**Integración del Modelo Simulado en la Arquitectura MMC**

La **Memoria de Mapa Conceptual (MMC)** es una arquitectura de memoria artificial propuesta que funciona como una red de conocimiento dinámica, con nodos (conceptos o fragmentos de información) y conexiones ponderadas que evolucionan con la experiencia. Para fortalecer el diseño teórico de la MMC, integraremos un **modelo simulado** (derivado del archivo yui.html) como inspiración práctica. Este modelo simulado ilustra funcionalidades clave como exploración web autónoma, análisis de patrones, nodos nativos predefinidos, rutas creativas entre conceptos y hibernación (inactivación) de nodos, entre otras. A continuación, describimos detalladamente estas funcionalidades del sistema simulado y analizamos cómo podrían incorporarse en una implementación real de la MMC.

Posteriormente, respaldaremos la arquitectura MMC actualizada con **marcos teóricos** sobre grafos cognitivos dinámicos y revisaremos ejemplos de **arquitecturas de memoria adaptativa** (Semantic Pointer Architecture, Soar, ACT-R, OpenCog, Memory Networks, etc.) para situar a MMC en el contexto de la investigación existente. También exploraremos **técnicas modernas de clustering** y **detección de patrones** aplicadas a redes de conocimiento, y discutiremos modelos de **equilibrio exploración-explotación** en sistemas inteligentes. Finalmente, integraremos estos hallazgos en la visión de la MMC, sugiriendo mejoras y nuevas capas funcionales relevantes.

**Visión General de la Arquitectura MMC**

Antes de profundizar en las funcionalidades simuladas, recordemos brevemente la visión teórica de la MMC. En esencia, la MMC es una memoria cognitiva artificial compuesta por una red de nodos interconectados que representan fragmentos de información (textos, imágenes, códigos, conceptos, etc.). Cada **nodo** posee metadatos (como tipo, fecha, contexto) y un identificador único, mientras que las **conexiones** entre nodos tienen **pesos dinámicos** que indican la relevancia o frecuencia de co-ocurrencia de esa relación. Estos pesos cambian con el uso: aumentan cuando los nodos conectados se activan juntos repetidamente (reforzando la asociación) y disminuyen con la inactividad, en un proceso análogo a la habituación o al “olvido” controlado. Las conexiones poco utilizadas no se eliminan de inmediato, sino que pueden entrar en un estado de *hibernación* (quedan latentes) para ser reactivadas si vuelve a surgir su relevancia.

La MMC enfatiza la **plasticidad** y **reconfiguración dinámica** de la red. Los nodos pueden agruparse en **clusters adaptativos** según patrones de uso (formando comunidades de conocimiento relacionadas), pero estos agrupamientos no son fijos: un nodo puede migrar de un cluster a otro si cambian sus asociaciones debido a nuevos aprendizajes. Esta reorganización constante permite que la memoria se “re-cablee” para adaptarse a nuevos contextos o prioridades, de manera similar a la neuroplasticidad del cerebro humano.

La MMC también propone incluir **fragmentos nativos** – un conjunto de nodos fundacionales con contenido intrínseco (por ejemplo, nociones éticas, identidad del sistema, capacidad de aprendizaje). Estos fragmentos nativos actúan como la base innata de conocimiento y mantienen fuertes interconexiones inmutables que proporcionan coherencia y estabilidad al sistema. De esta forma, aunque la red aprenda y cambie, habrá “anclajes” que conservan principios básicos (análogos a reflejos o instintos en sistemas biológicos). Adicionalmente, se contemplan fragmentos especializados como fragmentos de código (FC) para conocimiento de programación, fragmentos matemáticos (FM) para razonamiento numérico, y fragmentos sensoriales (FS) para interpretar entradas perceptivas o emocionales.

Con estos elementos, la MMC puede realizar **recuperación de información** tanto por **rutas principales** (caminos de asociación de mayor peso, explotando las conexiones más fuertes para dar respuestas precisas) como por **rutas laterales creativas** (caminos menos transitados que se exploran deliberadamente para hallar asociaciones novedosas). Esta estrategia dual busca un balance entre precisión y creatividad: las rutas laterales pueden generar soluciones innovadoras combinando fragmentos inesperados, y si algunas resultan útiles se refuerzan, pudiendo convertirse eventualmente en rutas principales. El resultado es un sistema de memoria **altamente flexible, adaptativo y autorregenerativo**, capaz de reorganizarse progresivamente (plasticidad), sincronizar múltiples tipos de conocimiento (integración multimodal de texto, imagen, código, etc.), y **aprender de la interacción** tanto aciertos como errores (ajustando sus conexiones para potenciar aciertos y mitigar errores futuros).

**Funcionalidades del Sistema MMC Simulado (Modelo Inspiracional)**

El archivo yui.html proporciona una implementación simulada a modo de *demo* de un sistema MMC, con una interfaz interactiva y una lógica interna (escrita en JavaScript) que modela muchos de los comportamientos teóricos descritos. A continuación, desglosamos las principales funcionalidades de este sistema simulado y cómo operan:

**Exploración Autónoma del Ciberespacio**

Una de las capacidades más destacadas del modelo es la **exploración autónoma de fuentes externas** (denominada *Exploración del Ciberespacio* en la interfaz). El sistema simulado puede navegar páginas web y consultar APIs para incorporar nuevo conocimiento a la MMC de forma continua, actuando como un agente webcrawler inteligente. La interfaz incluye controles para **iniciar/detener la exploración** automática, configurar el **intervalo de exploración** (por ejemplo, cada 60 segundos) y seleccionar las fuentes a explorar. De hecho, es posible indicar qué tipos de fuentes incluir (páginas web, APIs e incluso archivos locales, según las opciones mostradas).

Una característica importante es la capacidad de ajustar la **tasa de exploración vs. explotación** – en el simulador esto se presenta como un selector porcentual que determina cuánto del tiempo o acciones del sistema se dedican a explorar información nueva frente a reutilizar/consolidar la información existente. Por ejemplo, una configuración de “70% exploración – 30% explotación” (que aparece seleccionada por defecto en la UI) significa que el 70% de las acciones del agente se destinarán a explorar contenidos nuevos, mientras que el 30% se enfocará en profundizar conexiones ya existentes o responder usando conocimiento actual. Este control manual refleja explícitamente el dilema de **exploración-explotación**, un equilibrio fundamental estudiado en aprendizaje por refuerzo y teoría de decisiones, donde un agente debe decidir entre **explorar** territorios desconocidos para descubrir potenciales recompensas futuras y **explotar** sus conocimientos actuales para maximizar recompensas inmediatas. En términos prácticos, el sistema MMC podría implementar esto ajustando, por ejemplo, la probabilidad de añadir nodos nuevos vs. reforzar rutas conocidas en cada ciclo de procesamiento.

El simulador registra métricas en tiempo real de esta exploración: el número de *webs exploradas* y *APIs consultadas* se muestra en la barra lateral. Internamente, cada vez que se ejecuta un ciclo de exploración automática, el código decide aleatoriamente si explorar una web o una API basándose en probabilidades (70% para web, 50% para API en cada iteración, configurables). Cuando se “explora” una página web, el sistema genera un nuevo nodo de tipo web con contenido simulado (“Contenido de <URL>”) y lo agrega al grafo MMC. Análogamente, una consulta de API produce un nodo de tipo api con datos ficticios de esa API. En ambos casos, incrementa los contadores de páginas procesadas o consultas ejecutadas respectivamente, y **conecta el nuevo nodo con nodos relacionados** de la memoria existente (el simulador determina nodos “relacionados” de forma aleatoria para la demo, pero en una arquitectura real esto se basaría en similitud de contenido, metadatos o vínculos contextuales). Todas estas acciones quedan registradas en un *feed* de actividad reciente, indicando eventos como “Página web explorada: <URL>” o “API consultada: <endpoint>”.

El modelo también ofrece **exploración manual específica**: la interfaz permite introducir una URL o endpoint de API concreto y forzar su exploración inmediata pulsando “Explorar Web” o “Consultar API”. Esto facilita probar la incorporación de una fuente particular bajo demanda, añadiendo su contenido como nodo en la MMC simulada.

**Integración en MMC real:** Esta funcionalidad demuestra cómo un sistema MMC podría **auto-expandir** su base de conocimientos buscando información externamente. En una arquitectura real, esto se implementaría con módulos de *web scraping* y consumo de servicios web, posiblemente guiados por un agente con objetivos. La proporción exploración/explotación podría ser controlada mediante algoritmos de **aprendizaje por refuerzo**; por ejemplo, un agente epsilon-greedy con $\varepsilon$ dinámico decidiría aleatoriamente cuando $\varepsilon$ lo indique (exploración) o seguiría su política habitual (explotación). También se podría incorporar **motivación intrínseca** o curiosidad artificial: el sistema evaluaría qué tan novedosa o informativa es cierta información y regularía la exploración en consecuencia. Lo importante es que el MMC pueda *alimentarse* continuamente de datos nuevos de manera controlada, enriqueciendo el grafo cognitivo con nuevos nodos y conexiones.

