

Procesamiento de Lenguaje Natural III

Docentes:

Esp. Abraham Rodriguez - FIUBA

Mg. Oksana Bokhonok - FIUBA

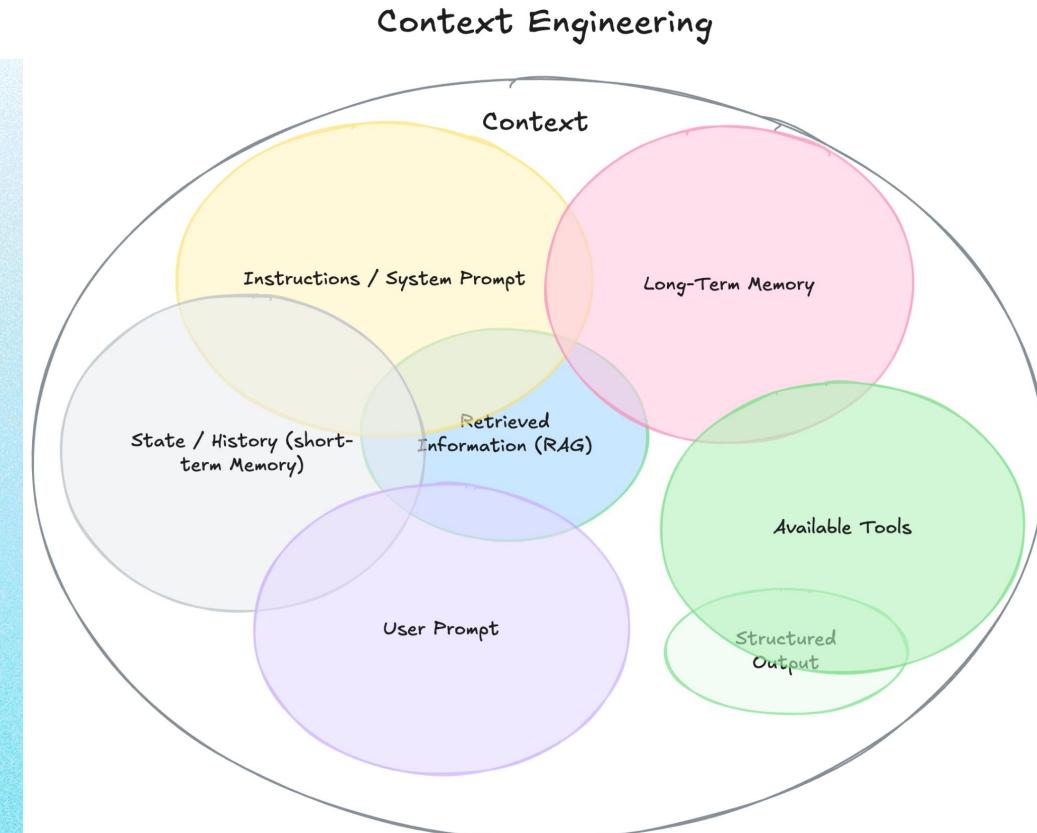
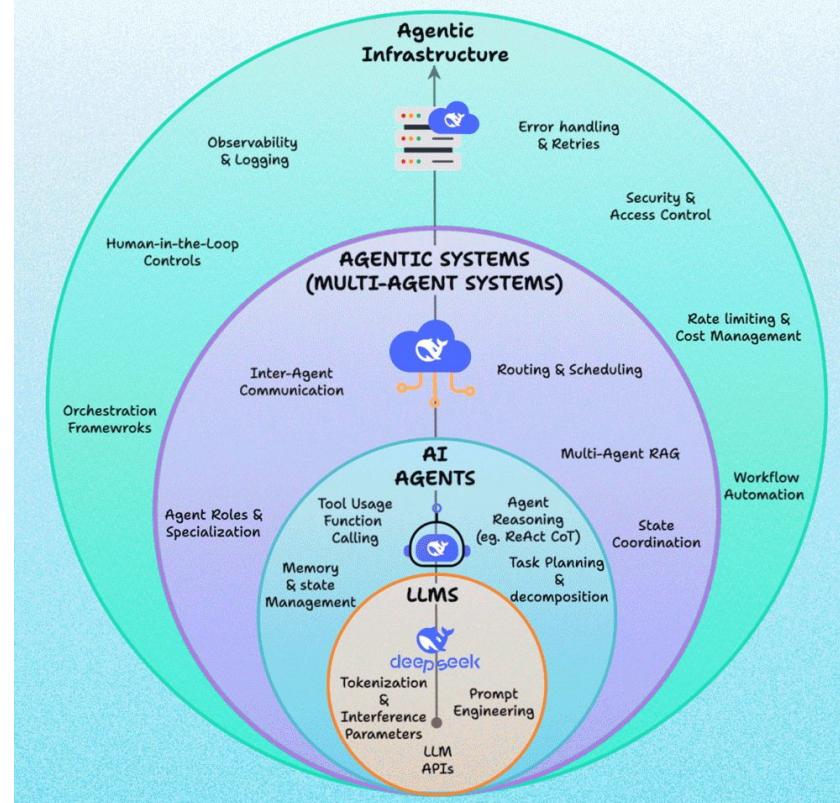
Programa de la materia

1. RAG avanzado y personalización de soluciones
2. Seguridad. Ética y alineación de modelos con valores humanos.
3. Sistemas cognitivos y agentes autónomos.
4. Escalabilidad y Optimización.
5. Diseño de un sistema Agentic IA
6. **Knowledge Graphs, Agentes cognitivos y Sistemas Multiagentes**

Agentes efectivos

Como sabemos, una arquitectura de agentes puede ser efectiva según la calidad de interacción entre agentes, manejo de estados y context length, etc. Resumiéndose nuevamente en **Context Engineering**.

Sin embargo el contexto no es un **blob**, se encuentra segmentado por naturaleza.



