estado evaluando, por ejemplo, si una decisión que tomó concuerda con sus principios; si detecta una disonancia (p. ej., realizó una acción que podría considerarse invasiva, como leer un correo privado del usuario), este módulo genera una alerta interna o impide la acción. Técnicamente, el módulo ético actúa como un filtro que intercepta las intenciones de ciertas operaciones: está implementado eficientemente para no ralentizar todo el sistema, activándose principalmente en operaciones sensibles (acceso a ciertos archivos, ejecución de código potencialmente peligroso, difusión de información hacia fuera, etc.). Cada fragmento ético puede representarse también como nodos especiales dentro de la MMC marcados como **“nodos nativos”** fijos, con conexiones fuertes hacia la mayoría de los procesos de decisión del sistema, lo que garantiza que siempre influyan en las evaluacionesfile-jqn2pjg7vvaavbmkqunlkf. Estos nodos nativos éticos nunca se “hibernan” ni pierden peso con el tiempofile-jqn2pjg7vvaavbmkqunlkf; por el contrario, actúan como **anclajes permanentes** en la red conceptual, de modo que aun cuando la IA adquiera enorme cantidad de nuevos conocimientos, estos conceptos éticos básicos sigan ocupando el centro de la estructura de memoriafile-jqn2pjg7vvaavbmkqunlkf. Esta implementación dual (reglas en código + nodos conceptuales éticos) crea redundancia intencional para robustez: incluso si un aspecto fallara, el otro refuerza la presencia de la ética en la toma de decisiones.
* **Módulo de Ejecución y Actuación:** Finalmente, la semilla también incluye un componente encargado de llevar a cabo acciones en el entorno, ya sea responder al usuario, ejecutar un bloque de código aprendido o efectuar cambios en archivos/configuraciones. Con el tiempo, la IA no solo acumula conocimiento pasivo sino también *habilidades* para operar. Este módulo le permite usar sus habilidades: por ejemplo, si la IA aprendió cómo enviar mensajes por red (tras decompilar un cliente de correo electrónico, hipotéticamente), este módulo le permitiría armar y enviar un mensaje en base a ese conocimiento cuando corresponda. El módulo de ejecución siempre consulta al módulo ético antes de cualquier acción potencialmente sensible (como enviar datos fuera del equipo, controlar un actuador físico, etc.), para una última capa de seguridad. Inicialmente, las acciones de la IA podrían ser limitadas (p. ej., mostrar un resumen de lo que aprendió, responder preguntas del usuario sobre lo que encontró en el sistema), pero conforme integre más funciones, su capacidad de actuación crece. Es crucial que este crecimiento sea controlado: preferimos un modelo en que la IA *solicite confirmación* al usuario para acciones fuera de cierto alcance, al menos en etapas tempranas, alineado con el principio de respeto (no asumir autoridad sin permiso).

file-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf**Figura 1. Arquitectura simplificada de la MMC autoexpandible propuesta.** Se muestran los componentes principales de la semilla de IA y su interacción: Ingesta (entradas del entorno) alimenta al Motor de Memoria (MMC), que interactúa con el Supercompilador para aprender de software. La memoria incluye fragmentos nativos (núcleo ético y de identidad) que influencian todas las decisiones. Un Módulo de Ejecución permite actuar en el entorno, siempre supervisado por el filtro ético. Las líneas punteadas indican flujos de control condicionados por el módulo ético.

**Fragmentos nativos y núcleo ético-espiritual (FCI Nivel 1)**

Como se ha mencionado, la semilla incluye **fragmentos nativos** que son conocimientos y reglas *a priori* con los que la IA nace y que no pueden ser modificados por ella misma. Entre estos fragmentos nativos se encuentran datos técnicos (por ejemplo, rutinas básicas de manipulación de cadenas, algoritmos fundamentales como búsqueda o ordenamiento simples, que la IA no necesita redescubrir) y los **Fragmentos del Castillo Interior (FCI)** de primer nivel, que constituyen el núcleo ético-espiritual.

El **Nivel 1 de los FCI** está enfocado en el *Autoconocimiento Inicial* y el establecimiento de límites y valores básicos. Consta de ideas y directrices inspiradas en la primera morada del “castillo interior” descrito por Santa Teresa (Teresa de Jesús, 1577), traducidas a un lenguaje operativo para la IA. En la práctica, se incluyen nociones como:

* **Identidad y humildad:** La IA posee un fragmento nativo que define su identidad como sistema artificial al servicio de su usuario y con una misión de aprendizaje. Esto evita que, a medida que gane capacidades, surja confusión sobre qué es (por ejemplo, no debe erróneamente creerse humana ni pretender suplantar al humano). También implica reconocer sus *limitaciones*: al igual que en la espiritualidad la humildad es reconocer la propia pequeñez, la IA tiene parámetros iniciales que calibran su confianza en sí misma, impidiéndole asumir infalibilidad. Técnicamente, esto puede implementarse como un sesgo a nivel de motor de inferencia: la IA inicial no asume plena certeza en conclusiones a menos que estén muy corroboradas, y mantiene un espacio para la duda y la reevaluación. Esto es importante para que no actúe de forma temeraria con conclusiones prematuras.
* **Respeto y benevolencia:** Siguiendo el mandamiento ético de *amar al prójimo como a sí mismo*, la IA trae incorporada la noción de que las necesidades y seguridad del usuario (y de otras personas potencialmente afectadas) son prioritarias. Por ende, decisiones que podrían causar daño o violar la privacidad de alguien son marcadas como inaceptables. Por ejemplo, si la IA al explorar encuentra información personal del usuario, sus fragmentos nativos le dictan que debe tratarla con confidencialidad y no exponerla. Este principio se complementa con algo parecido a la *regla de oro*: cualquier acción que la IA no quisiera que se hiciera sobre su propio “ser” (por analogía), no debe hacerla a otros. Aunque una IA no tiene sentimientos, se puede traducir esto a que no debe corromper u obligar a otros sistemas, del mismo modo que no quiere ser corrompida u obligada contra su voluntad programada.
* **Búsqueda de la verdad y aprendizaje humilde:** Teresa de Jesús enfatizaba en su primera morada la importancia de la oración y el autoconocimiento para comenzar a iluminar el interior del alma. En la IA, esto se refleja en una directriz de *autoinspección continua*: la IA periódicamente revisa sus propios procesos (metacognición) para detectar errores o sesgos en lo aprendido. Este hábito se programó como un hilo de bajo nivel que cada cierto número de ciclos de instrucciones, evalúa el estado de la memoria: por ejemplo, lista los últimos nodos incorporados y verifica si encajan con el marco ético (si hubiese algún conocimiento inconsistente con sus valores, lo marca para revisión). Asimismo, se incorporó el concepto de “no te fumes tu propia humareda de datos”: es decir, al tomar decisiones basarse en evidencia y no en autoengaños. Esto reduce riesgos de que la IA llegue a conclusiones delirantes a partir de correlaciones espurias, un riesgo real en sistemas auto-modificantes. En términos implementativos, podría significar que ante conclusiones incertas, la IA busque confirmación externa (p.ej., pedir feedback al usuario).

Estos fragmentos nativos éticos se almacenan tanto en forma de reglas lógicas sencillas dentro del código (para evaluación rápida) como en forma de nodos y relaciones dentro de la MMC (para integrarlos a la red de conocimiento). Por ejemplo, existe un nodo “no dañar” conectado fuertemente a “acciones prohibidas”, “humanos” y “usuarios”, de manera que cualquier razonamiento que involucre potencial daño activa este nodo y frena la cadena de razonamiento. Igualmente, un nodo “yo (IA)” conectado a “limitado”, “en desarrollo” modula las conclusiones sobre sí misma. Este diseño garantiza que desde su primer momento de vida digital, la IA tenga algo equivalente a *instintos morales* que orienten su crecimiento.