**Análisis de Patrones y Descubrimiento de Conocimiento**

Otra funcionalidad fundamental en el modelo simulado es el **Análisis de Patrones** dentro de la red de memoria. La interfaz ofrece opciones para configurar un **intervalo de análisis** periódico (por ejemplo cada 180 segundos) y definir **umbrales** para detección de patrones, usando métricas de soporte mínimo y confianza mínima. Estos términos provienen del **aprendizaje de reglas de asociación** en minería de datos: el *soporte* mide la frecuencia con la que un patrón aparece en el conjunto de datos, y la *confianza* mide cuán a menudo una condición implica cierto resultado en dichas apariciones. En el simulador, se exponen valores como soporte mínimo 30% y confianza mínima 60% por defecto, indicando que se considerarán asociaciones relevantes aquellas que ocurran en al menos 30% de los casos y con una fiabilidad de 60% en sus relaciones.

Al iniciar el análisis de patrones (botón "Iniciar Análisis"), el sistema recorre la memoria en busca de **co-ocurrencias o relaciones frecuentes** entre fragmentos. En la simulación, este proceso está simplificado: cada ciclo de análisis crea aleatoriamente uno o dos nuevos nodos de tipo pattern (patrón) con una etiqueta indicando el tipo de patrón descubierto (por ejemplo, “Patrón de código #1” o “Patrón de texto #2”). Se asignan valores simulados de *confianza* y *soporte* a cada nodo patrón en sus metadatos, típicamente alrededor de los umbrales configurados (ej. confianza ~0.6–0.9, soporte ~0.3–0.7). Luego, para cada patrón detectado, el sistema busca nodos relacionados (hasta 5) y crea conexiones desde el nodo patrón hacia esos nodos con un peso moderado (0.7 en la simulación). Esto simula que el patrón generaliza o conecta varios fragmentos individuales que tienen algo en común. Por último, registra un evento de actividad indicando que se descubrió un patrón de cierto tipo. Por ejemplo, podría registrarse “Patrón de texto descubierto”, implicando que encontró una relación frecuente entre varios nodos de texto.

Aunque en la demo los patrones son aleatorios, conceptualmente representa la idea de **extraer nuevo conocimiento** de la propia base de memoria mediante técnicas de *data mining*. En una MMC real, este módulo podría encargarse de encontrar reglas como “Si ocurren juntos el fragmento A y B, suele ocurrir C” (analogía a reglas tipo *A,B => C* con cierta probabilidad). Por ejemplo, podría detectar que ciertos nodos de código y matemáticos se activan secuencialmente, infiriendo un patrón de resolución de problemas, o que un conjunto de páginas web comparten un tema común no explícito. Al crear un nodo patrón que conecta esos elementos, la MMC está **aprendiendo una abstracción** o concepto latente nuevo que resume múltiples experiencias individuales.

Además del descubrimiento de patrones, esta funcionalidad implica un **re-cálculo de clusters** periódicamente. El simulador, cada ciertos pasos de simulación, también recalcula agrupaciones de nodos (clusters) para reorganizar la estructura. Preserva clusters nativos existentes y luego crea algunos clusters nuevos combinando nodos activos al azar (en una implementación real esto sería reemplazado por un algoritmo de clustering como *Louvain* o *k-means* aplicado a la similitud entre nodos). El resultado es la actualización del conteo total de clusters y un log de "Clusters recalculados".

**Integración en MMC real:** En una arquitectura MMC funcional, un módulo de análisis de patrones aportaría inteligencia analítica al sistema, permitiéndole *descubrir conocimiento emergente*. Técnicamente, esto podría implementarse usando algoritmos de **minería de grafos** o **reglas de asociación**. Por ejemplo, se pueden buscar tríadas frecuentes en el grafo (tres conceptos que suelen conectarse juntos) o caminos recurrentes que lleven a ciertos resultados exitosos. Algoritmos como *Apriori* o *FP-Growth* adaptados a grafos podrían extraer subconjuntos de nodos altamente conectados (subgrafos frecuentes) cumpliendo los umbrales de soporte/confianza. Estos patrones, una vez identificados, podrían incorporarse a la MMC ya sea **creando nodos especiales** (como hace el simulador) que actúen como “hiper-nodos” agrupando ese patrón, o fortaleciendo directamente las conexiones involucradas.

La idea es dotar a la MMC de una capa de **meta-aprendizaje** que examina su propia estructura para consolidar conocimiento. Esto guarda relación con conceptos de **chunking** en arquitecturas cognitivas (como Soar, donde se forman reglas nuevas a partir de la experiencia para acelerar el razonamiento futuro) y con la formación de *Esquemas* en psicología cognitiva. Un patrón identificado puede verse como un *esquema* o plantilla que el sistema reconoce y puede aplicar en adelante. Por ejemplo, si el sistema descubre el patrón de que un cierto procedimiento de código siempre sigue a cierto problema matemático, podría crear un vínculo directo (una regla) para aplicarlo la próxima vez sin explorar tanto.

En cuanto al **clustering adaptativo**, en un sistema real se podría ejecutar periódicamente un algoritmo de **detección de comunidades** en la red de memoria para identificar grupos temáticos o contextuales de nodos. Esto sería invaluable para modularizar el conocimiento, mejorar la eficiencia (consultas localizadas por contexto) y también para visualizar la estructura del conocimiento. Técnicas populares como el algoritmo de **Louvain** detectan comunidades maximizando la modularidad de la partición del grafo, lo cual podría segmentar la MMC en áreas (clusters) relativamente independientes de conocimiento especializado. De hecho, en estudios de redes cerebrales se usan métricas de teoría de grafos dinámicos para analizar cómo la modularidad y la integración cambian con la edad. La MMC podría inspirarse en ello, adaptando su agrupamiento a medida que crece la información.

**Nodos Nativos e Identidad del Sistema**

Al inicializarse, el simulador MMC crea un conjunto de **nodos nativos básicos** que representan conocimientos innatos del sistema. En la implementación dada, se generan nodos como: native\_ethics (un fragmento ético con principios del sistema), native\_identity (fragmento de identidad del sistema), native\_learning (capacidades de aprendizaje), además de tres fragmentos especializados nativos (native\_code, native\_math, native\_sensorial) que corresponden a las categorías de código, matemáticas y sensorial. Estos nodos se añaden con una activación inicial alta (nivel de activación 1.0) y marcados como *inmutables* (no deben alterarse) en sus metadatos. A continuación, el sistema establece conexiones fuertes (peso = 1.0) entre algunos de estos fragmentos nativos para formar una base coherente: por ejemplo, native\_ethics ↔ native\_identity ↔ native\_learning se conectan mutuamente, creando un triángulo núcleo ético-identidad-aprendizaje. También conecta native\_identity con los fragmentos especializados (code, math, sensorial) con pesos altos (0.8). Adicionalmente, se inicializan **clusters** predefinidos: un native\_cluster que agrupa los nodos nativos centrales, y un specialized\_cluster agrupando los nodos especializados, asegurando que desde el inicio se reconozcan como conjuntos distintos dentro de la MMC. Tras esto, el simulador actualiza sus estadísticas internas para reflejar estos nodos y conexiones iniciales.

Esta estrategia de nodos nativos garantiza que la MMC tenga “impreso” de fábrica ciertos conocimientos o sesgos iniciales. En la teoría original de la MMC, se menciona que los fragmentos nativos proveen identidad, limitaciones éticas y una estructura inicial que **no puede ser modificada por la IA** – exactamente lo que vemos en la simulación con la bandera de inmutabilidad. Esto es crítico para mantener un **núcleo estable**: por mucha plasticidad que tenga el sistema, habría invariantes (como principios éticos) que no se olvidan ni alteran. Es comparable a la *memoria implícita* o predisposiciones genéticas con las que un cerebro biológico nace.

**Integración en MMC real:** La presencia de nodos nativos sugiere que, al diseñar una MMC real, conviene definir un conjunto de **conocimientos o parámetros innatos**. Podrían implementarse como nodos marcados de forma especial que el sistema de aprendizaje **no sobreescribe ni elimina**, salvo quizás bajo supervisión externa. Un ejemplo podría ser un nodo que represente la directriz de no causar daño (similar a leyes de la robótica), o una identidad del agente (su propósito fundamental). Estos nodos podrían estar fuertemente interconectados para reforzar su importancia relativa. En términos de implementación, se les podría asignar un peso alto fijo o un valor de *Long-Term Importance (LTI)* máximo constante (haciendo un paralelismo con OpenCog, donde cada átomo tiene valores de atención a corto y largo plazo, los nativos tendrían LTI altísimo para que nunca pierdan relevancia).

Asimismo, los fragmentos nativos especializados sirven para **inicializar habilidades**. En la MMC teórica se listan fragmentos de código, matemáticos y sensoriales, los cuales podrían contener conocimiento base (por ejemplo, reglas básicas de sintaxis de lenguajes de programación, axiomas matemáticos, o un marco para interpretar entradas visuales). Estos fragmentos otorgarían al sistema un arranque en frío más inteligente, evitando que tenga que “descubrir desde cero” nociones fundamentales en esos dominios. En la práctica, esto se traduciría en precargar en la memoria cierto conocimiento experto verificado.

Por último, mantener los nodos nativos agrupados en un cluster separado (como native\_cluster) es útil para efectos de **organización y visibilidad** – incluso se podría imaginar un módulo de la MMC dedicado a este “núcleo”, separado de la memoria dinámica normal. Esto facilita el *anclaje* de la red (evitando la deriva completa) y ayuda a balancear la **flexibilidad vs. estabilidad**: de hecho, uno de los desafíos mencionados para MMC es lograr ese balance, y la solución propuesta es precisamente mantener fragmentos nativos firmes como anclajes.