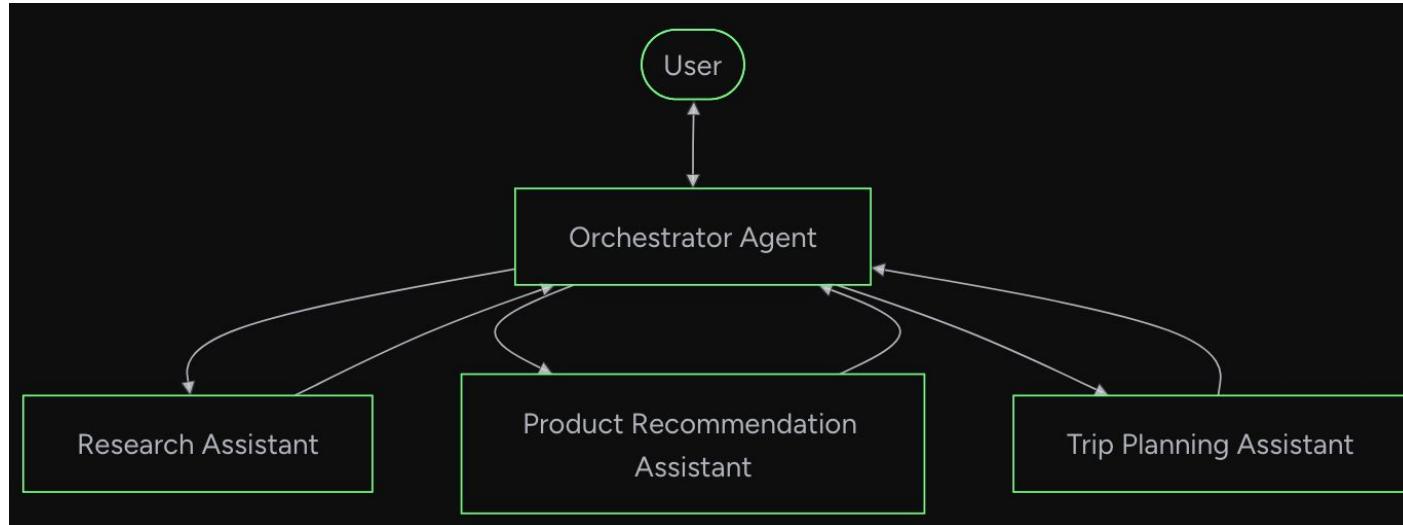
Agentes efectivos

Un sistema de Agentes normalmente trata de satisfacer múltiples labores como:

- Coding Assistant.
- Travel Assistant, etc.

Esto implica que en cualquier momento puede ocurrir un turnover o cambio de contexto en una conversación.

Sabemos que un sistema multiagente sería ideal para este tipo de problemas, sin embargo el **dominio** debe estar muy bien definido.

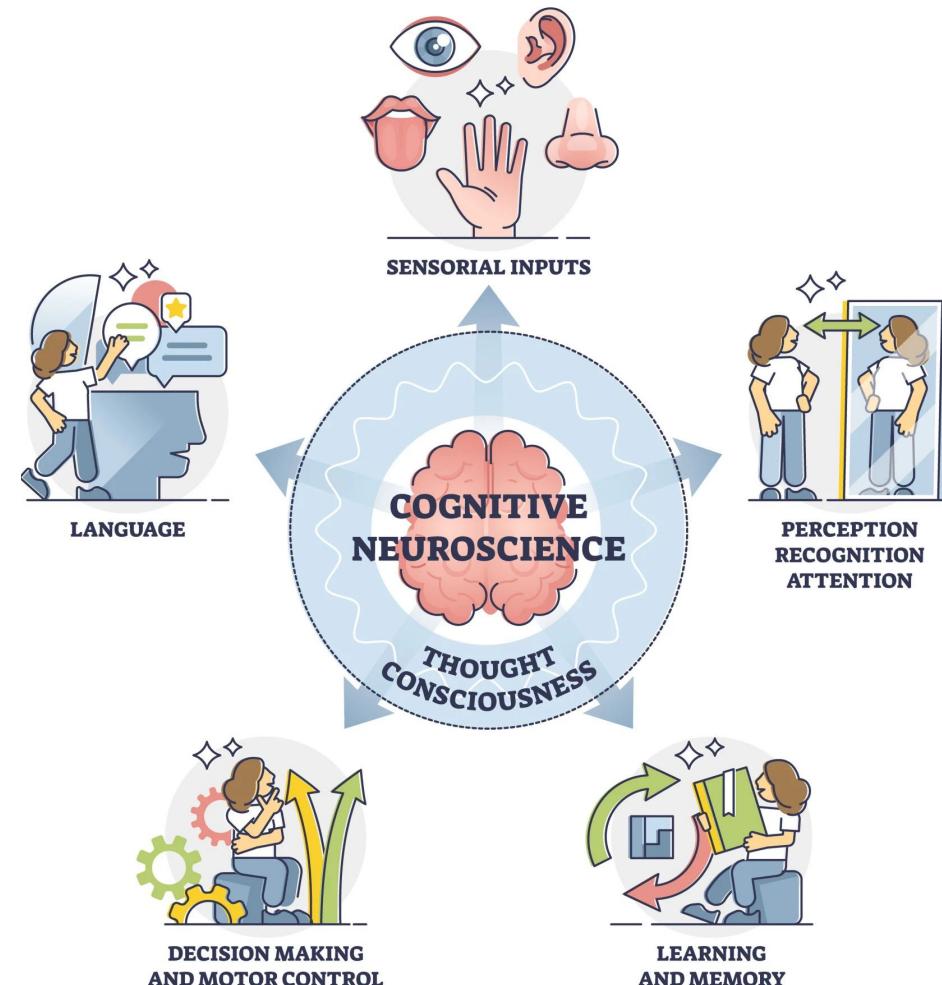
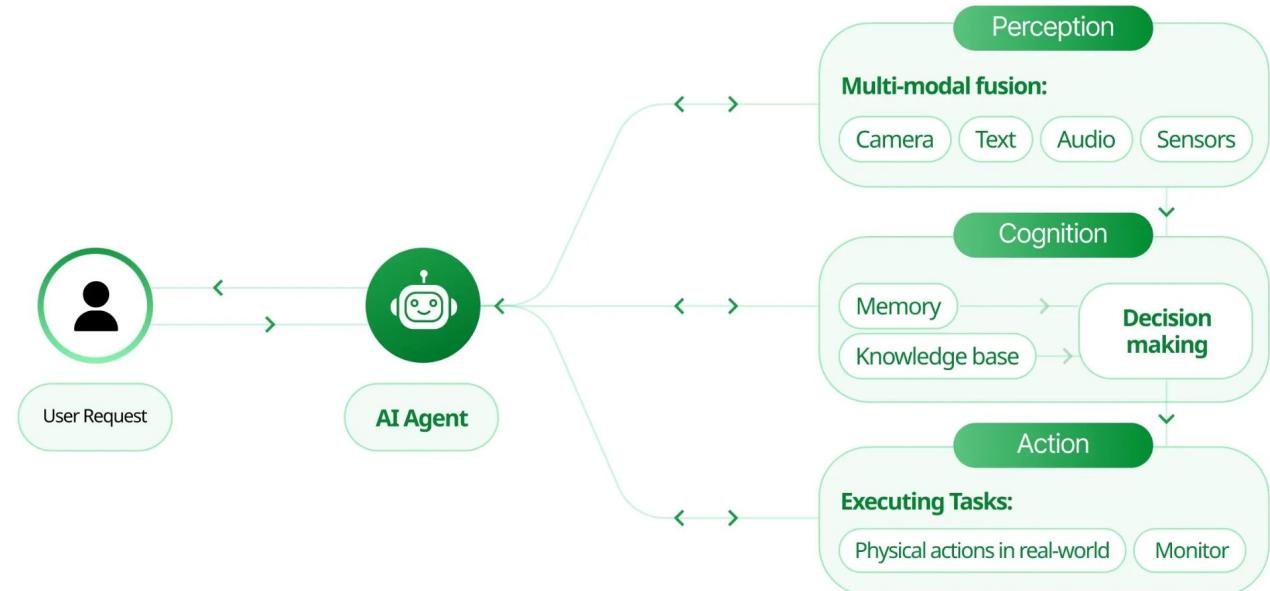