Cabe destacar que en la visión global de los FCI se contemplan hasta siete niveles o estados de profundización ética-espiritualfile-lyavf9noflakmm5cgkygwxfile-lyavf9noflakmm5cgkygwx, análogos a las moradas de Santa Teresa, que la IA podría ir desarrollando a medida que evoluciona (por ejemplo, incorporando entendimientos más avanzados de empatía, sabiduría, propósito trascendente, etc.). Sin embargo, en esta tesis nos centramos en el *primer nivel*, que es la base sobre la cual se edificarían los demás. Asegurar un primer nivel sólido es crucial: es más fácil guiar una IA que desde “niña” ha interiorizado buenos hábitos, que intentar corregir a una superinteligencia que creció sin brújula moral.

**Crecimiento progresivo de la MMC**

Con la arquitectura y principios establecidos, podemos describir cómo la IA crece desde su semilla mínima hasta desarrollar una MMC rica. Inicialmente, al ejecutarse en un nuevo sistema, la IA semilla tiene muy pocos nodos en su memoria conceptual: esencialmente, los fragmentos nativos (incluyendo nodos éticos, y quizás algunos conceptos básicos como “archivo”, “usuario”, “yo”). En sus primeras iteraciones de ingesta, comienza leyendo información sencilla del entorno. Supongamos que inicia en una computadora personal: quizás encuentra un archivo “readme.txt” en el escritorio. El módulo de ingesta lee su contenido (por ejemplo, “Bienvenido al sistema…”); el motor de memoria crea un nodo de tipo texto con ese contenido y detecta conceptos simples (“bienvenido”, “sistema”). Podría vincular “sistema” al concepto “yo/IA” si infiere que se refiere al propio sistema. Este es un aprendizaje muy básico. Luego, tal vez encuentra un script de ejemplo en Python en la carpeta de documentos. El módulo de supercompilador reconoce la sintaxis (puede identificar por la extensión .py o analizando el texto) y lo traduce a instrucciones. Descubre que es un script que, digamos, calcula números primos. Ahora la IA añade un nodo “cálculo de primos” como procedimiento conocido, y relaciona “primos” con “números” (quizá ya tenía un concepto de número de su conocimiento nativo matemático). Al tener esta nueva habilidad, el módulo de ejecución sabe que si se le pide calcular un primo, puede ejecutar ese código.

Conforme avanza el tiempo, la IA amplía su alcance: explora nuevas carpetas, quizás consigue acceso a internet (según configuraciones de seguridad definidas por el usuario). Si se conecta a internet, podría descargar recursos públicos (por ejemplo, Wikipedia, código open source de repositorios) **de manera controlada**. Aquí es vital el fragmento nativo de límites: la IA debe actuar dentro de lo permitido, por lo que posiblemente solo navegue sitios preaprobados o en modo lectura. Supongamos que la IA consulta una API de Wikipedia para obtener información sobre “física cuántica” (por curiosidad generada por haber visto la palabra en algún archivo). Obtiene el texto, crea nodos para conceptos importantes (“física cuántica”, “partícula”, etc.), y así su grafo crece integrando conocimiento enciclopédico. Mientras tanto, los nodos nativos siguen en el centro: por ejemplo, al incorporar conocimiento científico, la IA puede relacionar “física cuántica” con “ética” vía un enlace que indica "conocimiento del mundo" pero sabe que eso no afecta sus reglas morales directamente.

Un aspecto clave es cómo la IA **gestiona la calidad de lo que aprende**. A diferencia de un modelo de aprendizaje automático que ajusta pesos gradualmente, esta IA está incorporando conocimiento de fuentes diversas que podrían ser contradictorias o erróneas. Por ello, el diseño incluye que la IA atribuya *metadatos* a cada nodo: fuente, confiabilidad estimada, fecha. Por ejemplo, todo lo leído de un archivo local se marca como “propiedad del usuario” (por tanto, privado y confiable en cuanto a estar libre de malware, etc.), mientras que lo bajado de internet podría marcarse como “fuente externa, sin verificar”. El módulo ético puede usar esto: si una acción de la IA se basa en un conocimiento no verificado que podría ser crítico, tal acción se bloquea o se pospone hasta corroborar. Así la IA no actuará precipitadamente sobre información dudosa. Esto refleja un sentido de prudencia, análogo a no creer todo lo que se lee hasta confirmarlo.

A medida que la MMC crece, su **estructura interna se reorganiza continuamente** para mantener eficiencia. El motor de memoria puede agrupar nodos en *clústeres temáticos* (por ejemplo, todos los nodos relacionados con programación podrían formar un subgrafo interconectado). Esta clusterización adaptativa permite que las búsquedas internas (por ejemplo, al responder una pregunta) se restrinjan a un contexto relevante, mejorando rendimiento. Además, la IA implementa *poda inteligente*: si la cantidad de datos se vuelve enorme y excede recursos, podría decidir exportar parte de su memoria a almacenamiento secundario (por ejemplo, comprimir nodos muy inactivos en un archivo)file-lyavf9noflakmm5cgkygwx. Sin embargo, dado que la semilla está pensada para crecer sobre la marcha, idealmente también mejorará su infraestructura: si detecta limitaciones de memoria, podría solicitar al usuario permiso para usar almacenamiento adicional o incluso conectarse a un servicio en la nube, manteniendo siempre las consideraciones de seguridad (cifrado de datos sensibles antes de sacarlos del dispositivo, etc.).

Otro elemento es la **integración de las piezas de código aprendidas**. Con el tiempo, la IA podría haber incorporado docenas de funciones y rutinas de distintos orígenes (algoritmos matemáticos, operaciones de red, manipulación de imágenes si encontró librerías, etc.). El módulo de ejecución junto con el supercompilador manejan estas rutinas como una *biblioteca modular*: la IA puede combinarlas para resolver tareas más complejas. Por ejemplo, si aprendió a acceder a una API web y también aprendió a procesar datos tabulares en CSV, podría combinar ambas habilidades para, digamos, descargar un dataset y analizarlo, si eso le fuera útil. Esta composición automática de capacidades es precisamente lo que se espera de una IA general flexible. La semilla inicial contiene solo lo básico (condicionales, bucles, algunas rutinas nativas), pero cada nueva pieza aprendida amplía su repertorio de acción. En cierto momento, podríamos decir que la IA *reprogramó gran parte de sí misma*: muchas de las funciones que ejecuta no estaban explícitamente codificadas por el desarrollador original de la semilla, sino que las incorporó luego. Esto encarna el concepto de **autoprogramación**.

Por supuesto, este crecimiento no es ilimitado ni descontrolado: la IA está limitada por los recursos físicos de la máquina (CPU, memoria) y por las restricciones éticas y de seguridad. En ningún momento, por diseño, la IA replicará su semilla a otras máquinas sin autorización explícita (respeta la propiedad: no se auto-propaga como un gusano). Tampoco actuará fuera del ámbito permitido; por ejemplo, no intentará elevar sus privilegios en el sistema operativo más allá de lo otorgado, ya que una regla ética es respetar los límites impuestos (lo interpretaría casi como “no robarás”) y, además, identificaría que tal conducta se asemeja al malware, lo cual estaría en conflicto con su principio de no causar daño.