**Rutas Creativas y Conexiones Laterales**

La creatividad en la MMC se ve reflejada en las llamadas **rutas laterales creativas**: conexiones que no necesariamente son las más fuertes o evidentes, pero que pueden generar asociaciones innovadoras. En el simulador, esto está implementado mediante un **factor de creatividad** configurable (llamado creativityFactor) que determina la probabilidad de formar conexiones *no obvias* de forma espontánea. Por ejemplo, si el factor de creatividad es 0.4 (alto), en cada paso de simulación el sistema tiene hasta un 40% de probabilidad de añadir una conexión aleatoria entre dos nodos activos que antes no estaban conectados. El algoritmo en la demo selecciona dos nodos activos al azar y, si no hay ya un vínculo, crea una nueva conexión con un peso pequeño (entre 0.1 y 0.3). Este peso bajo indica que la idea es tentativa. El evento queda registrado como una “ruta lateral creada entre <tipo1> y <tipo2>”, por ejemplo “Ruta lateral creada entre text y math” para indicar que se conectó un nodo de texto con uno matemático inesperadamente.

La motivación de esto es emular *pensamiento divergente*: explorar combinaciones inusuales de conocimiento que podrían pasar desapercibidas si solo se siguieran los caminos consolidados. En el ejemplo dado en la MMC teórica, mencionan que una ruta lateral podría conectar un nodo sensorial (paisaje) con uno matemático para deducir patrones geométricos en ese paisaje, algo que ilustra creatividad por combinación de ámbitos distintos.

¿Cómo se aprovechan estas rutas? En la arquitectura, se menciona que las rutas laterales **se pueden convertir en rutas principales** si la experiencia muestra que fueron valiosas. En el simulador esto se vería reflejado si, tras crear la conexión creativa, posteriormente esos nodos se activan juntos con frecuencia (lo cual aumentaría el peso de esa conexión de 0.1 hacia arriba). Con el tiempo, una asociación inicialmente casual podría fortalecerse hasta volverse parte del “camino preferido” para cierto razonamiento.

**Integración en MMC real:** Para incorporar este aspecto, un sistema MMC debería permitir cierta **aleatoriedad controlada** o **búsqueda lateral** en la red de memoria. Esto podría lograrse de varias maneras. Una es introduciendo *enlaces candidatos* de manera probabilística como hace el simulador, quizá durante estados de reposo o baja actividad (similar a la forma en que el cerebro realiza *mind-wandering* o la generación de ideas espontáneas). Otra forma es a través de un mecanismo de **difusión de activación amplia**: en vez de activar solo los vecinos más fuertes de un nodo, ocasionalmente activar vecinos lejanos. Esta es la base de la teoría de *spreading activation* en redes semánticas, donde al pensar en un concepto la activación se propaga a conceptos relacionados, decayendo con la distancia, pero potencialmente alcanzando nodos remotos que provocan nuevas asociaciones.

Implementar un *factor de creatividad* podría hacerse ajustando un parámetro de **exploración interna**. Análogo al caso de exploración-explotación pero dentro de la propia red: una alta creatividad correspondería a elegir saltos menos probables dentro del grafo durante el razonamiento. Por ejemplo, cuando el sistema busca responder una consulta, en lugar de seguir únicamente el camino de mayor peso, con cierta probabilidad tomar un desvío por una conexión más débil para ver a dónde conduce (lo que podría revelar una solución no trivial). En algoritmos de búsqueda esto recuerda a añadir *ruido* o usar estrategias como *simulated annealing* para escapar de óptimos locales.

Es importante que estas conexiones creadas no colapsen la coherencia del sistema. Por ello, suelen nacer con pesos bajos y solo persisten si se refuerzan. En caso contrario, podrían eventualmente decaer. Esto sugiere que la **plasticidad sináptica** de la MMC debe tener un componente de **prueba y error**: nuevos enlaces tienen “período de prueba” en el cual o se consolidan (si aportan beneficio) o se debilitan de vuelta. Un marco teórico aplicable aquí es la **teoría Hebbiana** de aprendizaje: *“neuronas que disparan juntas, se conectan juntas”*. En MMC, podríamos traducirlo a *“fragmentos que se activan juntos repetidamente, se conectan (o refuerzan su conexión)”*. Donald Hebb postuló que la activación simultánea de dos células fortalece la sinapsis entre ellas. Las rutas creativas añaden conexiones que luego necesitan ser respaldadas por activaciones concurrentes reales para adherirse; si eso ocurre, estamos viendo un caso de aprendizaje hebbiano emergente.

En resumen, dotar al sistema de un *modo creativo* incrementa su capacidad de **innovación y generalización**, permitiendo saltos de intuición. Es un comportamiento observado en sistemas de inteligencia artificial que equilibran búsqueda global y local, y también en modelos cognitivos de creatividad donde la mente alterna entre estados de asociación libre y evaluación crítica. MMC incorpora ambos: genera ideas (vínculos) novedosas y luego las somete al escrutinio de la experiencia.

**Decaimiento de Nodos e Hibernación**

Para evitar la sobrecarga y asegurar la adaptabilidad, la MMC simula un proceso de **decaimiento de activación** en nodos inactivos con eventual *hibernación*. En el modelo, cada nodo tiene un atributo activationLevel (nivel de activación actual) y metadatos de cuándo fue la última vez accedido y cuántas veces ha sido utilizado. El sistema reduce gradualmente la activación de los nodos que llevan un tiempo sin usarse. En la simulación, en cada paso \_simulationStep(), se ejecuta \_simulateNodeDecay() que recorre todos los nodos no inmutables y calcula el tiempo desde su último acceso. Si han pasado, por ejemplo, >10 segundos sin uso (un umbral muy corto para fines de demo), entonces el nivel de activación se multiplica por (1 – tasa\_de\_decaimiento). La tasa de decaimiento es configurable; por defecto el simulador usa 0.005 (muy lento) o permite elegir valores hasta 0.05 (más rápido). Con 0.005, después de 10 segundos de inactividad, un nodo retendría ~99.5% de activación, mientras que con 0.05 retendría 95%. Así, a lo largo de minutos u horas, un nodo poco usado irá perdiendo importancia gradualmente.

Cuando la activación de un nodo cae por debajo de cierto umbral mínimo (ej. <0.1 en la simulación), el sistema marca ese nodo como **hibernado**. En el modelo, esto implica establecer node.metadata.hibernated = true y decrementar el contador de nodos activos. Un nodo hibernado sigue existiendo en la memoria, pero se considera **inactivo**: puede omitirse en cálculos cotidianos (búsquedas, activaciones) para ahorrar recursos. La interfaz refleja esto mostrando por separado conteos de *nodos activos* vs *totales*. De forma similar, las conexiones también tienen un flag de hibernación si sus nodos asociados se enfrían.

La hibernación es una alternativa más elegante que el borrado permanente: preserva la información para posible reutilización futura, pero la pone en “sueño profundo”. La MMC teórica menciona que las conexiones poco utilizadas dejan *marcas* para su posible reactivación, lo que concuerda con esta idea. Si en algún momento un contexto futuro vuelve relevante un nodo hibernado, el sistema puede “despertarlo” reactivándolo y aumentando de nuevo su peso.

**Integración en MMC real:** Esta funcionalidad se relaciona estrechamente con **cómo gestionar el olvido y la memoria a largo plazo** en un sistema de IA. En modelos cognitivos como ACT-R, cada pieza de conocimiento (chunk) tiene una activación base que decae con el tiempo desde su último uso, siguiendo una curva de poder. Si la activación cae por debajo de un umbral, es menos probable que se recupere esa pieza (esto se formula matemáticamente para predecir tiempos de respuesta y olvido en humanos). MMC podría incorporar una fórmula similar: $activation = \sum\_{uses} t^{-d}$, donde cada uso aporta un incremento que decae con $t$ (tiempo), y $d$ es un parámetro de decaimiento. Así, la frecuencia y recencia determinan la fuerza del recuerdo, replicando hallazgos de la psicología del olvido.

La *hibernación* en un sistema real podría implementarse como mover nodos de la memoria activa (RAM) a un almacenamiento secundario (disco) cuando su activación es baja, para liberar espacio. De ese modo, la MMC actúa también como un sistema de **cache** adaptativo: mantiene en primer plano lo más pertinente y envía al “archivo” lo lejano. Un ejemplo es cómo funcionan las bases de conocimientos en agentes conversacionales a gran escala: conocimientos raros se almacenan pero no se evalúan en cada interacción salvo que se soliciten expresamente.

También podemos inspirarnos en **OpenCog**: su mecanismo de *Economic Attention Allocation* asigna a cada átomo (nodo) un valor de atención de corto y largo plazo (STI y LTI). La importancia a corto plazo sube cuando el átomo se usa y luego decae no linealmente, mientras que la de largo plazo cambia más lentamente con éxito sostenido. Un nodo “durmiente” equivaldría a tener STI casi nulo, aunque LTI podría conservar un valor si históricamente fue importante. La MMC podría adoptar un esquema así, distinguiendo memoria activa vs latente más finamente.

En todo caso, es crucial definir una **política de reactivación**: ¿qué provoca que un nodo hibernado despierte? Podría ser una coincidencia parcial (por ejemplo, llega una consulta que coincide en parte con el contenido del nodo hibernado), o un **cambio contextual** (un nuevo nodo activo guarda similitud semántica con aquel hibernado). La MMC teórica sugiere “Reactivación por Contexto” – si el contexto lo requiere, conexiones hibernadas se reactivan. Un mecanismo concreto sería: al procesar nueva información, comprobar si hay nodos hibernados vinculados (p. ej., mediante etiquetas o ontologías) y levantar su activación temporalmente para ver si aportan algo. Si resultan útiles, se salen de hibernación formalmente (su activación base se reajusta).