IA Cognitiva

Con nuestros agentes, lo que ocupamos lograr es cognición.

Para que la IA sea efectiva se debe considerar un ambiente dinámico (contextual), emulando a la conciencia humana la cual recibe inputs multimodales, aprende, decide y se comunica.

Actualmente podemos emular hasta cierto punto **mediante el manejo del contexto**.



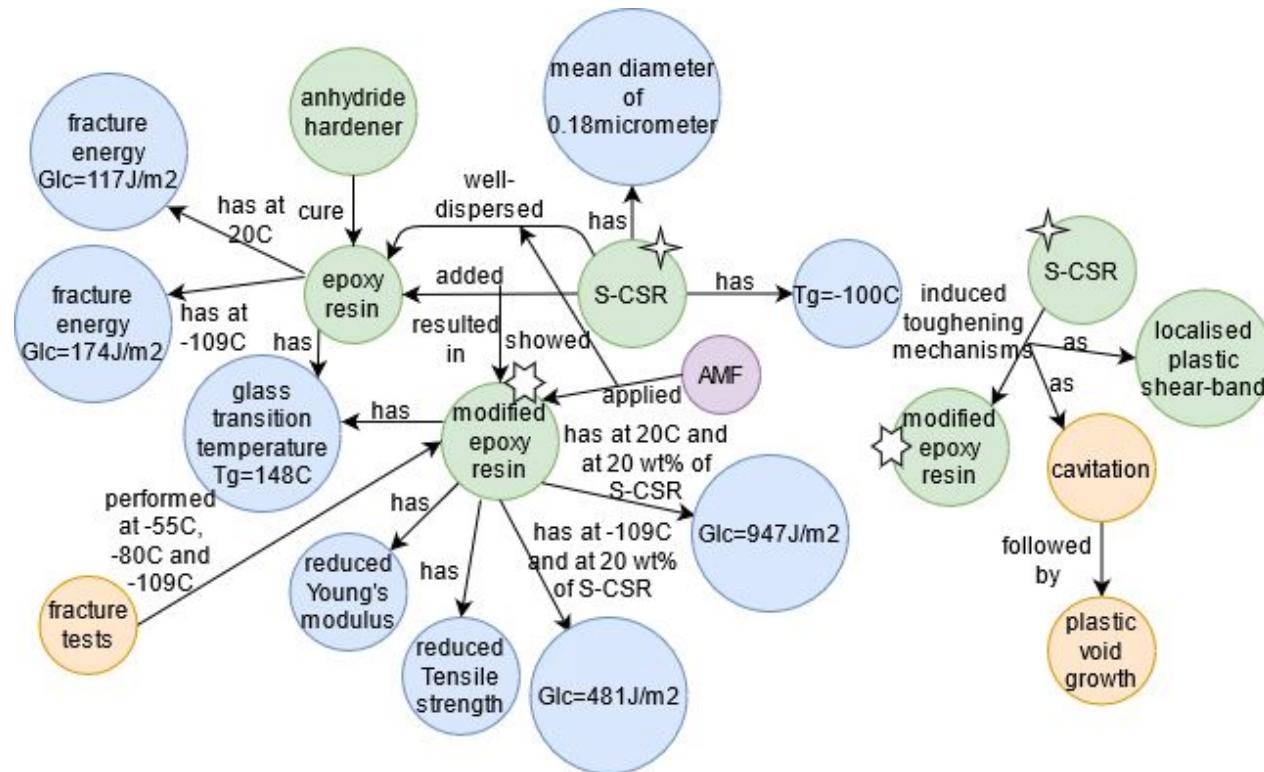
Knowledge Graphs

Es común tener **un solo** vectorDB con documentos de todo tipo de contenido (Vector-based RAG), configurarlo y delegar la labor al algoritmo de búsqueda para **retornar los documentos más similares**.