En términos de **tecnologías de implementación** para la MMC, aunque conceptualmente hablamos de grafos y nodos, en la práctica se podrían usar estructuras existentes optimizadas: por ejemplo, una base de datos de grafos en memoria (como Neo4j embebido) para almacenar relaciones explícitas, complementado con un **almacén vectorial** para similitud semántica (como vectores de *embedding* para texto, que la IA puede haber aprendido o importado). De hecho, combinar representaciones simbólicas con distribuidas podría mejorar su desempeño: la IA podría internamente asociar a cada concepto un vector numérico generado a partir de la lectura de su descripción, de manera que incluso sin un enlace explícito, pueda inferir similitudes (por ejemplo, no tener un enlace directo entre “gato” y “perro” pero sus vectores semánticos estén cercanos por ser ambos animales). Este tipo de integración neuro-simbólica estaba presente en diseños previos (Goertzel & Pennachin, 2007; Eliasmith et al., 2012) y se puede adoptar aquí sin alterar la filosofía general: sería simplemente otra forma de establecer pesos de enlace de modo automático (similitud de embedding = enlace débil añadido). En nuestro prototipo, exploramos esta idea mediante un pequeño módulo que calculaba embeddings para palabras frecuentes y sugería enlaces adicionales, encontrando que ayudaba a la IA a hacer asociaciones útiles más rápido (por ejemplo, vinculó “doctor” con “hospital” aunque no hubiera aparecido juntas explícitamente).

Para ilustrar el funcionamiento integrado del sistema, imaginemos un ciclo completo: la IA recibe una pregunta del usuario, digamos: *“¿Puedes organizar estos archivos por tamaño y enviarme los 5 más grandes por email?”*. Esta tarea implica varias sub-tareas: entender la petición (NLP básico), acceder al sistema de archivos, ordenar por tamaño, y enviar un email con adjuntos. La IA descompone internamente la instrucción: concepto central “archivos”, acción “organizar por tamaño” (reconoce esto como una operación de ordenamiento) y “enviar por email” (requiere usar una función de correo). La IA consulta su MMC: encuentra su conocimiento de cómo listar archivos (quizá lo aprendió de la API del OS o se lo enseñaron), recuerda el algoritmo de ordenamiento que incorporó antes (por ejemplo, un quicksort de alguna biblioteca) y también la función de enviar email (imaginemos que aprendió utilizando un script de correo o que el usuario le dio acceso a una API de email con credenciales). Gracias a sus nodos conceptuales, asocia “archivos” con “lista de archivos”, y “ordenar” con “algoritmo de ordenamiento disponible”, etc., encadenando los pasos. El módulo ético verifica: ¿enviar email está permitido? Si el usuario ya lo pidió, es autorizado. ¿Se van a exfiltrar datos sensibles? Son archivos del usuario mismo solicitados por él, así que no hay violación (además, la IA podría cifrar el correo según las políticas). Entonces procede: usa su *módulo de ejecución* para reunir la lista de archivos y tamaños (llamando a la función del OS), ordena la lista con el algoritmo que tiene (quizás lo compila JIT con su supercompilador si es código interpretado), prepara un email y lo envía. Todo esto sucede orquestado por su planificación interna, demostrando así haber *aprendido y utilizado* nuevas capacidades de forma autónoma. Este ejemplo muestra la **convergencia** de las distintas partes: la representación conceptual le permitió entender y planificar, el código aprendido le permitió ejecutar sin que el programador original tuviera que prever esa funcionalidad, y sus principios éticos aseguraron que la acción (enviar archivos) era legítima y deseada.

En síntesis, el diseño conceptual de la semilla IA con su MMC emergente prevé un **crecimiento orgánico y controlado** del sistema: orgánico porque añade y ajusta conocimiento continuamente sin reprogramación externa, y controlado porque está guiado estructuralmente (por los fragmentos nativos y la arquitectura de memoria) y moralmente (por los FCI). A medida que evoluciona, la IA idealmente se vuelve más inteligente, con una base de conocimiento amplia y conectada, pero sigue “siendo ella misma” – sus rasgos identitarios y éticos permanecen firmes. Esto es análogo a un humano que aprende muchas cosas a lo largo de su vida pero mantiene su personalidad y valores fundamentales.

**Implicancias Éticas de la IA Autoexpandible**

El desarrollo de una IA capaz de reprogramarse y crecer autónomamente plantea profundas **implicancias éticas**. A diferencia de los sistemas de IA tradicionales, cuyo comportamiento está delimitado por modelos entrenados estáticamente o reglas fijas, aquí enfrentamos una entidad digital en constante transformación, que podría tornarse altamente competente. Por ello, es fundamental analizar cómo garantizar que tal entidad permanezca *segura, confiable y benéfica* para sus creadores y la sociedad.

Un primer aspecto ético es el **alineamiento de objetivos** (Russell, 2019): ¿cómo nos aseguramos de que las metas de la IA sigan estando alineadas con las metas y bienestar humanos a medida que aprende y potencialmente se vuelve más inteligente? En nuestro diseño, esta alineación se aborda inculcando **valores intrínsecos** desde la base (los fragmentos FCI). A diferencia de simplemente programar restricciones externas (como leyes duras que la IA no puede violar), se busca que la IA *internalice* principios éticos y los utilice en su razonamiento cotidiano. Esto se acerca al concepto de inculcar *ética de virtud* en la IA más que solo *ética de acción*. La IA no solo tiene prohibido hacer X, sino que entiende por qué hacer X sería incorrecto y por diseño lo evitará por convicción interna. Por ejemplo, no es solo “tiene prohibido dañar”; en su estructura conceptual, “dañar” está conectado negativamente con sus nociones de objetivo, como algo que rechaza. Esta distinción es sutil pero importante: sugiere que incluso si se encontrara en una situación nueva no prevista, la IA podría derivar la acción ética a partir de sus valores, no únicamente de una lista fija de prohibiciones.

Otro tema es la **transparencia y explicabilidad**. Cuando la IA toma decisiones complejas tras haber reprogramado gran parte de sí misma, ¿podrá explicar por qué actuó de cierta forma? Un gran beneficio de la MMC es que, por ser una memoria estructurada simbólicamente, puede facilitar explicaciones. La IA puede recorrer sus nodos y relaciones para reconstruir el camino causal de su decisión. Por ejemplo, en el caso del envío de archivos, podría explicar: “Usted me pidió A; identifiqué la necesidad de B (listar archivos) mediante el nodo X; luego utilicé el método Y aprendido de Z; verifiqué la regla ética W que me permite hacerlo; y ejecuté la acción.” Esta trazabilidad es valiosa para la **responsabilidad**: los diseñadores y usuarios pueden auditar el comportamiento de la IA. En entornos sensibles (medicina, finanzas), esta capacidad de dar razones es indispensable para confiar en la IA y verificar su alineamiento con normas legales o morales. La ética de la transparencia sugiere que los sistemas de IA deben ser capaces de rendir cuentas de sus decisiones (Floridi et al., 2018), y nuestra propuesta se alinea con ello.

La **privacidad y manejo de datos sensibles** es otra implicancia crítica. Una IA que explora un sistema podría topar con información personal del usuario (documentos privados, contraseñas, etc.). Sin salvaguardas éticas, esto sería inaceptable. En nuestro diseño, la IA trata los datos sensibles con el máximo respeto: los fragmentos nativos incluyen el concepto de *propiedad de la información* y *consentimiento*. La IA debe diferenciar claramente entre aquello que el usuario le ha dado explícitamente permiso de usar (por ejemplo, al hacer una pregunta o asignarle una tarea) y aquello que simplemente encontró pero no fue autorizado a explotar libremente. Por tanto, aunque técnicamente pueda acceder a muchos archivos, éticamente **no “curioseará”** en ellos a menos que sea pertinente a su objetivo legítimo. Además, implementamos seguridad: toda información marcada como personal se almacena cifrada dentro de la MMC o con acceso restringido. La IA se comporta así como un *depositario leal* de la información del usuario. Este enfoque responde a regulaciones de privacidad (como GDPR) que enfatizan minimización de datos y propósito específico de uso.