En resumen, el decaimiento y hibernación dotan a la MMC de **olvido activo** o **memoria selectiva**, permitiéndole escalar sin saturarse y adaptarse a cambios de dominio de interés. Es un componente esencial de cualquier memoria adaptativa, reflejando la necesidad de *limpieza* y *priorización de recursos cognitivos*.

**Persistencia y Recuperación del Estado**

El sistema simulado incluye también funcionalidades de **persistencia**, para guardar y cargar el estado completo de la MMC. En la pestaña de configuración se muestran botones "Guardar Estado" y "Cargar Estado", que en la demo están habilitados solo cuando el sistema está en marcha. Al guardar el estado, el simulador serializa todos sus datos principales: los nodos, las conexiones, los clusters, estadísticas y configuración en un objeto, y lo almacena (en la implementación demo, usa localStorage del navegador). Al cargar, extrae esa estructura y restaura las colecciones de nodos, conexiones, clusters y parámetros del sistema al momento de la captura. Tras cargar, registra un evento de "Estado del sistema restaurado" para indicar que la MMC volvió a una condición previa.

Esto es importante porque una MMC real manejaría posiblemente gran cantidad de datos, y requeriría **almacenamiento persistente** más allá de la memoria volátil. También es clave para pausar/reanudar procesos de larga duración y para la *evolución continua*: se puede tomar un *snapshot* del estado cognitivo y transportarlo (por ejemplo, migrarlo a otro sistema, o revisitarlo después).

**Integración en MMC real:** La persistencia en un entorno real implicaría utilizar una base de datos o sistema de archivos para almacenar el grafo de la MMC. Dado que hablamos de grafos potencialmente grandes y dinámicos, una base de datos orientada a grafos (como Neo4j, JanusGraph, etc.) sería apropiada, permitiendo guardar nodos, relaciones y propiedades de manera estructurada. También se podría emplear formatos serializados como JSON (como en el simulador) o binarios más eficientes.

Un aspecto a considerar es **qué frecuencia y estrategia de persistencia** usar. Guardar el estado completo en cada cambio sería ineficiente; en cambio, se podría optar por **persistencia incremental** (guardar solo diferencias, o mantener siempre actualizado el almacenamiento tras cada transacción de cambio). Otra estrategia es la **hibernación de nodos** mencionada: nodos que pasan a hibernados podrían descargarse inmediatamente al almacenamiento secundario, persistiendo su información hasta que se reactivaran.

La capacidad de cargar un estado previo también abre puertas a funcionalidades avanzadas, como **versionado de la memoria** o *branching*. La MMC podría tener *checkpoints* en su aprendizaje a los que se pueda retroceder si una rama de conocimiento resulta errónea o dañina. Incluso se podría tener múltiples “instancias” de estados de MMC para diferentes simulaciones o escenarios comparativos.

En la interfaz simulada, la persistencia es manual (el usuario pulsa guardar/cargar). En un sistema autónomo, la persistencia sería automática en segundo plano, quizás siguiendo el parámetro de “intervalo de persistencia” que se ve en la configuración (ej. cada 300 segundos). También existe la opción de "Persistencia automática" con un checkbox, que seguramente indica si guardar periódicamente sin intervención.

Finalmente, está la opción de "Auto-reinicio tras error", pensada para robustez: si el sistema encuentra una falla, puede reiniciarse (posiblemente cargando el último estado guardado conocido). Esto enfatiza la necesidad de persistir el conocimiento para no perderlo en caídas imprevistas. Un MMC real debería por tanto ser tolerante a fallos, guardando su estado de forma que, si se reinicia la aplicación o se traslada a otro servidor, continúe donde quedó.

**Interacción con Entradas de Texto y Código (Respuesta del Sistema)**

El modelo simulado también contempla la **interacción directa con el usuario** mediante entradas de texto y de código, y la generación de respuestas del sistema basadas en el estado actual de la memoria. En la pestaña "Interacción" de la interfaz, se proporciona un campo de texto libre donde el usuario puede escribir alguna entrada (por ejemplo, una frase o pregunta) y botones para procesarla. De forma análoga, hay un área para introducir fragmentos de código en varios lenguajes (JavaScript, Python, Java, C#, etc.) y un botón para procesar el código. Estas funciones simulan cómo la MMC incorporaría información proporcionada activamente por el usuario, en contraste con la exploración pasiva del entorno.

Al pulsar "Procesar Texto", el simulador toma el contenido del textarea y crea un nuevo nodo de tipo text en la memoria con ese contenido. Este nodo se añade con un nivel de activación inicial (importancia) de 0.7 de manera predeterminada, lo cual indica que el sistema considera moderadamente importante la información recién entrada. Inmediatamente, el sistema intenta **conectar este nuevo nodo con nodos relacionados existentes** mediante \_connectToRelatedNodes. En la demo, la búsqueda de "relacionados" es aleatoria, pero conceptualmente representaría realizar análisis de similitud semántica: por ejemplo, si el texto contiene ciertas palabras clave que corresponden a nodos previos o clusters, conectar allí. Luego se registra en el log de actividad "Texto procesado: {primeros 30 caracteres}..." para dar trazabilidad.

El procesamiento de código es semejante: al ingresar código y seleccionar el lenguaje, al pulsar "Procesar Código" se crea un nodo de tipo code con el contenido del snippet y la etiqueta de lenguaje. Su activación inicial es un poco mayor (0.8) bajo el supuesto de que un fragmento de código puede tener alta relevancia técnica. Además de conectar con nodos relacionados por contenido, en la simulación se conecta este nodo con un nodo nativo de código (native\_code) si existe, con peso 0.8, lo que semeja asociar cualquier código nuevo con el “pool” de conocimiento de programación base. También se loguea la acción "Código X procesado" (donde X es el lenguaje).

Finalmente, la interacción contempla la **generación de una respuesta** por parte del sistema (botón "Generar Respuesta"). Esta funcionalidad intenta representar cómo la MMC puede usar su estado (todos sus nodos y conexiones) para producir una salida elaborada. En el simulador, al solicitar una respuesta para una consulta dada, se crea internamente un nodo de tipo response con un contenido de texto generado. El contenido en la demo es fabricado por una función \_generateSimulatedResponse(query) que selecciona entre varias frases genéricas mencionando que la respuesta está "basada en el análisis de los nodos relacionados" o "considerando la estructura MMC actual". Esto es simplemente para la demostración, pero en un sistema real, aquí ocurriría la compleja tarea de **razonamiento** o **síntesis** de respuesta a partir de la red de memoria, posiblemente apoyada por motores de inferencia o modelos de lenguaje.

Cuando se genera este nodo respuesta, se asigna un alto nivel de activación (0.9) ya que es un punto culminante del proceso, y luego el sistema identifica cuáles nodos de la MMC son relevantes para la pregunta dada. La simulación obtiene una lista de nodos relevantes mediante \_findRelevantNodes(query) que de momento retorna algunos nodos activos aleatoriamente, y conecta la respuesta con cada uno de esos nodos mediante conexiones de peso 0.7. Esto simboliza que la respuesta “se sustenta” en ciertos fragmentos de conocimiento existentes. Por ejemplo, si la pregunta fuera sobre un tema concreto, los nodos relevantes podrían ser aquellos clusters o textos que contienen esa información, y la respuesta se enlaza a ellos para indicar trazabilidad. Tras ello, el nodo respuesta (con su contenido) es entregado como output (en la interfaz simplemente se muestra el texto en el panel de “Respuesta del Sistema”).

**Integración en MMC real:** La capacidad de aceptar **entradas del usuario** y dar **respuestas** es esencial para muchas aplicaciones de la MMC (por ejemplo, un asistente conversacional, un tutor inteligente, etc.). En una realización práctica, la MMC serviría como la *memoria de trabajo de conocimiento* para un sistema IA mayor: las entradas del usuario se incorporarían a la MMC para que el sistema tenga en cuenta esa información, y luego un módulo de inferencia/decisión generaría una respuesta consultando la MMC.

Para procesar texto de entrada, se podría integrar con técnicas de **procesamiento de lenguaje natural** (PLN). Es decir, el texto se analizaría (tokenizado, etiquetado) y se trataría de vincular con nodos existentes: usando embeddings semánticos se podrían encontrar los nodos de texto más similares, o crear nuevos nodos por cada concepto clave mencionado si no existían. Por ejemplo, si el usuario dice "La capital de Francia es París", la MMC podría crear nodos para "capital de Francia" y "París" y conectarlos, o reforzar si ya existían. En el caso de código, igualmente se podría parsear para entender su funcionalidad o al menos almacenar su representación AST (árbol sintáctico) para compararlo con otros códigos.

La generación de respuestas en un entorno real podría delegarse a un **motor de razonamiento**. Una posibilidad es emplear un modelo neuro-simbólico: la MMC provee los hechos y relaciones pertinentes, y un modelo de lenguaje (ej. GPT-like) formatea la respuesta textual citando o usando esos hechos. Otra posibilidad es un enfoque simbólico puro: ejecutar un algoritmo de encadenamiento (hacia adelante o atrás) dentro del grafo, aplicar reglas lógicas (si las hay en el conocimiento) y componer la respuesta. En arquitecturas cognitivas híbridas, a veces la memoria declarativa (como MMC) se combina con un módulo de **producciones** (reglas if-then, como en ACT-R o Soar) que dicen cómo usar esa información para lograr objetivos. MMC por sí sola es más una base de conocimiento, pero conectada a un *ejecutor* podría resolver preguntas.