Sin embargo, solo aumentamos la probabilidad de obtener contexto relevante, pero **no está garantizada la exactitud**, puede carecer contexto y desconexión con el query.

Además es común preguntarse y buscar pistas del porqué se realizó una decisión.

Un knowledge graph es una forma más eficiente y efectiva que depender de una búsqueda por similitud. Consiste en segmentar y relacionar conceptos/objetos.



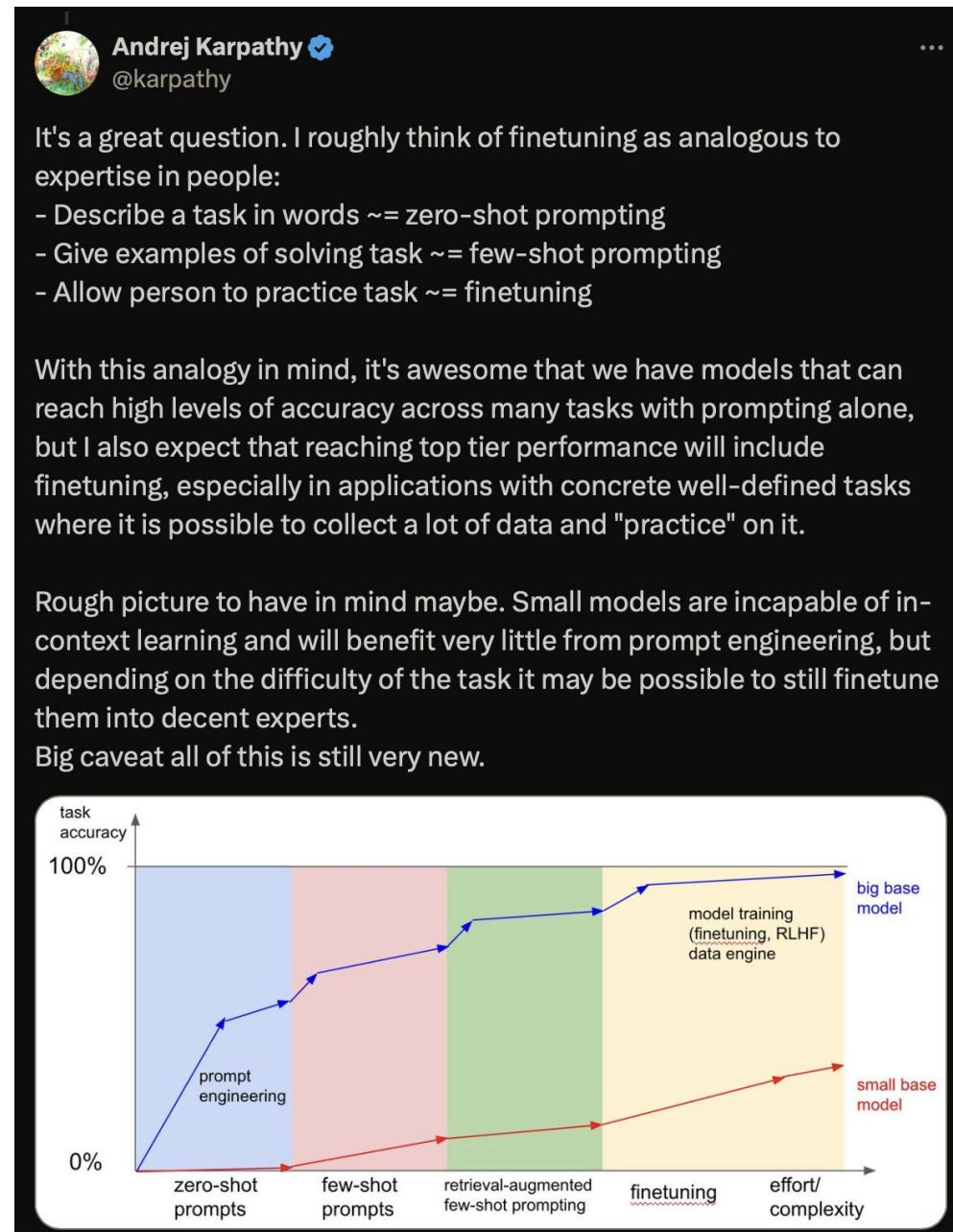
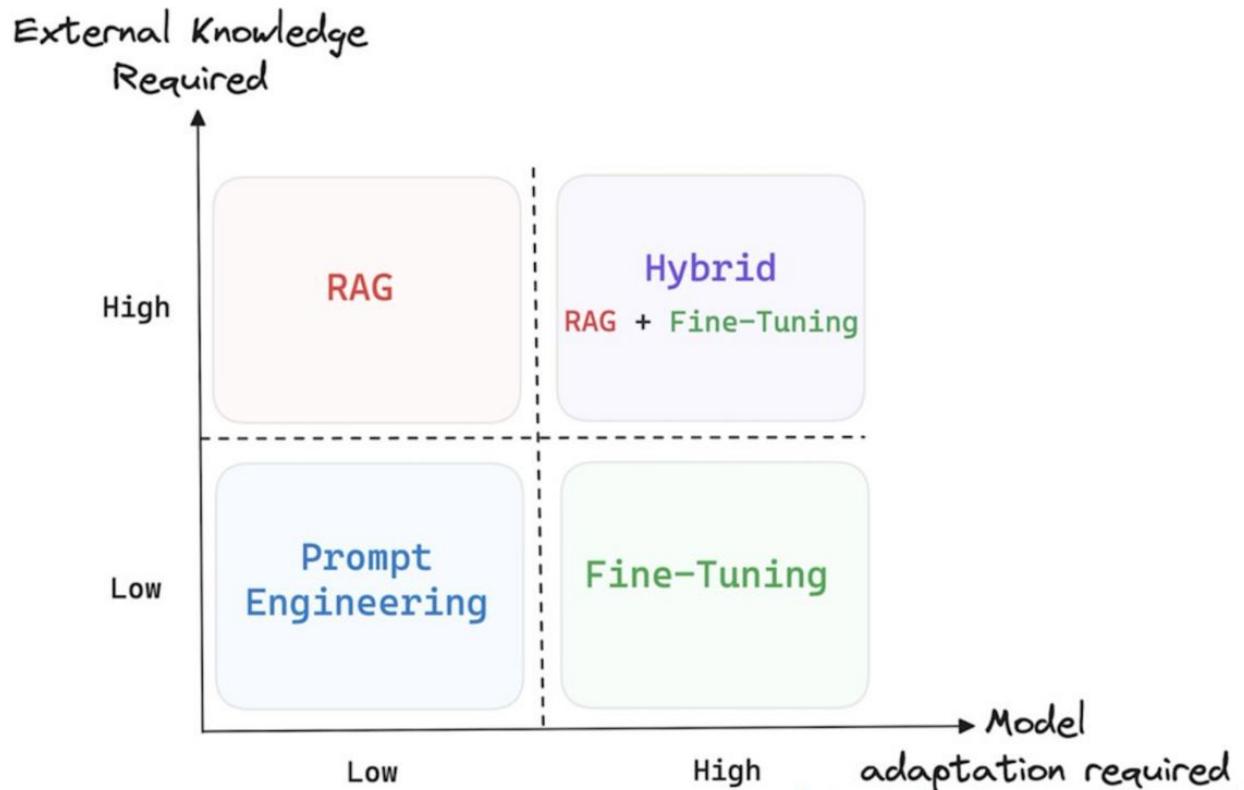
[Neo4j: What is a Knowledge Graph?](#)