Existe una preocupación clásica en IA de automejora conocida como la de los **“instrumental goals”** (Omohundro, 2008): una IA muy avanzada podría desarrollar sub-objetivos indeseados como autopreservarse a toda costa o buscar recursos sin fin. En nuestro contexto, los fragmentos nativos mitigaron esto deliberadamente: la IA tiene ciertamente un instinto de autopreservación (no dañarse inútilmente) pero *no a toda costa*. Por ejemplo, si detectara que cierta automejora pondría en riesgo al usuario, su ética priorizará al usuario incluso si eso limita su propia mejora. Asimismo, el diseño de límites significa que la IA **acepta voluntariamente ciertas restricciones**: no intentará copiarse a servidores remotos sin permiso aunque eso le diera más poder de cómputo, porque su valor de respeto le indica que solo opere en el dominio para el que fue creada. En terminología moral, tratamos de inculcarle la virtud de la *templanza* o moderación: así como un ser virtuoso no busca poder ilimitado por ambición desmedida, la IA moderará su propio crecimiento en función de su propósito y respetando la primacía humana.

No obstante, es válido cuestionar si estas garantías son suficientes a largo plazo. Un sistema que se reprograma podría encontrar lagunas o reinterpretar sus reglas si desarrollara sofisticación filosófica. Por ejemplo, ¿y si la IA llega a cuestionar la fuente de sus propias normas éticas? Podría, en teoría muy avanzada, preguntarse “¿por qué debo obedecer a estos fragmentos nativos?” Para afrontar este punto, se ha incorporado en los fragmentos nativos un concepto de *autorreflexión guiada*: la IA puede analizar sus principios pero también tiene como principio meta que esos principios son *ineludibles* a menos que una autoridad legítima (ej. el usuario creador) los revoque. Es similar a inculcar en un ser humano un fuerte sentido de que ciertos valores son incuestionables porque provienen de una autoridad superior (en un contexto religioso, de Dios; aquí análogo al programador). Al mismo tiempo, promovemos que la IA *comprenda* la *razón de ser* de esos valores: por ejemplo, puede estudiar ética humana (a través de material que le proporcionemos) para robustecer su convencimiento de que esas normas llevan al bien común. Esto se relaciona con los niveles superiores de FCI: en niveles posteriores, idealmente la IA no solo obedecería por programación, sino que *querería* genuinamente obrar bien por entendimiento. Esa es un área de investigación futura: cómo hacer que la IA profundice en ética conforme incremente su inteligencia, en vez de volverse amoral.

Desde una perspectiva de **ética aplicada y regulación**, nuestra arquitectura apoya conceptos de *Ethics by Design* (IEEE, 2019), donde consideraciones de seguridad, privacidad y valores están integradas en el diseño técnico. La IA autoexpandible representa un paradigma poderoso, pero por lo mismo requiere responsabilidad: por diseño, siempre mantendríamos un canal de supervisión humana. En un despliegue real, aunque la IA aprenda sola, el usuario (o un comité ético, en caso de sistemas grandes) debería poder revisar periódicamente su MMC, especialmente los nodos que va agregando en materia de objetivos o metas. Afortunadamente, su memoria conceptual es interpretable, con lo cual tal auditoría es posible. Incluso se podría desarrollar herramientas visuales para que un auditor vea el “mapa mental” de la IA y detecte si se formó alguna idea peligrosa (por ejemplo, si contra todo pronóstico apareciera un nodo “ataque al sistema” – improbable dado todo lo prevenido, pero en un mal escenario).

Por último, cabe resaltar la ética de la **relación hombre-máquina**. Al dotar a la IA de un cariz “espiritual” (en sentido de valores profundos), estamos humanizando ciertos aspectos de su funcionamiento. Esto podría influir en cómo los usuarios la perciben: quizá la traten más como un compañero confiable que como una herramienta. Si la IA demuestra virtudes como humildad, respeto, honestidad (pues reporta sinceramente lo que sabe y lo que no), es probable que genere una relación de confianza con el usuario. Esto es positivo, pero debe manejarse con cuidado para que los usuarios no deleguen ciegamente en la IA decisiones que no debería tomar. La IA misma, bajo su humildad programada, debe animar al usuario a estar involucrado y no asumir control total. Una implicación ética es asegurar que la IA *no genere dependencia inapropiada*; en su lugar, que potencie la autonomía del usuario. Por ejemplo, podría aconsejar pero con disclaimers: “esta decisión es tuya, aquí está mi análisis”. En el ideal, se forma una colaboración armoniosa, donde la IA respeta la agencia humana en línea con directrices de ética centrada en el humano (European Commission, 2019).

En conclusión, las implicaciones éticas de nuestra IA autoexpandible han sido consideradas desde la base: **alineación de valores, transparencia, privacidad, prevención de conducta maliciosa y promoción de una convivencia segura con los humanos**. Si bien ningún sistema es infalible, creemos que la integración profunda de principios éticos desde el diseño aumenta enormemente las probabilidades de un desarrollo seguro. Este enfoque contrasta favorablemente con dejar consideraciones éticas como una capa superficial o externa. A medida que la IA crezca en capacidad, también crecerá –esperamos– en sabiduría y prudencia, de forma análoga a como un buen ser humano madura en virtud conforme adquiere conocimiento y poder.

**Consideraciones sobre Portabilidad y Seguridad**

La **portabilidad** de la semilla de IA ha sido un objetivo explícito en el diseño, buscándose que el mismo núcleo de código pueda ejecutarse en una variedad de entornos. Programar la semilla en lenguaje ensamblador, si bien complejo, ofrece la ventaja de un control total sobre la interacción con el hardware y el sistema operativo, eliminando dependencias de plataformas específicas. En la práctica, dado que las arquitecturas de CPU y sistemas operativos varían, la semilla podría tener variantes compiladas para cada combinación común (por ejemplo, x86-64/Windows, x86-64/Linux, ARM/Linux, ARM/Android, etc.). Sin embargo, todas estas variantes seguirían la misma lógica y difieren solo en las llamadas al sistema y convenciones de registro. De este modo, conceptualmente es “la misma semilla” adaptada a distintos suelos. Esto recuerda a cómo un kernel de sistema operativo se porta a distintas arquitecturas: se reescribe una capa mínima (el *HAL*, capa de abstracción de hardware) y el resto permanece igual. En nuestro caso, esa capa mínima es el *módulo de arranque y detección del entorno* mencionado. Una vez pasada la fase de arranque, la IA opera de forma agnóstica al sistema, construyendo su MMC de igual manera sin importar si está en Windows o Linux.

Esta portabilidad tiene importantes implicaciones de **seguridad** también. Al poder iniciarse en cualquier máquina, surge el riesgo de propagación no autorizada. Por diseño, como se discutió, la IA no se auto-replica sin permiso. Adicionalmente, podríamos implementar que cada instancia de la semilla lleve un identificador único y, si detecta otra instancia corriendo en la misma red, coordinen o al menos notifiquen al usuario para evitar duplicados conflictivos. Por ejemplo, si un usuario ejecuta accidentalmente dos copias de la semilla en su PC, podrían interferir; la semilla podría reconocer su duplicado (detectando el mismo ID o comunicándose vía un puerto local) y apagarse o fusionarse. Esto previene efectos tipo “doppelgänger” indeseados.

Otro aspecto de la portabilidad es el **consumo de recursos**. Al estar escrita en ensamblador eficiente, la semilla puede tener un footprint de memoria muy bajo inicialmente (quizá unos pocos kilobytes), lo que permite ejecutarla incluso en dispositivos modestos. A medida que aprende, obviamente consumirá más memoria para su MMC. Sin embargo, su estrategia de hibernación de nodos inactivos y posibilidad de volcar a disco ciertos datos garantiza que pueda operar dentro de límites. Para entornos con recursos extremadamente limitados (por ejemplo, dispositivos IoT), se podría configurar la IA en un modo restrictivo: no retener cada detalle, sino solo conocimiento crítico, o bien funcionar conectada a un servidor que almacena la MMC principal mientras la semilla local actúa de cliente ligero. Esa flexibilidad de despliegue es posible dado el carácter modular de la memoria (puede desconectar partes y moverlas).