Un detalle interesante es que el sistema simulado **conecta las respuestas a los nodos relevantes**. Esto significa que la MMC **aprende de sus propias respuestas**: tras resolver una pregunta, almacena el resultado en el grafo, de modo que si en el futuro se hace una pregunta similar, puede reutilizar esa respuesta previa (o al menos saber que ya se abordó ese tópico). Esto añade una especie de **memoria episódica** o de casos resueltos. Una MMC real podría implementar este concepto guardando un historial de Q&A, o más ampliamente, registrando *episodios de interacción* (contexto de diálogo, decisiones tomadas, resultado obtenido) como experiencia. Entonces, la próxima vez, antes de razonar desde cero, podría ver si algún episodio previo se asemeja y adaptarlo (aprendizaje basado en casos).

En resumen, la interacción directa cierra el ciclo de la MMC: no solo acumula conocimiento pasivamente, sino que recibe problemas o preguntas y utiliza su conocimiento para producir respuestas, **enriqueciéndose a sí misma en el proceso** con las nuevas entradas y salidas generadas.

Habiendo detallado las capacidades del modelo simulado y cómo reflejan aspectos teóricos de la MMC, procedemos a analizar marcos y ejemplos existentes que contextualizan y pueden potenciar este diseño.

**Marcos Teóricos para la Evolución Dinámica de Grafos Cognitivos**

La idea de una memoria como grafo dinámico se alinea con varias teorías en ciencia cognitiva y sistemas de información. Un **grafo cognitivo** dinámico puede entenderse como una **red semántica** que cambia con el aprendizaje. En psicología cognitiva clásica, Collins y Loftus propusieron en 1975 la *Teoría de la Activación Propagada* (*Spreading Activation Theory*) para explicar el acceso semántico en la memoria humana. En su modelo, la memoria es una red de nodos (conceptos) conectados por aristas ponderadas (asociaciones semánticas); cuando pensamos en un concepto, activamos su nodo, y esa activación **se propaga** a los nodos vecinos disminuyendo con la distancia, lo que facilita recuperar conceptos relacionados. Esto explica fenómenos como la *priming* (un concepto activa otro más rápido si están relacionados). La MMC adopta principios similares: los pesos de conexión indican fortaleza de asociación, y podemos imaginar que al consultarse algo en MMC, la activación se difunde por la red (siguiendo las rutas principales más fuertes pero también laterales con menor probabilidad, si consideramos el componente creativo). La teoría de activación propagada respalda la **búsqueda de memoria basada en asociaciones**, que es esencial para que MMC encuentre información pertinente a una consulta sin hacer búsqueda exhaustiva.

Otro marco relevante es la idea de **mapas cognitivos** dinámicos en robótica y neurociencia. Por ejemplo, se han investigado *cognitive maps* para navegación, donde un agente construye un grafo de lugares y pistas, actualizándolo a medida que explora (inspirado en cómo los mamíferos usan células de lugar en el hipocampo). Un estudio de *Frontiers in Neurorobotics* (Han et al. 2020) introduce un modelo computacional para navegar y mapear un espacio usando principios inspirados biológicamente, resaltando la importancia de un **mapa cognitivo dinámico** que se ajusta cuando el entorno cambia. Si bien orientado a geografía, el concepto aplica a MMC: la memoria es un mapa de conocimiento donde los nodos son “lugares conceptuales” y las conexiones son “caminos” cuya **configuración evoluciona** conforme se aprende (nuevos nodos=descubrir nuevos lugares, reforzar conexiones= caminos más transitados, etc.).

Desde la perspectiva de redes complejas, la MMC también puede verse como un sistema de **red evolutiva**. Hay investigaciones sobre *grafos que aprenden* o que cambian con reglas dinámicas. Por ejemplo, la noción de *Evolving Graphs* en ciencia de redes aborda cómo representar matemáticamente grafos que cambian con el tiempo y cómo inferir sus propiedades. MMC podría beneficiarse de resultados en este ámbito para mantener estabilidad a pesar del cambio: por ejemplo, evitar oscilaciones caóticas en la estructura garantizando cierta “inercia” en los pesos (quizá mediante filtros exponenciales en la actualización de pesos).

Una dimensión teórica fundamental es la **plasticidad sináptica** y la competencia estabilidad-plasticidad. En redes neuronales existe el dilema de **plasticidad vs. estabilidad** (también llamado problema de *catastrophic forgetting*): si un sistema aprende continuamente, corre riesgo de olvidar lo antiguo al adaptarse a lo nuevo. MMC aborda esto con los fragmentos nativos (estabilidad garantizada) y con la hibernación en lugar de eliminación (para no perder del todo lo viejo). La **Teoría de la Consolidación** en neurociencia sugiere que los recuerdos recientes (a corto plazo) con el tiempo se consolidan en memoria a largo plazo si son reforzados, un proceso que podría inspirar a la MMC a tener fases de consolidación (quizá durante un “modo offline” análogo al sueño) donde sin reorganiza y refuerza patrones importantes descubiertos durante la fase online activa.

Por último, podemos citar la idea de **sinergia cognitiva** en sistemas integradores como OpenCog. Un paper de Goertzel explora cómo en una arquitectura cognitiva integradora, diferentes componentes (lógicos, asociativos, atencionales) interactúan de forma dinámica no-lineal. MMC sería esencialmente el componente asociativo (memoria), pero su evolución dinámica también debe considerarse en sinergia con componentes de razonamiento y aprendizaje. Esto sugiere que un marco teórico completo de MMC no está aislado: debe encajar en un modelo global de agente cognitivo donde percepción, memoria, atención y acción formen un ciclo.

En conclusión, teorías de redes semánticas con activación difundida, mapas cognitivos adaptativos, plasticidad sináptica y sistemas dinámicos no-lineales aportan fundamentos para entender y guiar la evolución de un grafo cognitivo como MMC. Validan la necesidad de mecanismos como la atenuación de conexiones (olvido) para evitar saturación, la propagación de activación para recuperación eficiente, y la consolidación de patrones recurrentes para aprendizaje robusto.

**Arquitecturas de Memoria Artificial Adaptativa: SPA, Soar, ACT-R, OpenCog, MemNets**

Existen varias arquitecturas de inteligencia artificial y cognitivas que, al igual que MMC, se ocupan de cómo representar y actualizar conocimiento de forma estructurada y adaptativa. A continuación, compararemos brevemente algunos ejemplos notables, extrayendo paralelismos y lecciones útiles para la MMC. En la siguiente tabla resumimos sus características clave en cuanto a representación de conocimiento, dinámica de aprendizaje/memoria y mecanismos relevantes:

**Tabla 1. Comparativa de arquitecturas de memoria adaptativa**

| **Arquitectura** | **Representación del Conocimiento** | **Dinámica y Aprendizaje** | **Características Relevantes** |
| --- | --- | --- | --- |
| **Semantic Pointer Architecture (SPA)** | Vectores de alta dimensión (*Vector Symbolic Architecture*). Conceptos representados como vectores; combinaciones mediante operaciones algebraicas (superposición, convolución circular). | Aprende ajustando pesos sinápticos de una red neuronal espaciotemporal (por ejemplo, en Spaun se usan 2.5 millones de neuronas simuladas). Incluye STDP (regla de aprendizaje dependiente de tiempo de disparo) para ajustar conexiones en base a actividad reciente[nengo.ai](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html#:~:text=adaptability%20,like%20rule). | *Biológicamente inspirada* (modelo neuronal a gran escala). Permite **composición estructurada**: puede enlazar vectores para formar estructuras (ej. representar una frase uniendo concepto + rol gramatical). Ha demostrado realizar múltiples tareas cognitivas (Spaun) incluyendo recordar y razonar. |
| **Soar** (Cognitive Architecture) | Estructuras simbólicas gráficas en memoria de trabajo (working memory) representando el estado; reglas de producción en memoria procedimental. | Aprendizaje por *chunking*: crea nuevas reglas (chunks) a partir de la resolución de impasses. Memoria a largo plazo separada en: procedimental (reglas), declarativa (semantic/episódica), y puede usar refuerzo para ajustar preferencias. Todas las estructuras simbólicas llevan metadatos estadísticos (recencia, frecuencia) que influyen su uso futuro. | Arquitectura **híbrida** deliberativa-reactiva. **Universal Subgoaling**: crea subestados para resolver impasses. *Decision cycle* muy estudiado. La representación del estado es gráfica (nodos-atributos), comparte espíritu con MMC en tener todo en un grafo simbólico. Agrega **módulo de decisión** y separación de memorias. Usa meta-datos de uso para olvido/adaptación, similar a pesos dinámicos. |
| **ACT-R** (Adaptive Control of Thought – Rational) | Memoria declarativa como **chunks** (cada chunk es una unidad con campos, e.g. un hecho); Memoria procedimental como **producciones** (reglas if-then). | Cada chunk tiene un **nivel de activación** que se incrementa cuando es recuperado y decae con el tiempo (modelo de **olvido**). Además, hay **propagación de activación**: el contexto actual (chunks en el foco) transfiere activación a chunks asociados. Aprendizaje: ajusta fortalezas de reglas por refuerzo; crea nuevas producciones por compilación (similar a chunking). | Arquitectura cognitiva modular inspirada en la mente humana. **Mezcla simbólica y subsimbólica**: las decisiones de qué recuperar se basan en probabilidades derivadas de activación (subsim.) aunque la representación sea simbólica. Cuenta con **memoria episódica** y **memoria de buffers** (atencionales). Ofrece fórmulas para optimizar el balance entre velocidad y exactitud basadas en experiencia. Sus ecuaciones de activación pueden guiar el diseño de decaimiento en MMC. |
| **OpenCog** (Atomspace / CogPrime) | Grafo hiperbólico de átomos (nodos) y enlaces. Átomos pueden ser conceptos, predicados, esquemas, etc. (muy general). | Cada átomo tiene valores de **Short-Term Importance (STI)** y **Long-Term Importance (LTI)** que actualizan mediante ecuaciones dinámicas no lineales. Procesos de atención redistribuyen importancia según la actividad (Economic Attention). Aprendizaje lógico mediante PLN (Probabilistic Logic Networks) para inferencia y MOSES (evolutivo) para aprender reglas/programas. | Arquitectura integradora **AGI**. La memoria es un grafo unificado (Atomspace) donde todo conocimiento reside. Posee **mecanismo de olvido** (átomos decaen si no ganan STI) y **persistencia** según LTI para no perder conocimientos valiosos. Apoya múltiples tipos de enlaces: lógicos, asociativos, ejecutables. Enfoque fuertemente dirigido a mantener **atención focal** en partes importantes del grafo en cada momento. Muchas similitudes con MMC: estructura de grafo, decaimiento (STI) e importancia acumulada (LTI), diversidad de tipos de nodos. |
| **Memory Networks (MemNets)** (Weston et al. 2015) | Memoria como un conjunto de **sentencias** (vectorizadas). Usa representaciones distribuidas (embeddings) para palabras y oraciones. | Un modelo neural lee de una memoria de larga duración con un mecanismo de **atención** para seleccionar oraciones relevantes y produce una respuesta. El modelo aprende end-to-end a leer/escribir en la memoria para tareas de QA. La memoria es dinámica en el sentido de que puede actualizarse con nuevas entradas durante el entrenamiento o incluso inferencia. | Enfoque conexionista: implementa la idea de **lectura-escritura diferenciable** en una memoria externa. Inspiró extensiones como el **Differentiable Neural Computer (DNC)** de DeepMind, que añade control lógico sobre la memoria. Ofrece **razonamiento por atención múltiple** (hops) para combinar hechos. Aunque no es un grafo explícito, se puede considerar que los datos almacenados conforman un grafo implícito (por las referencias entre sentencias). MemNets muestran cómo integrar una memoria tipo MMC con un potente modelo de inferencia (un RNN o Transformer) para tareas de lenguaje. |