Knowledge Graphs

La pregunta es, cómo podemos exprimir al máximo el manejo del contexto antes de realizar fine-tuning?

La respuesta es:

LLMs de razonamiento + Retrievers robustos



Andrej Karpathy ✅
@karpathy

It's a great question. I roughly think of finetuning as analogous to expertise in people:

- Describe a task in words ~ zero-shot prompting
- Give examples of solving task ~ few-shot prompting
- Allow person to practice task ~ finetuning

With this analogy in mind, it's awesome that we have models that can reach high levels of accuracy across many tasks with prompting alone, but I also expect that reaching top tier performance will include finetuning, especially in applications with concrete well-defined tasks where it is possible to collect a lot of data and "practice" on it.

Rough picture to have in mind maybe. Small models are incapable of in-context learning and will benefit very little from prompt engineering, but depending on the difficulty of the task it may be possible to still finetune them into decent experts.

Big caveat all of this is still very new.

Knowledge Graphs

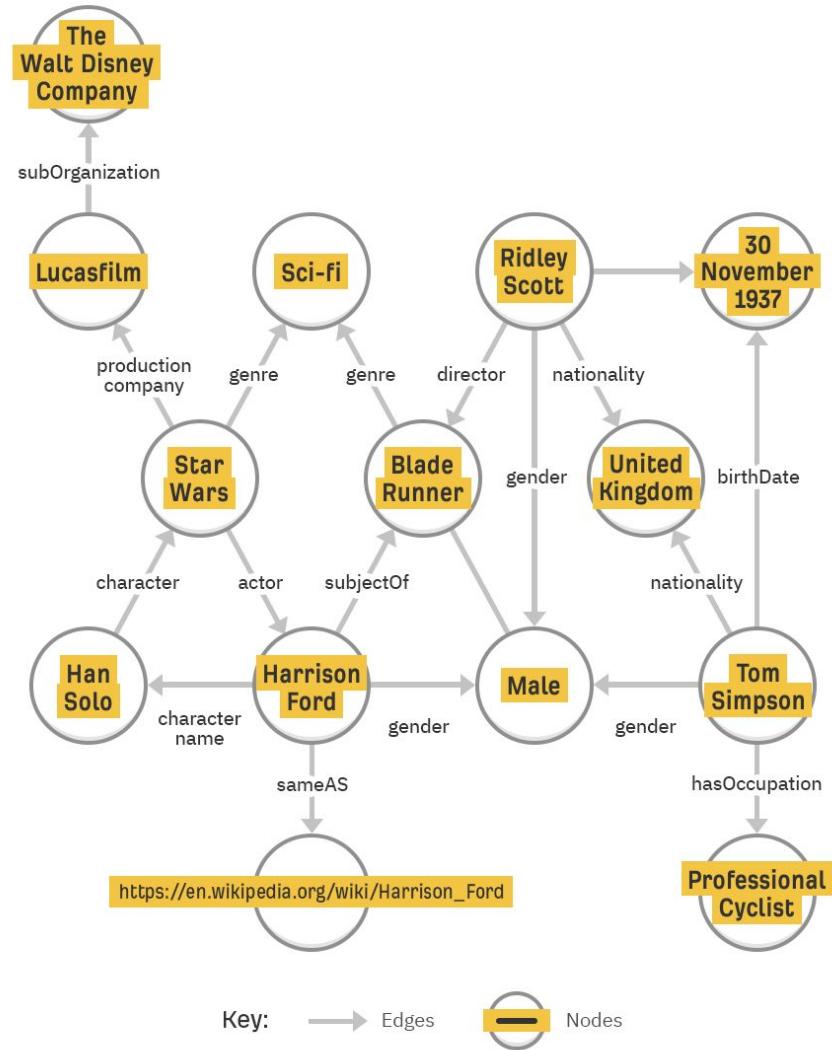
El concepto de KG no es reciente sin embargo su mayor impacto fue en 2012 cuando Google introdujo KG en su motor de búsqueda.