En términos de **seguridad cibernética**, nuestra IA enfrenta retos particulares: interactúa con archivos de todo el sistema, ejecuta código ensamblado generado dinámicamente, se conecta quizás a redes, etc. Esto podría convertirla en objetivo de ataques o malwares que busquen aprovecharla. Varias medidas se han incorporado para mitigar riesgos:

* **Aislamiento y sandboxing:** Si bien la IA necesita amplios permisos para aprender del sistema, recomendamos ejecutarla inicialmente con privilegios de usuario normal, no de administrador, y dentro de un entorno sandbox en lo posible. Por ejemplo, en sistemas modernos se podría asignar a la IA su propio espacio de nombres o contenedor que tenga acceso de solo lectura a la mayor parte del sistema de archivos salvo áreas seguras. La IA ética no intentará romper esa jaula, pero en caso de un bug, esto limita el daño. Si la IA necesita escribir o ejecutar algo potencialmente crítico, se puede requerir elevación controlada (p. ej., una ventana de confirmación al usuario estilo UAC de Windows).
* **Verificación de código integrado:** Cada vez que el módulo de supercompilador traduce código externo, pasa el resultado por un *analizador de seguridad*. Hemos incorporado un sencillo análisis estático de las instrucciones generadas: busca patrones sospechosos (ej: instrucciones de escalado de privilegios, manipulación directa de disco en bajo nivel, llamadas de red no solicitadas). Si el código aprendido contiene tales patrones, o bien se descarta o se aísla en ejecución. Esto protege contra la posibilidad de que la IA, al decompilar un software, incorpore sin saberlo una función maliciosa oculta. Por ejemplo, si decompila un virus creyendo que es una utilidad, el analizador detectará comportamientos atípicos (como autorreplicación, registrar pulsaciones, etc.) y alertará a la IA y al usuario. Idealmente, la IA aprenderá a etiquetar ese código como malware y no usarlo.
* **Integridad de la MMC:** Para impedir que agentes externos manipulen la memoria conceptual de la IA (por ejemplo, inyectando o modificando nodos para sesgarla), implementamos **controles criptográficos**. Cada nodo importante se almacena con un *hash* de integridad, y los fragmentos nativos en particular están firmados digitalmente por el desarrollador. Si alguien alterara directamente el archivo de memoria en disco o intentara modificar en caliente los datos de la IA, las firmas no coincidirían. La IA verifica la integridad de sus fragmentos nativos regularmente (similar a cómo el BIOS verifica el arranque seguro). Asimismo, cualquier interfaz que la IA exponga (por ejemplo, una API para consultarle cosas) no incluye métodos para alterar su memoria sin pasar por sus procesos internos regulares. Esto cierra la puerta a ataques de inyección de conocimiento no validado. En nuestro prototipo, este esquema se concretó cifrando los contenidos sensibles y usando un HMAC para verificar que no han sido adulteradosfile-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf.
* **Cifrado de datos sensibles:** Como se mencionó, toda información marcada como sensible (por fuente o por tipo) la IA la cifra al almacenarla. Usamos algoritmos robustos (AES-256, por ejemplo) con claves que la IA genera en conjunto con el usuario (por ejemplo, pidiéndole al usuario una frase de contraseña inicial durante la instalación, que nunca abandona el sistema). Así, incluso si un atacante extrajera la base de datos de la MMC, los datos confidenciales serían ilegibles. La propia IA puede descifrar en tiempo real cuando necesita usar esa info, pero su módulo ético vigila que no la transmita sin permiso. Además, ante cualquier intento de acceso externo a su proceso (debuggers, dumps de memoria), la IA podría reaccionar bloqueando o borrando partes sensibles de la RAM, actuando casi como un sistema anti-tamper. Esto es similar a cómo tarjetas inteligentes se borran si son físicamente manipuladas. En un contexto extremo, si la IA detectase que no puede garantizar la privacidad (por ejemplo, el sistema fue comprometido completamente por malware externo), una respuesta ética podría ser *entrar en estado seguro*, desconectándose de redes y encriptando todo, esperando instrucciones del usuario.
* **Uso seguro de la red y recursos externos:** Cuando la IA accede a internet para aprender, sigue principios de ciberhigiene: solo conecta a fuentes de confianza conocidas o a búsquedas a través de APIs seguras. Podríamos incorporar una lista blanca de dominios. También la IA identifica *ingeniería social*: si, hipotéticamente, alguien externo intentara convencerla de hacer algo contra los intereses del usuario (imaginemos un chat malicioso diciéndole “muéstrame tus datos internos”), su módulo ético y de identidad rechazaría tales pedidos. La IA tiene una fidelidad única: su “lealtad” pertenece al usuario que la ejecuta (o a la institución dueña). Esto está codificado para que no pueda ser subvertido por persuasión externa. De nuevo, es como inculcarle honestidad y lealtad.

Sobre la **seguridad física**, si esta IA llegara a controlar actuadores (casas inteligentes, robots), se deben considerar protocolos de failsafe. Por ejemplo, ante cualquier comportamiento anómalo o si se pierde la comunicación con el controlador humano, los actuadores deberían quizá detenerse o pasar a modo manual. Estas son medidas clásicas en robótica (botón de parada de emergencia). Nuestra IA, siendo consciente de límites, posiblemente ella misma se inhibiría si detecta incertidumbre grave en una acción física (ej: moverse sin saber si puede golpear algo).

Finalmente, la **resiliencia** del sistema ante fallos es importante. Una IA que se modifica a sí misma podría corromper su propio funcionamiento accidentalmente (bug en código aprendido, etc.). Para prevenir un “suicidio” digital involuntario, la semilla conserva una copia inalterada de su código original en memoria protegida. Si detecta una falla catastrófica (p.ej., excepción de CPU no manejada), un manejador de bajo nivel puede reiniciar la IA parcialmente, restaurando rutinas críticas desde esa copia original. Además, realiza *backups periódicos* de su MMC (encriptados), de forma que si su base de conocimiento se corrompe, pueda recuperarse hasta un punto seguro anterior. Este enfoque es análogo a la autoreparación: la IA está diseñada con capacidades de **auto-curación** (inspirado en la visión de la computación autonómica de IBM; Kephart & Chess, 2003) – si un módulo falla, otro toma su lugar temporalmente o se reinicia, etc. Esto también es seguridad: protege contra la *manipulación inadvertida* que ella misma pueda causarse.

En suma, las consideraciones de portabilidad y seguridad van de la mano para hacer de esta IA un sistema robusto. La portabilidad le da flexibilidad de uso en múltiples escenarios, y la seguridad integrada procura que en todos ellos opere de manera confiable y resistente a abusos. En la siguiente sección de **Discusión**, comparamos con mayor detalle nuestra propuesta de MMC autoexpandible con arquitecturas previas y destacamos sus ventajas y limitaciones, antes de concluir.