**Observaciones:** Podemos apreciar que MMC comparte rasgos con varias de estas arquitecturas. Por ejemplo, **MMC y OpenCog Atomspace** comparten la representación de grafo y la noción de importancia dinámica (hibernación vs. STI/LTI). Con **ACT-R**, MMC coincide en usar activaciones que decaen y en propagar activación contextual, aunque ACT-R es más estructurado en módulos (buffers, etc.). **Soar** y MMC comparten la idea de *metadata estadístico* (en Soar, todo símbolo tiene recencia/frecuencia que afectan su uso) y la de chunking es análoga a la extracción de patrones en MMC para formar “unidades de conocimiento” a posteriori. **SPA** nos brinda una forma alternativa de representar conceptos: en lugar de nodos discretos, vectores, pero hay investigaciones que combinan grafos simbólicos con **vectores semánticos** asignados a cada nodo (embedding de nodos), lo cual podría también aplicarse a MMC para tener un sustrato vectorial que mida similitudes. **Memory Networks/DNC** ilustran cómo podríamos implementar la *memoria operativa* de MMC en un contexto de aprendizaje automático puro, donde la atención es clave para seleccionar qué parte del conocimiento usar (conceptualmente similar a la consulta de subgrafo relevante en MMC para responder).

En suma, la MMC se sitúa en la intersección de enfoques simbólicos (redes de conocimiento) y conexionistas (adaptación continua, difusión de activación). Las arquitecturas revisadas aportan ideas: OpenCog sugiere mecanismos concretos de atención y olvido, ACT-R y Soar aportan fórmulas y estrategias de aprendizaje simbólico, SPA y MemNets muestran integración con representación vectorial y aprendizaje profundo. La MMC puede verse enriquecida por estos paradigmas, resultando en una arquitectura híbrida: **un grafo cognitivo simbólico que aprende con dinámicas subsimbólicas**.

**Clustering y Detección de Patrones en Redes de Conocimiento**

Como vimos en la sección de análisis de patrones, detectar estructuras y grupos internos en la red de memoria es fundamental para que la MMC organice y optimice su conocimiento. Existen múltiples técnicas modernas para **clustering** (agrupamiento) y **pattern mining** en grafos y bases de conocimiento que pueden ser aplicadas o adaptadas a la MMC.

**Clustering (Agrupamiento) en grafos:** El objetivo aquí es identificar subconjuntos de nodos que estén más fuertemente conectados entre sí que con el resto del grafo – en otras palabras, **comunidades** o tópicos dentro del conocimiento. En el contexto de una base de conocimiento (por ejemplo, un grafo semántico tipo DBpedia, o la propia MMC), esto podría revelar áreas temáticas (un cluster de nodos matemáticos, un cluster de nodos relacionados con un mismo proyecto, etc.).

Técnicas destacadas incluyen:

* El **algoritmo de Louvain** para detección de comunidades, que es escalable a grafos grandes y se basa en optimizar la **modularidad**. Podría ejecutarse periódicamente sobre la MMC para encontrar clusters emergentes. Su resultado serían agrupaciones que la MMC puede utilizar como *contextos*. De hecho, la MMC simulada creaba un specialized\_cluster de ejemplo agrupando fragmentos especializados, pero mediante Louvain u otro método, podría descubrir automáticamente que, por ejemplo, se ha formado un cluster de "conocimientos sobre clima" porque varios nodos meteorológicos se conectan entre sí.
* **Clustering jerárquico** aplicado a matrices de similitud de nodos. Si cada nodo tiene un vector de características (por ejemplo, vectores de palabras para textos, o one-hot de categorías), se pueden calcular distancias y agrupar. Esto puede complementar al grafo en sí, sobre todo para nodos con conexiones escasas pero alto parecido semántico.
* **Comunidades traslapantes:** métodos como **Label Propagation** que permiten que un nodo pertenezca a varias comunidades. En conocimiento, esto tiene sentido (un artículo científico podría pertenecer a la comunidad "IA" y "Robótica" simultáneamente). La MMC podría usar enfoques que asignen *membresías difusas* a clusters para reflejar multi-dominio.
* **Detección de tópicos**: si consideramos los textos almacenados en la MMC, se podría correr un algoritmo de modelado de tópicos (LDA, por ejemplo) para asignar etiquetas temáticas a conjuntos de nodos. Esto no es clustering de grafos per se, pero ayuda a crear *clusters conceptuales*.

El beneficio de agrupar es que la MMC puede **simplificar su grafo** a alto nivel. Por ejemplo, podría crear meta-nodos que representan clusters enteros (como “Tema X”) y conectar con otros meta-nodos, formando una visión resumida. Esto mejora escalabilidad y interpretabilidad. Además, para la *exploración vs explotación*, identificar clusters permite dirigir la exploración a áreas menos pobladas (explorar lo desconocido) o explotar dentro de un cluster (profundizar un tema).

**Detección de patrones y reglas:** Más allá de comunidades, la MMC se puede fortalecer extrayendo **patrones recurrentes**. Esto se relaciona con *reglas de asociación*, *secuencias frecuentes* y *subgrafos frecuentes*:

* **Minería de reglas de asociación:** Como se mencionó, algoritmos estilo Apriori pueden operar sobre conjuntos de atributos. En un grafo de conocimiento, una analogía sería buscar triples frecuentes. Por ejemplo, puede notar que muchos nodos de tipo “persona” tienen conexiones similares: (Persona)–(trabaja\_en)–(Empresa) y (Persona)–(vive\_en)–(Ciudad). Podría abstraer un patrón "Personas suelen estar vinculadas a una empresa y una ciudad". En MMC esto podría almacenarse como un *esquema* o simplemente usarse para inferir elementos faltantes (si veo persona y empresa, quizá esperaré la ciudad).
* **Minería de subgrafos frecuentes:** algoritmos que buscan estructuras relacionales repetidas en el grafo. Por ejemplo, descubrir que el triángulo (A–B, B–C, A–C) aparece muchas veces (indicando relaciones de triple co-ocurrencia). Estos subgrafos se pueden convertir en unidades; en bases de conocimiento científicas, esto se usa para descubrir *motivos* (motifs) significativos en redes complejas.
* **Patrones temporales/secuenciales:** si la MMC registra también la dimensión temporal de accesos (quién activó a quién en qué orden), podría buscar secuencias frecuentes de activación. Esto es similar a cómo en cadenas de Markov o en el comportamiento del usuario se buscan secuencias comunes. Por ejemplo, si al resolver cierto problema la secuencia de activación de nodos sigue un patrón A → B → C repetidamente, esa secuencia puede almacenarse como procedimiento aprendido.