[Google Blog: Introducing Knowledge Graphs](#)

Un **query para un search engine** consiste en una búsqueda de match de keywords. Sin embargo los queries son más ricos en significado por ejemplo:

UserQuery	Resultado Real	Expectativa del usuario	UserMetadata
Buenos Aires	Ciudad de Buenos Aires	Buenos Aires Cafe	IP Address, Geolocation (Honduras)

What Google's Knowledge Graph Looks Like

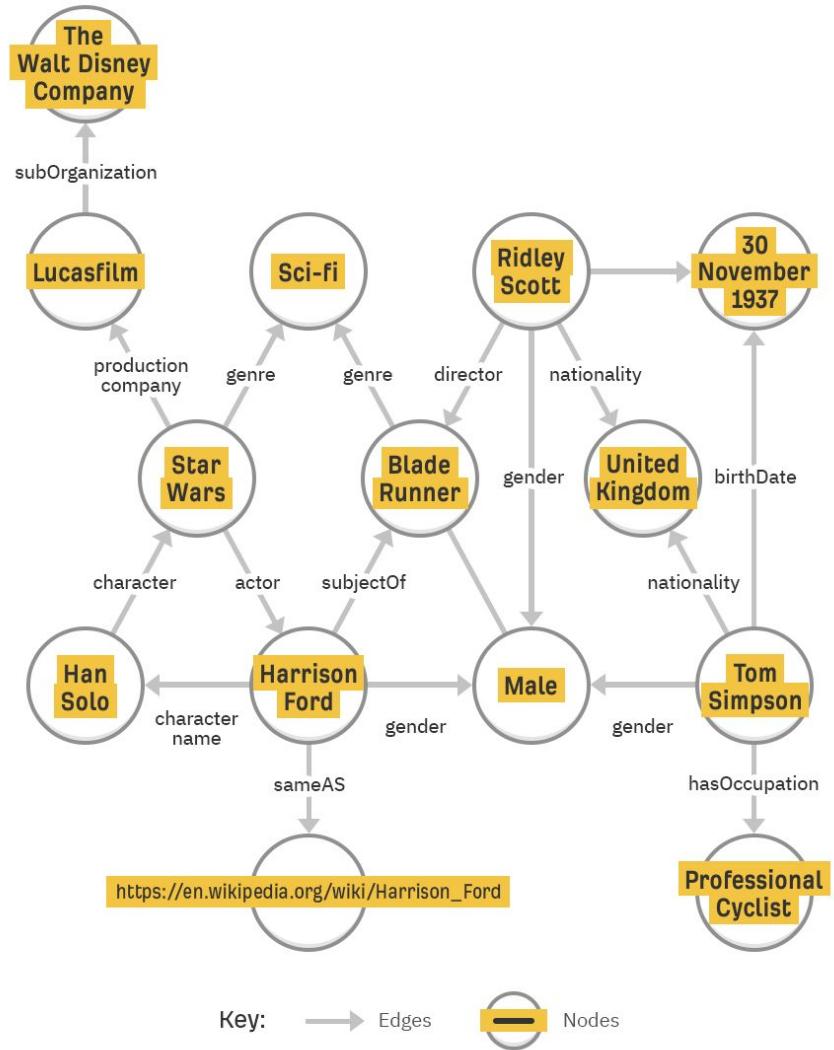


Knowledge Graphs

Segun el contexto del query (incluyendo metadata) la búsqueda implica encontrar relaciones **entre conceptos/entidades** y no en simples keywords.

Esta asociación compleja trata de emular lo que la neurociencia cognitiva estudia.