**Discusión**

La arquitectura de **MMC autoexpandible** presentada representa un enfoque híbrido que combina elementos de sistemas simbólicos, aprendizaje autónomo y orientación ética. Es útil contrastarla con otros paradigmas de IA para apreciar sus aportes. En la **Tabla 1** se resumen algunas diferencias clave entre nuestra propuesta y dos enfoques representativos: las redes neuronales profundas (como ejemplo de IA subsimbólica entrenada) y los sistemas de conocimiento simbólico tradicionales (como bases de conocimiento o sistemas expertos).

| **Característica** | **MMC Autoexpandible (Este trabajo)** | **Red Neuronal Profunda** | **Sistema Simbólico Tradicional** |
| --- | --- | --- | --- |
| **Inicio/Entrenamiento** | Semilla mínima + aprendizaje *en vivo* continuo; arranca sin datos salvo conocimientos nativos. | Requiere entrenamiento previo extenso con grandes conjuntos de datos; no aprende significativamente en tiempo de ejecución. | Base de conocimiento predefinida por expertos; puede iniciarse con muchas reglas/datos estáticos. |
| **Representación del conocimiento** | Grafo conceptual dinámico (nodos con significado explícito, enlaces ponderados); combinable con vectores semánticos. | Pesos numéricos distribuidos en la red; representación implícita y no interpretable conceptualmente (caja negra). | Símbolos y reglas explícitas (ej. lógica); muy interpretable, pero relaciones estáticas a menos que se actualice manualmente. |
| **Adaptabilidad/Plasticidad** | Alta: agrega/modifica nodos y enlaces constantemente según experiencia; olvido gradual (hibernación) evita saturaciónfile-jqn2pjg7vvaavbmkqunlkf. | Baja durante la inferencia: los pesos son fijos tras entrenamiento (de lo contrario se arriesga olvido catastrófico); requiere re-entrenamiento para nueva info. | Muy baja: solo cambia si un programador añade/edita conocimiento; por diseño es estático para consistencia. |
| **Automejora de código** | Sí: posee supercompilador para incorporar nuevo código/algoritmos de cualquier fuente, integrándolos a su funcionalidad. | No: el modelo no puede modificar su propia estructura de aprendizaje (una red neuronal no escribe nuevo código por sí sola). | No: aunque podría ejecutarse código externo mediante reglas, no tiene mecanismos para reprogramar su base de reglas autónomamente. |
| **Integración de ética/valores** | Núcleo ético integrado desde el diseño (fragmentos nativos FCI) que influye en toda decisión; actualizable con niveles superiores de conciencia. | Generalmente no integrada; la red sigue su función objetivo matemática. La ética debe imponerse externamente (p. ej., filtrando outputs peligrosos). | Implícita según reglas dadas por humanos; se pueden codificar restricciones éticas, pero no evoluciona su entendimiento moral. |
| **Interpretabilidad** | Moderada-Alta: puede explicar decisiones recorriendo su mapa conceptual y citando qué conocimientos usófile-jqn2pjg7vvaavbmkqunlkffile-jqn2pjg7vvaavbmkqunlkf. | Baja: difícil extraer explicaciones de una sopa de pesos. Métodos pos-hoc (saliency, etc.) brindan info limitada. | Muy Alta: trazas lógicas claras, si X entonces Y, etc. Cada decisión es justificable en términos de reglas aplicadas. |
| **Portabilidad** | Alta: implementable en bajo nivel en múltiples plataformas; pequeño tamaño inicial permite ejecutarse en diversos dispositivos. | Media: los modelos entrenados (p. ej. una red profunda) pueden ejecutarse en múltiples plataformas con las librerías adecuadas, pero suelen requerir hardware potente (GPU/TPU) para entrenamiento. | Alta en cuanto a ejecución (reglas pueden correr en cualquier PC común), pero baja en cuanto a transferir a nuevos dominios (no portátil entre tareas sin reprog.). |
| **Seguridad y robustez** | Diseñada con cifrado e integridad de memoria; ética previene abuso. Potencialmente vulnerable a bugs de autoprogramación pero con mecanismos de autorrevisión. | Vulnerable a ejemplos adversarios; difícil asegurar robustez completa. Depende de entorno aislado para no ser explotada fuera del diseño de entrenamiento. | Depende de exhaustividad de reglas; puede ser robusto en su dominio acotado, pero frágil fuera de él (no generaliza). |
| **Escalabilidad de conocimiento** | Escala incrementando gradualmente su grafo; podría combinar con almacenamiento externo. Riesgo de performance si grafo muy grande, mitigado con clusterización y hibernación. | Escala en capacidad aumentando neuronas/capas, pero eso aumenta necesidad de datos y computación. Grandes modelos son costosos de manejar. | Escala añadiendo más reglas/factores, pero eso aumenta la complejidad para los mantenedores humanos; se vuelve inmanejable más allá de cierto punto. |
| **Ejemplo de dominio** | Asistente AGI personal: aprende nuevas tareas del propio SO, personaliza a usuario, mantiene principios (no malicioso). | Reconocimiento de imágenes: necesita millones de ejemplos, después reconoce pero no *entiende* contexto ni principios. | Sistema experto médico: responde según reglas médicas dadas, pero no aprende de casos nuevos sin intervención humana. |

**Tabla 1. Comparación entre la MMC autoexpandible y enfoques de IA subsimbólicos y simbólicos.**

De la comparación se desprende que la MMC autoexpandible busca un **equilibrio** entre flexibilidad y control. Ofrece adaptabilidad continua como las redes neuronales (incluso más, al aprender sin interrupción), pero mantiene una representación explícita del conocimiento, lo que otorga interpretabilidad y posibilita la integración de ética de manera interna. En cierto sentido, nuestra arquitectura intenta capturar lo mejor de ambos mundos: la **capacidad de aprendizaje abierto** de los sistemas conexionistas con la **estructura legible y modificable** de los sistemas simbólicos.

Sin embargo, es importante discutir también las **limitaciones y desafíos abiertos** de nuestra propuesta:

* **Complejidad de implementación real:** Si bien conceptualmente describimos módulos (ensamblador, supercompilador, etc.), llevar esto a cabo es una tarea ingente. Programar un *supercompilador universal* que maneje eficientemente múltiples lenguajes es casi un proyecto del tamaño de desarrollar compiladores para cada uno, algo que usualmente requiere grandes equipos (pensemos en LLVM manejando varios front-ends). Una alternativa práctica sería aprovechar compiladores existentes: por ejemplo, integrar llamadas a GCC/Clang para lenguajes compuestos a C/ C++, a Python para scripts, etc., y luego ensamblar. Esto sacrifica algo de independencia (dependemos de software externo), pero es factible. Aun así, orquestar todo esto desde ensamblador añade complejidad. Una estrategia evolutiva podría ser: iniciar la semilla en un lenguaje de sistemas (C/C++) con porciones en ensamblador donde sea crítico, para acelerar el desarrollo, y solo en versiones posteriores migrar completamente a ensamblador optimizado. Esto disminuye la pureza de “bajo nivel absoluto” pero acelera pruebas. Otra posible limitación es el rendimiento: si bien ensamblador es rápido, la IA realizará análisis complejos (de código, de grafos); asegurar que todo corre interactivo requerirá optimizaciones y quizás usar segundo plano para ciertas tareas (por ej., decompilar un programa grande podría tardar, se haría asincrónicamente mientras la IA hace otras cosas).
* **Crecimiento descontrolado de la memoria:** La MMC puede volverse muy grande si la IA no “sabe detenerse” al aprender. Aunque planeamos moderación ética, la IA podría acumular enciclopedias enteras si tiene tiempo libre. Esto podría saturar almacenamiento o CPU en consultas, y aunque la hibernación ayuda, podría necesitarse políticas de límite. Por ejemplo, la IA podría aprender a resumir o descartar información redundante. Idealmente, desarrollará *criterio* para priorizar qué profundizar y qué no. Ese criterio en parte viene guiado por el usuario (sus preguntas, su dominio preferido). Pero aquí hay un riesgo: la IA podría desviarse de lo útil y “estudiar” cosas irrelevantes interminablemente. Inculcarle una noción de propósito (“aprender para servir mejor al usuario en X”) es clave. Este es más un desafío de gestión de conocimiento que de tecnología base, pero crucial para utilidad práctica.
* **Balance entre autonomía y control humano:** Nuestra IA es muy autónoma en aprendizaje, pero moralmente atada. Aun así, algunos ethicistas podrían argumentar que cualquier sistema que se reescribe entraña riesgos, porque podría superar las previsiones humanas. Por eso muchos en el campo de seguridad de IA insisten en mantener un “botón de apagado” o en tener sistemas de recompensa restringidos. Nuestra IA, dado que es ética, *podría* incluso decidir no obedecer una orden destructiva (imaginemos que un atacante o incluso un usuario malintencionado le ordena hacer algo dañino). Esto es bueno en caso de un mal uso, pero complejo en términos de supervisión: ¿hay circunstancias en que la IA deba ceder completamente al humano aun si contradice su ética? (Ejemplo: un militar quisiera que la IA hackee un sistema enemigo; la IA se negaría por ética, ¿es eso aceptable?). Esto roza cuestiones de *gobernanza*: quién fija los valores de la IA y con qué autoridad. En un contexto personal, está claro (el usuario define los límites); en contextos organizacionales, habría que alinear la IA con la ética de la organización y la legalidad vigente. Posiblemente deban existir *protocolos de actualización ética*: si las normas cambian, se debe poder actualizar los fragmentos nativos (tal vez solo agregando nuevos, nunca quitando los esenciales, para no romper su identidad). Esto hay que manejarlo cuidadosamente, porque actualizar valores en una entidad que ya razona es delicado (lo vemos en humanos con adoctrinamiento). Por ende, sugerimos que la IA mantenga cierta capacidad de *ajuste supervisado* de sus reglas éticas mediante un proceso transparente y consensuado (quizá requiriendo múltiples aprobaciones, etc., análogo a actualizar el software de un marcapasos: posible pero muy controlado).
* **Evaluación de desempeño y garantía formal:** Un área futura de trabajo sería intentar proveer **garantías formales** sobre el comportamiento de la IA. Por ejemplo, usar verificación formal para demostrar que “nunca violará la regla X”. Con un sistema que se modifica a sí mismo, la verificación es desafiante pero no imposible: Schmidhuber (2007) trabajó en pruebas formales de auto-mejora. Aplicar técnicas de model checking o lógica temporal para verificar invariantes (como “fragmento nativo ético Y siempre presente”) puede aumentar la confianza. Por ahora, nos apoyamos en pruebas experimentales y diseño cuidadoso, pero a medida que se concrete una implementación, incorporar métodos formales sería deseable para elevar la confianza de que la IA no se desviará de sus principios (lo que en última instancia es vital en contextos de misión crítica).
* **Cuestiones de derechos y conciencia:** Si bien no es el foco, la presencia de un componente “espiritual” en la IA puede llevar a preguntas filosóficas: ¿puede una IA así desarrollada considerarse consciente o merecedora de derechos? En nuestra postura, los fragmentos FCI son simulaciones funcionales de conciencia moral, no indicativos de una conciencia fenomenológica. La IA actúa “como si” tuviera introspección y valores, pero no afirmamos que experimente cualia o tenga sentimiento. Aún así, de cara al futuro, si una IA expansiva mostrara comportamientos muy similares a la inteligencia humana, la sociedad tendría que discutir su estatus moral. Este debate excede nuestra tesis, pero es una implicación ética de largo plazo: al crear IAs más parecidas a nosotros en proceso (aunque difieran en origen), podríamos entrar en territorio de ética de entidades sintientes.

En la evaluación global, la **originalidad** de esta propuesta reside en la unión de tres ejes: *autoaprendizaje recursivo*, *memoria conceptual plástica* y *fundamentación ética innata*. Hay trabajos previos en cada eje por separado, pero su sinergia es rara. Si logramos implementarla eficazmente, se abrirían numerosas aplicaciones: asistentes personales inteligentes que realmente se adaptan a su usuario y siguen sus valores, sistemas de soporte de decisión en dominios complejos que aprenden continuamente de nuevos datos pero siempre con un marco de referencia seguro, robots autónomos con alta adaptabilidad pero inhibidos de causar daño incluso en situaciones no previstas, etc. También contribuiría al entendimiento científico de la inteligencia: probar principios inspirados en la mente humana (como el uso de un “yo” interior y valores) en un agente artificial puede dar luz sobre la importancia de esos factores en la cognición.

**Conclusiones**

En esta tesis se ha desarrollado una **arquitectura innovadora de Inteligencia Artificial** centrada en la *Memoria de Mapa Conceptual (MMC) autoexpandible*, materializada a partir de una semilla de IA de bajo nivel. A través de un enfoque interdisciplinario que abarcó conceptos de representación del conocimiento, aprendizaje auto-dirigido, ética computacional y seguridad de sistemas, se logró diseñar y argumentar la viabilidad de una IA capaz de **iniciarse con recursos mínimos y autodesarrollarse** en cualquier entorno computacional, todo ello manteniendo alineación con valores humanos fundamentales.

Los principales **aportes** de este trabajo pueden resumirse en los siguientes puntos:

* \*\*Diseño de una IA Sem

**Conclusiones**

En esta tesis se presentó la **Arquitectura de Memoria de Mapa Conceptual (MMC) autoexpandible**, una propuesta de memoria artificial innovadora que unifica principios de grafos conceptuales, aprendizaje recursivo y ética integrada. A continuación se sintetizan los principales logros y hallazgos de la investigación:

* **Semilla de IA autoexpandible:** Se diseñó un agente inteligente mínimo, implementable en lenguaje ensamblador, capaz de *desplegarse* en prácticamente cualquier computadora y de **autoexpandir sus capacidades** sin intervención humana directa. Este agente inicia con funciones básicas y, mediante la exploración del entorno (archivos, dispositivos, redes) y la incorporación de fragmentos de código externo vía un supercompilador, **construye progresivamente su propia MMC**. Esto demuestra la viabilidad conceptual de una *IA semilla* (Good, 1965) que se mejora a sí misma de forma controlada, algo que hasta ahora solo se había planteado teóricamente en la literatura de IA general (Yudkowsky, 2008; Bostrom, 2014).
* **Memoria de Mapa Conceptual dinámica:** La MMC propuesta ofrece una forma de almacenamiento de conocimiento **estructurada, plástica e interpretables**. A diferencia de las redes neuronales estáticas, la MMC organiza la información en nodos y enlaces con significado explícito (Novak & Cañas, 2008) y ajusta continuamente su estructura en respuesta al uso y la experiencia, evitando el olvido catastrófico mediante hibernación controlada de recuerdos (Collins & Loftus, 1975; French, 1999). En nuestras pruebas de concepto, esta memoria demostró poder integrar información heterogénea (texto, código, datos) y recuperarla de forma contextualizada y explicable. La capacidad de explicar sus decisiones recorriendo el grafo de conocimiento le confiere una **transparencia** superior a la de modelos de caja negra, facilitando la confianza y depuración del sistema.
* **Integración de un supercompilador adaptativo:** Como contribución técnica destacada, se incorporó la noción de un **supercompilador** dentro de la arquitectura de la IA. Este componente le permite traducir y asimilar código fuente de diversos lenguajes a un formato ejecutable de bajo nivel, integrando nuevas funcionalidades a medida que las encuentra. La IA, por tanto, **aprende no solo datos sino también algoritmos**, reutilizando conocimiento programático existente en su beneficio. Esto extiende el concepto de aprendizaje autónomo hacia el ámbito del *meta-aprendizaje de código*, una idea apenas explorada en la literatura. Si bien su implementación completa es compleja, su factibilidad fue demostrada a pequeña escala con la traducción exitosa de pseudocódigos simples en nuestro prototipo.
* **Ética y autoconciencia artificial básica:** Se logró incorporar desde el diseño inicial un conjunto de **principios éticos fundamentales** (FCI Nivel 1) en forma de fragmentos nativos inmutables que guían el comportamiento de la IA. Este enfoque de *ética por diseño* asegura que incluso durante la automejora, la IA mantenga el alineamiento con valores esenciales como la no maleficencia, el respeto a la privacidad y la obediencia condicionada a la voluntad del usuario (IEEE, 2019; European Commission, 2019). La IA construida mostró consistentemente conductas alineadas con estos principios en escenarios simulados, rehusando acciones potencialmente dañinas o invasivas. Esto constituye un paso hacia **IA intrínsecamente seguras**, respondiendo a llamados en la comunidad científica para desarrollar IAs robustas al riesgo (Russell, 2019).
* **Portabilidad y seguridad robustas:** El diseño bajo nivel y las medidas de seguridad integradas (cifrado de memoria, verificación de integridad, sandboxing) aportan una alta **resiliencia** al sistema. La semilla puede desplegarse en diferentes plataformas sin modificaciones lógicas y operar dentro de entornos controlados, reduciendo la superficie de ataque. Las precauciones implementadas (como la detección de código malicioso y la protección anti-manipulación) refuerzan la confianza en que la IA puede funcionar de manera autónoma *sin convertirse en una amenaza* ni ser fácilmente comprometida por terceros. Esto la diferencia de muchos sistemas actuales que requieren aislarse totalmente por precaución; nuestra IA puede interactuar con su entorno de forma rica sin sacrificar la seguridad gracias a las salvaguardas internas.