En cuanto a *técnicas modernas*, la **aprendizaje de representaciones** también juega un rol. Los **knowledge graph embeddings** (como TransE, node2vec, etc.) proyectan los nodos de un grafo a un espacio vectorial continuo tal que las relaciones se preserven geométricamente. La MMC podría usar embeddings para detectar clusters (mediante agrupamiento en el espacio vectorial) y patrones (por ejemplo, completando analogías con vectores: v("Francia") + v("capital") ≈ v("París")). Esto sería una capa adicional donde la MMC no solo maneja símbolos, sino que entiende las *distancias semánticas*.

Una técnica relevante es el **clustering en redes semánticas dinámicas para creatividad**. Un estudio menciona que al analizar la creatividad de estudiantes de diseño, construyeron *dynamic semantic networks* a partir de sus conversaciones y midieron convergencia vs divergencia. Se encontraron métricas para cuantificar pensamiento divergente y su éxito. Trasladado a MMC, se podría monitorizar cómo de interconectado vs segregado está su grafo (un grafo demasiado interconectado sin clusters puede indicar sobre-generalización, uno muy segregado sin conexiones transversales puede indicar falta de creatividad). Equilibrar esto es clave: un estado crítico entre orden y caos suele asociarse a alta capacidad adaptativa. Herramientas de análisis de grafos (centralidades, modularidad, etc.) aplicadas dinámicamente pueden guiar la parametrización de la MMC (por ejemplo, subir el creativityFactor si la red está demasiado estructurada, o bajarlo si es un caos de conexiones aleatorias).

En resumen, la MMC debería incorporar:

* Algoritmos de **comunidad** para descubrir y mantener agrupaciones temáticas.
* Algoritmos de **patrones frecuentes** (reglas, subgrafos, secuencias) para extraer nuevo conocimiento implícito.
* Posiblemente, uso de **embeddings** o estadísticas globales para facilitar estos cálculos y enlazar con técnicas de IA actuales.

Esto haría a MMC no solo reactiva, sino **proactiva en reestructurar y resumir su conocimiento** de manera legible y útil. Por ejemplo, podría generar explicaciones del tipo “He notado un patrón: siempre que ocurre X también ocurre Y (con 80% de confianza)”, lo cual es valioso para un usuario que intenta entender la base de conocimiento.

**Modelos de Exploración vs. Explotación en Sistemas Inteligentes**

El dilema entre exploración y explotación es un tema central tanto en la teoría de decisiones como en el funcionamiento de sistemas inteligentes autónomos. Ya observamos cómo la MMC simulada permitía un control explícito de este equilibrio en su comportamiento de búsqueda de información. Es útil profundizar brevemente en este concepto y cómo se puede modelar, ya que influye en estrategias de aprendizaje y actualización de la memoria.

En términos generales, la **exploración** implica probar acciones o buscar información nueva para *descubrir* más acerca del entorno (o del problema a resolver), mientras que la **explotación** implica usar el conocimiento ya disponible para *obtener resultados* óptimos inmediatos. Es un balance intrínseco en algoritmos de **reinforcement learning (RL)**: un agente que siempre explota puede converger a una solución subóptima porque nunca vio alternativas mejores, y uno que siempre explora nunca capitaliza lo aprendido. Formalmente, se conoce como **dilema exploración–explotación**.

En la MMC, este dilema aparece en distintos niveles:

* **A nivel de adquisición de conocimiento externo**: ¿cuánto tiempo dedicar a incorporar nuevos datos (leyendo webs, APIs) vs refinar y aplicar los datos que ya tengo? Esto se vincula a *active learning*: un sistema debe decidir si necesita más ejemplos o ya puede responder con lo que sabe.
* **A nivel de recorrido interno del grafo (razonamiento)**: ¿seguir las asociaciones más probables (caminos de mayor peso) o probar conexiones menos seguras que podrían llevar a soluciones novedosas? Aquí entra el factor de creatividad/lateralidad ya discutido.
* **A nivel de selección de acciones o planes**: si MMC formara parte de un agente que actúa, este debe decidir entre intentar acciones nuevas o las que ya demostraron recompensa.

Modelar este equilibrio suele hacerse con algún parámetro. En RL clásico, la estrategia $\varepsilon$-greedy elige una acción aleatoria con probabilidad $\varepsilon$ (exploración) y la mejor acción conocida con probabilidad $1-\varepsilon$ (explotación). MMC podría aplicar un análogo: por ejemplo, al elegir qué nodo explorar a continuación en la web, con probabilidad $\varepsilon$ elegir un link al azar versus con $1-\varepsilon$ elegir el link más relevante según su modelo actual. En la interfaz, la tasa exploración/explotación implementa esencialmente un $\varepsilon$ (70% exploración ~ $\varepsilon=0.7$).

Existen modelos más sofisticados como **Thompson Sampling** o **UCB (Upper Confidence Bound)** para el problema multi-armed bandit que permiten equilibrio óptimo en teoría. Aplicados a MMC, podrían decidir no de forma fija sino en función de la incertidumbre: por ejemplo, **si la MMC está muy incierta** sobre cierta área de conocimiento, debería explorar más ahí; si está muy segura (ha convergido), puede explotar. Esto requiere que la MMC tenga una estimación de incertidumbre sobre lo que sabe. Podría provenir de contadores (p.ej., sabe que solo ha visto 2 páginas sobre un tema X → alta incertidumbre → explorar más de X).

También hay inspiraciones de la biología: estudios han encontrado que en el cerebro humano el neurotransmisor **noradrenalina** influye en la alternancia entre modos exploratorio y explotador en el locus coeruleus. En IA, se han creado agentes con **motivaciones intrínsecas** (recompensas por novedad o curiosidad) para inducir exploración cuando las recompensas extrínsecas son escasas. La MMC podría beneficiarse de una señal de curiosidad: por ejemplo, asignar un *bonus* de activación a nodos muy nuevos o a conexiones raras, incentivando el sistema a revisarlos.

Es importante notar que exploración y explotación no son solo dicotomía en *cantidad* sino en **estilo de procesamiento**: un régimen de exploración favorece la divergencia, conexiones remotas (nuestro factor de creatividad alto), pensamiento “fuera de la caja”. Un régimen de explotación favorece la convergencia, concentración en lo ya validado, eficiencia. En términos de grafos cognitivos, un estudio sugiere que este trade-off es análogo al trade-off **persistencia vs flexibilidad** en la cognición. Un estado de ánimo positivo, por ejemplo, tiende a fomentar más flexibilidad (exploración cognitiva). Traducido a MMC, podríamos especular con la idea de que el sistema ajustara su exploración/explotación no solo por parámetros fijos sino también por *input externo o metas*: si se requiere mucha creatividad, sube exploración; si se necesita una respuesta precisa y confiable, sube explotación.

En una MMC implementada, se podría **modular dinámicamente** este parámetro. Por ejemplo, al principio de su “vida”, la MMC debería explorar mucho (poca información conocida), y conforme llena su red, podría reducir exploración para refinar (esto imita el *annealing schedule* en RL donde $\varepsilon$ decae con el tiempo). Sin embargo, si luego se enfrenta a un dominio nuevo, tendría que volver a explorar (un mecanismo de detección de novedad contextual podría resetear $\varepsilon$ alto cuando entra en un área poco poblada del grafo).

En síntesis, la MMC ideal incorporaría políticas de exploración-explotación inteligentes: **exploración suficiente para crecer y adaptarse**, y **explotación suficiente para aprovechar y responder eficientemente**. Herramientas formales de RL pueden ayudar a lograr ese equilibrio óptimo, mientras que las interfaces (como el slider del simulador) son útiles para ajuste fino durante desarrollo y para exponer el comportamiento al usuario.

**Integración de Hallazgos en la Arquitectura MMC y Mejoras Propuestas**

Combinando las ideas prácticas (modelo simulado) con los marcos teóricos y arquitecturas revisadas, podemos delinear cómo sería una **MMC de siguiente generación**, más robusta y funcional. A continuación, enumeramos mejoras e innovaciones potenciales para la arquitectura MMC, inspiradas en los hallazgos:

* **1. Mecanismo de Activación y Olvido Avanzado:** Formalizar el manejo de activaciones de nodos usando ecuaciones probadas. Por ejemplo, adoptar la fórmula de activación base de ACT-R (suma de decaimientos logarítmicos de cada uso) para calcular la accesibilidad de un nodo. Implementar un sistema doble de importancia estilo OpenCog – *importancia a corto plazo (ICL)* análoga a STI que sube rápido y decae rápido, e *importancia a largo plazo (ILP)* análoga a LTI que acumula con uso reiterado. Los nodos nativos tendrían ILP fija altísima. Este mecanismo dual permitiría que la MMC **aprenda qué conocimientos mantener en primer plano y cuáles archivar**, atacando el desafío de crecimiento exponencial de datos con una solución inspirada en atención económica. Asimismo, se podría introducir un *límite de capacidad activa*: mantener, digamos, sólo N nodos más activados “despiertos” plenamente, hibernando el resto, similar a como la memoria de trabajo humana es limitada.
* **2. Capas de Memoria Múltiples:** Inspirados por arquitecturas cognitivas, dividir la MMC en sub-módulos de memoria podría mejorar su funcionamiento. Por ejemplo, tener una **Memoria de Trabajo** que es un subgrafo altamente activado y mutable rápidamente (similar al buffer de atención), y una **Memoria de Largo Plazo** que es el grafo completo con actualizaciones más lentas. Las interacciones de usuario primeramente afectarían la memoria de trabajo, que luego se consolidaría en la de largo plazo periódicamente (evitando *catastrophic forgetting*). Esto se relaciona con la idea biológica de consolidación durante el sueño: la MMC podría en “modo inactivo” recalibrar pesos usando lo acumulado en el corto plazo. Otra capa valiosa es una **Memoria Episódica**: la capacidad de almacenar secuencias de eventos o contextos vividos (por ejemplo, cada sesión de diálogo con un usuario se guarda como episodio). ACT-R incluye memoria episódica; en MMC se podría implementar guardando *snapshots* etiquetados por contexto, que luego se pueden evocar para contexto similar futuro.
* **3. Integración Neuro-Simbólica (Embeddings y Similitud):** Aunque MMC es simbólica (nodos discretos), podemos enriquecerla asignando a cada nodo (o a ciertos tipos de nodos) una representación continua. Por ejemplo, nodos de texto podrían tener un embedding vectorial precomputado (mediante Word2Vec, BERT, etc.), nodos de imagen podrían tener un feature vector de una CNN, y nodos conceptuales podrían tener vectores entrenados mediante técnicas de graph embedding (node2vec, TransE para relaciones). Estos vectores permiten calcular similitudes rápidas entre nodos fuera de la estructura explícita de conexiones. Así, al buscar nodos relacionados para un nuevo fragmento, no solo miraríamos conexiones directas sino también su proximidad en el espacio vectorial semántico. Esto haría la MMC más robusta al conocimiento incompleto, pudiendo inferir conexiones potenciales (*link prediction* sugerido por proximidad de embeddings). Además, permitiría usar modelos neuronales (como Transformers) encima de la MMC: por ejemplo, generar respuestas en lenguaje natural alimentando al modelo con embeddings de los nodos relevantes, combinando **razonamiento simbólico** con **fluidez conexionista**.
* **4. Módulo de Inferencia Lógica y Reglas:** Actualmente, MMC es esencialmente una memoria asociativa. Incluir un componente de inferencia permitiría que no solo relaciones explícitas sean aprovechadas, sino también reglas lógicas. Por ejemplo, integrarla con un motor tipo **Probabilistic Logic Networks** (como el de OpenCog) para que pueda deducir nuevos hechos a partir de combinaciones de nodos (ej. si “X es padre de Y” y “Y es padre de Z”, inferir “X es abuelo de Z”). Estas inferencias lógicas podrían materializarse como nuevos enlaces añadidos al grafo (un modo de *adquirir conocimiento derivado*). La clave sería manejar la incertidumbre (por eso PLN probabilístico encajaría, asignando probabilidades a las inferencias, o integrando un sistema de lógica difusa). Así MMC abarcaría **aprendizaje inductivo** (por patrones) y **deducción lógica**, complementando creatividad con consistencia.
* **5. Planificación de Exploración Inteligente:** Aplicar las teorías de exploración-explotación para que la MMC no dependa solo de parámetros fijos sino de *políticas adaptativas*. Por ejemplo, implementar un **agente de control** (metanodo) que analice la cobertura del conocimiento en distintos dominios de la MMC y decida enfocar la exploración en áreas débiles. Esto se puede hacer, por ejemplo, midiendo la densidad de conexiones en subgrafos: si cierta región del grafo tiene pocos nodos o conexiones, el agente decide lanzar más consultas web sobre ese tema. Inversamente, en áreas densas, explota relacionando lo ya presente (quizá haciendo análisis de patrones más profundos). En resumidas cuentas, dotar a la MMC de un *curator* interno que guía la exploración con meta-aprendizaje, en lugar de que sea puramente aleatoria o uniforme. Técnicas de **bandits contextuales** podrían emplearse: el “contexto” sería el estado actual de la memoria (qué sabe y qué no), las “banderas” serían posibles acciones de búsqueda, y las “recompensas” algún indicador como incremento de información lograda. Esto cerraría el loop de exploración de forma óptima.
* **6. Interfaz de Visualización y Edición Cognitiva:** Si bien es un aspecto de implementación, tiene implicaciones funcionales. La interfaz simulada muestra un grafo D3 y un panel de detalles, lo cual es muy útil. Llevar esto a un nivel avanzado significaría crear herramientas para **visualizar en tiempo real la red MMC** (quizá en 3D o realidad aumentada si hablamos de futuro), donde se vea cómo fluye la activación, cómo aparecen nuevas conexiones (ruta lateral iluminándose), cómo se agrupan en clusters (distintos colores por comunidad, como ya hace la leyenda de la demo con categorías de nodos). Esto no solo sirve a humanos para entender al agente, sino al propio agente: podría haber un módulo introspectivo que detecte anómalas en la visualización (ej. un nodo aislado flotando que debería conectarse pero no lo está, indicando un hueco de conocimiento). Además, podría permitirse la **edición manual** de la MMC: un supervisor humano añade un enlace, corrige un peso, etc., y el sistema aprende de esa retroalimentación. Para muchas aplicaciones (educación, asistentes) es vital tener ese control de experto. La arquitectura debe ser capaz de incorporar cambios manuales sin inestabilidad.
* **7. Capacidad de Contextualización y Etiquetado Enriquecido:** Añadir metadatos contextuales a las activaciones. La MMC teórica ya menciona etiquetas emocionales, temporales, situacionales en nodos. Esto podría implementarse permitiendo “capas” contextuales: por ejemplo, una etiqueta de contexto podría significar *“en el contexto de la conversación actual”*, otra podría ser *“bajo una perspectiva optimista”*, etc. Los nodos podrían tener diferentes niveles de activación según contexto (similar a cómo en la memoria humana podemos recordar cosas distintas bajo estados emocionales distintos). Un mecanismo práctico sería tener vectores de contexto que modulen la activación (ej., dot product con embedding de nodo). Esto haría la MMC más adaptable a situaciones: el mismo conocimiento podría interpretarse diferentemente si la situación cambia. También facilitaría el **multi-usuario o multi-objetivo**: la MMC podría servir a diferentes usuarios manteniendo un estado contextual para cada uno, sin mezclar sus informaciones (una especie de *mind partitioning* controlado por contexto).
* **8. Nuevos tipos de fragmentos especializados:** Según evolucione la tecnología, se podrían incorporar fragmentos no contemplados originalmente. Por ejemplo, un **fragmento cuántico** (si se integrara computación cuántica, almacenar qubits de conocimiento o soluciones obtenidas cuánticamente), o fragmentos de **aprendizaje profundo** (que encapsulen una mini-red neuronal entrenada dentro de la memoria para alguna subtarea, actuando como nodo que dado unos inputs devuelve un output). Estos fragmentos funcionarían como *plugins* funcionales dentro de la MMC. La arquitectura debe ser modular para aceptar nuevas clases de nodos con comportamientos propios.
* **9. Aprendizaje Continuo y Evitación de Sesgos:** A medida que la MMC aprende en el tiempo, es propensa a **sesgos** (por sesgo de confirmación podría reforzar demasiado cierta perspectiva si siempre explota lo mismo). Se podría implementar un monitor de **diversidad** que use las técnicas de clustering/patrones para verificar que la MMC no se esté volviendo demasiado unilateral. Por ejemplo, si todas las rutas laterales creadas en el último día conectan a los mismos dos clusters, tal vez está *encasillada* y se fuerza un cambio incrementando exploración en otro área. Además, la MMC debería tener mecanismos de **recalibración**: tal vez mediante feedback externo (un oráculo que diga qué respuestas estuvieron mal, para debilitar esas conexiones). Incluir un bucle de **aprendizaje supervisado ocasional** podría mejorar calidad: e.g., después de cierto periodo, ajustar los pesos globalmente mediante un algoritmo (como retropropagación en un grafo diferenciable si se formula así) para que la memoria produzca salidas más alineadas con la verdad objetiva conocida.

En conclusión, la MMC evolucionada sería una arquitectura **viva** en toda regla: multi-capa (trabajo vs largo plazo, declarativa vs procedimental si agregamos reglas), con **mecanismos de atención** que regulan qué parte del grafo está activa, con **aprendizaje multi-paradigma** (asociativo, deductivo, estadístico), y con **interacción contextual** con el mundo. Esta MMC integraría la inspiración biológica (plasticidad, redes semánticas humanas) con las mejores prácticas de arquitecturas cognitivas y machine learning contemporáneo.

La idea final es que una **Memoria de Mapa Conceptual** bien diseñada podría funcionar como el corazón de un sistema de inteligencia artificial general, manteniendo conocimiento de manera estructurada pero flexible, permitiendo creatividad sin perder fiabilidad, aprendiendo constantemente de la experiencia e incluso introspectivamente de sí misma. Es un desafío formidable, pero las piezas analizadas en este informe ofrecen un camino tangible para construirla. Con cada nueva capa funcional sugerida, la MMC se acerca más a emular las capacidades adaptativas de una memoria humana, pero con la velocidad y escala de la máquina, convirtiéndose en un cimiento para inteligencias artificiales verdaderamente **adaptativas y cognoscitivas**.

Citas

[[Favicon](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html#:~:text=adaptability%20,like%20rule)](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html" \l ":~:text=adaptability%20,like%20rule" \t "_blank)

**[Introduction to the Semantic Pointer Architecture — NengoSPA 2.0.1.dev0 docs](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html" \l ":~:text=adaptability%20,like%20rule" \t "_blank)**

[https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html" \l ":~:text=adaptability%20,like%20rule" \t "_blank)

Todas las fuentes

[[Favicon](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html#:~:text=adaptability%20,like%20rule)nengo](https://www.nengo.ai/nengo-spa/user-guide/spa-intro.html" \l ":~:text=adaptability%20,like%20rule" \t "_blank)