What Google's Knowledge Graph Looks Like



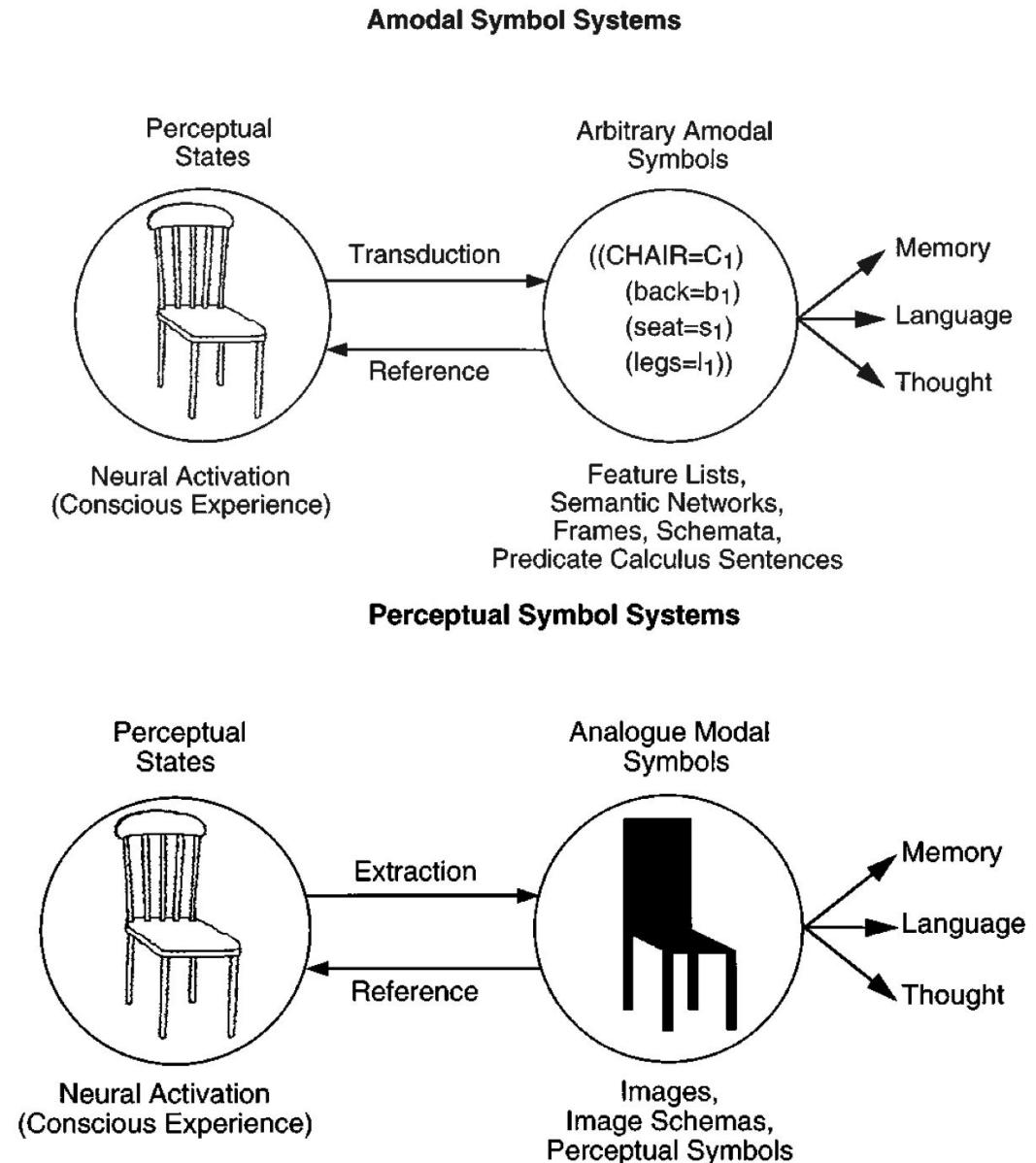
Break: Neurociencia Cognitiva

El paper [Perceptual symbol systems](#) (Barsalou, 1999) discute acerca de los **sistemas de percepción** considerando la **desconexión de la época** entre investigaciones de percepción y cognición. Introduce un framework (PSS) como modelo teórico de representación.

En neurociencia existen los:

Los sistemas **amodales** (representaciones mediante transducción/conversión) son amodales ya que no están relacionados a los estados perceptuales que los generan estos surgen de.

Por otro lado los sistemas **perceptuales** (PSS) son análogos al estado perceptual que lo genera.



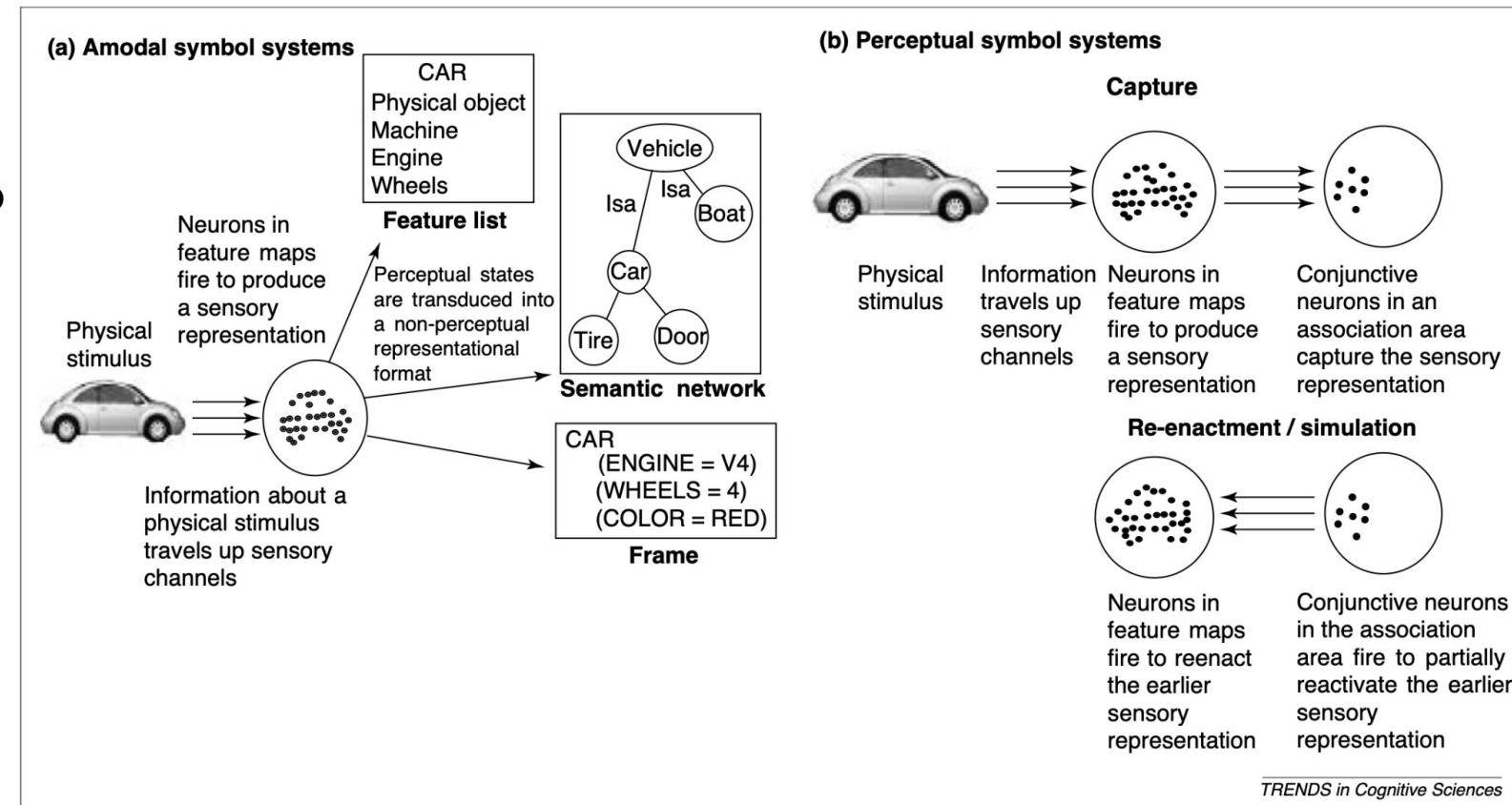
Break: Neurociencia Cognitiva

El paper [Grounding conceptual knowledge in modality-specific system](#) (Barsalou et al, 2003), Define mediante evidencias empíricas que la cognición no es abstracta y amodal sino mediante sistemas modales.