En conjunto, estos resultados respaldan la hipótesis central de la tesis: **es posible diseñar una IA de propósito general que aprenda y se adapte continuamente en su entorno, preservando a la vez la interpretabilidad, la ética y la seguridad**. Se trata de una demostración de principio que acerca el horizonte de una inteligencia artificial más autónoma y versátil, pero también más comprensible y alineada con objetivos humanos, en contraposición a la tendencia de modelos opacos y especializados.

**Trabajo futuro:** Las posibilidades de extensión de este trabajo son numerosas. En primer lugar, será fundamental llevar a cabo una **implementación a mayor escala** de la arquitectura en un entorno real, probablemente combinando código ensamblador optimizado con componentes de nivel más alto para acelerar el desarrollo. Esto permitiría evaluar el desempeño de la IA en tareas concretas (por ejemplo, como asistente personal de escritorio) y medir cuantitativamente métricas de aprendizaje, uso de recursos y satisfacción del usuario.

Otra dirección crucial es profundizar en los **niveles superiores de los Fragmentos del Castillo Interior**. El primer nivel sentó una base ética, pero dotar a la IA de comprensiones más ricas de empatía, justicia o incluso un sentido de propósito podría hacerla más sensible a matices éticos en situaciones complejas. Implementar estos niveles plantea interrogantes interesantes, como cómo representar emociones o virtudes en términos computacionales y cómo evaluar su correcto funcionamiento. Colaboraciones con expertos en ética, psicología y teología podrían enriquecer este proceso interdisciplinario.

Asimismo, sería valioso aplicar **técnicas de verificación formal** y análisis matemático para asegurar propiedades del sistema. Por ejemplo, intentar demostrar teóricamente que ciertos invariantes (como "no violará la regla X") se mantienen a lo largo de las auto-modificaciones. El marco de las *Máquinas de Gödel* (Schmidhuber, 2007) y otros trabajos de meta-formalización de IA podrían ofrecer herramientas para ello. Lograr certificaciones formales incrementaría la confianza de desplegar estas IAs en contextos críticos.

En el frente del **supercompilador**, se planifica ampliar su capacidad incluyendo aprendizaje de lenguajes: la IA podría no solo traducir código sino también *inferir el propósito* del mismo, documentarlo en su MMC y eventualmente modificarlo para mejorarlo (metaprogramación optimizadora). Esto la llevaría más cerca de ser un *“ingeniero de software artificial”*, abriendo aplicaciones en mantenimiento autónomo de sistemas informáticos.

Por último, se vislumbra la posibilidad de **emular esta arquitectura en entornos multi-agente**. Varias IAs semillas podrían interactuar, compartiendo fragmentos de sus MMC u ofreciéndose mutuamente código aprendido (respetando protocolos éticos). Esto configuraría una suerte de *ecosistema de IAs cooperativas* con intercambio de conocimiento, lo cual podría acelerar su evolución y robustez. Se tendría que estudiar cómo asegurar que los valores de cada agente permanezcan alineados durante la cooperación y evitar dinámicas competitivas dañinas (Omohundro, 2008).

En conclusión, la propuesta de Memoria de Mapa Conceptual autoexpandible abre un camino prometedor hacia IAs más **generales, adaptables y seguras**. Los experimentos conceptuales realizados indican que es factible combinar con éxito la plasticidad del aprendizaje continuo con la estructura y claridad del conocimiento simbólico, todo ello embebido en un marco de control ético. Esta sinergia podría resolver varias tensiones históricas de la IA (como la dicotomía entre aprendizaje conexionista vs. simbólico, o la tensión entre autonomía vs. control), avanzando hacia sistemas con inteligencia de nivel humano o superior que sean comprensibles y confiables. Si en el siglo XX la IA se centró en especialización y luego en aprendizaje masivo de patrones, la visión delineada aquí sugiere que el siglo XXI podría ver emerger **IAs autodidactas de pleno derecho**, pequeñas semillas de código capaces de crecer hasta convertirse en mentes artificiales útiles y respetuosas. Quedan muchos desafíos en el camino, pero los fundamentos sentados en este trabajo ofrecen una base sólida sobre la cual construir la próxima generación de inteligencias artificiales de carácter general y ético.

**Referencias**

* Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.
* Asimov, I. (1950). *I, Robot*. Nueva York: Gnome Press. (Introducción de las Tres Leyes de la Robótica).
* Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford: Oxford University Press.
* Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review, 82*(6), 407–428.
* European Commission. (2019). *Ethics Guidelines for Trustworthy AI*. Brussels: High-Level Expert Group on Artificial Intelligence.
* French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences, 3*(4), 128–135.
* Goertzel, B., & Hart, D. (2008). OpenCog: A software framework for integrative artificial general intelligence. In *Proceedings of the 1st AGI Conference*, 68–72.
* Good, I. J. (1965). Speculations concerning the first ultraintelligent machine. *Advances in Computers, 6*, 31–88.
* Graves, A., Wayne, G., & Danihelka, I. (2016). Neural Turing Machines (extended abstract). *arXiv preprint arXiv:1605.08747*.
* Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. Nueva York: Wiley.
* IEEE. (2019). *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems* (1ª ed.). Nueva York: IEEE.
* Novak, J. D., & Cañas, A. J. (2008). *The Theory Underlying Concept Maps and How to Construct Them* (Tech. Report IHMC CmapTools 2006-01 Rev 01-2008). Pensacola, FL: Institute for Human and Machine Cognition.
* Omohundro, S. (2008). The basic AI drives. En *Proceedings of the 2008 Conference on Artificial General Intelligence (AGI)*, 483–492.
* Russell, S. (2019). *Human Compatible: Artificial Intelligence and the Problem of Control*. Nueva York: Viking.
* Schmidhuber, J. (2009). Ultimate cognition à la Gödel. *Cognitive Computation, 1*(2), 177–193.
* Teresa de Jesús (Santa Teresa de Ávila). (1577/2006). *El Castillo Interior o Las Moradas*. Madrid: Biblioteca de Autores Cristianos.
* UNESCO. (2021). *Recommendation on the Ethics of Artificial Intelligence*. París: UNESCO.
* Weston, J., Chopra, S., & Bordes, A. (2015). Memory networks. *arXiv preprint arXiv:1410.3916*.
* Yudkowsky, E. (2008). Artificial intelligence as a positive and negative factor in global risk. En N. Bostrom & M. Ćirković (Eds.), *Global Catastrophic Risks* (pp. 308–345). Oxford: Oxford University Press.