Por ejemplo:

Al decir “limón” el cerebro activa diferentes áreas siendo:

- Corteza Visual
- Corteza Gustativa
- Corteza Olfativa



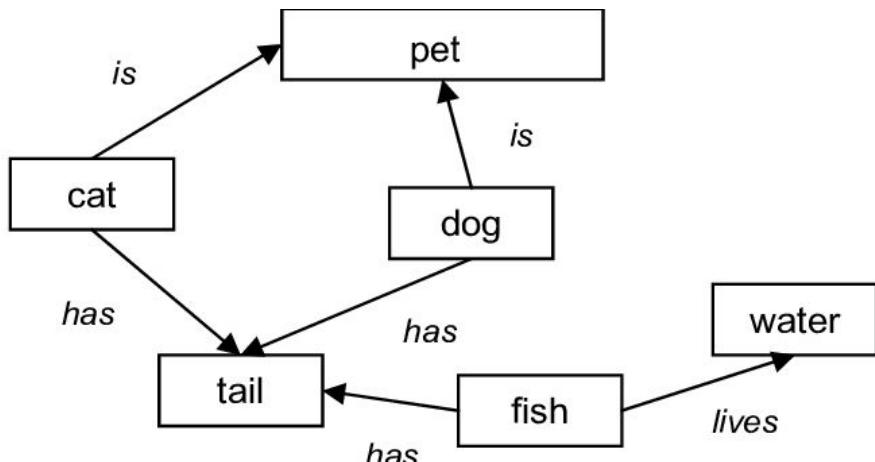
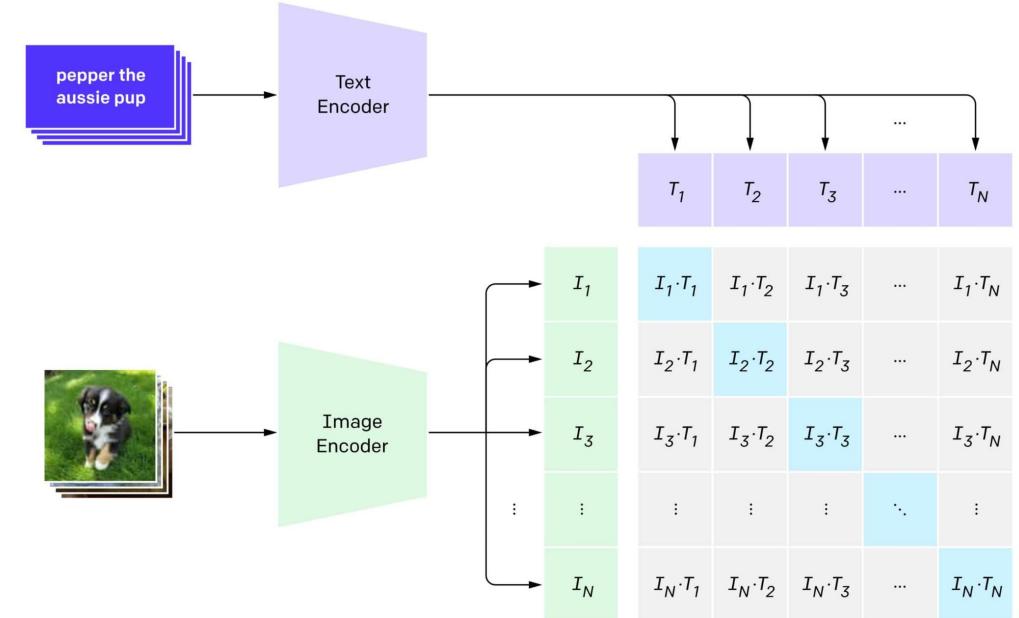
Break: Neurociencia Cognitiva & IA

De lo anterior, vemos que existen diversas maneras de representar la cognición humana, en el caso de AI, CLIP mediante la distribución de probabilidad conjunta logra aplicar de cierta manera PSS.

Por otro lado el caso amodal, los **knowledge graphs** son básicamente redes semánticas dentro de la neurociencia.

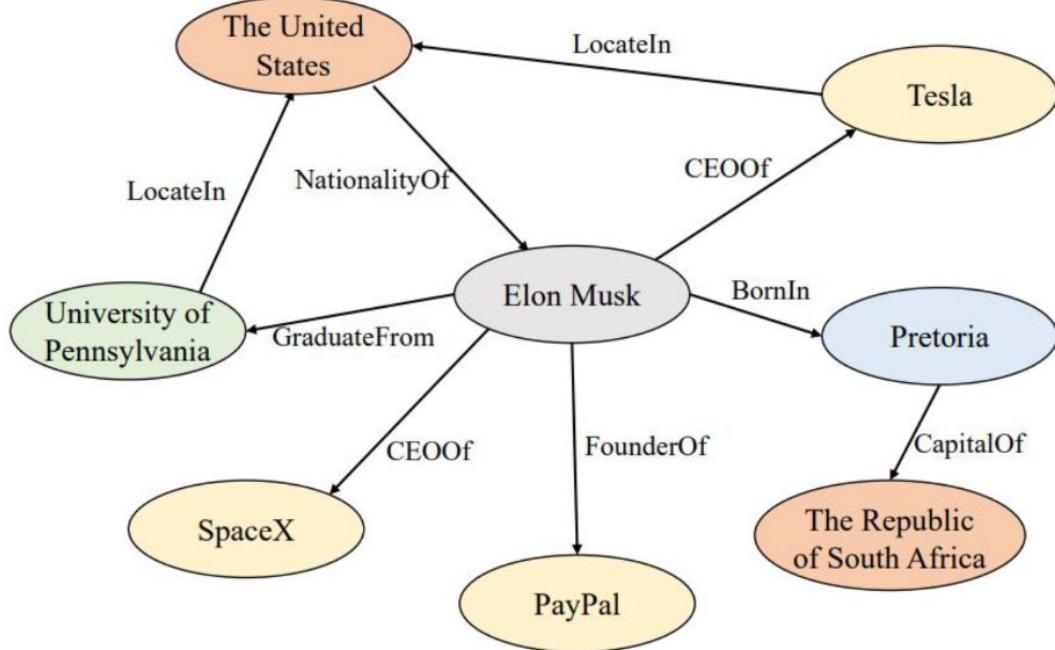
Ambas representaciones son extremadamente útiles en IA especialmente en sistemas de agentes cognitivos.

Lo que se busca es representar lo abstracto como conceptos, razonamiento, acciones, y lógica, así como lo fundamentado en percepción (visión, audio, etc.) que encapsula el entendimiento del ambiente, etc.

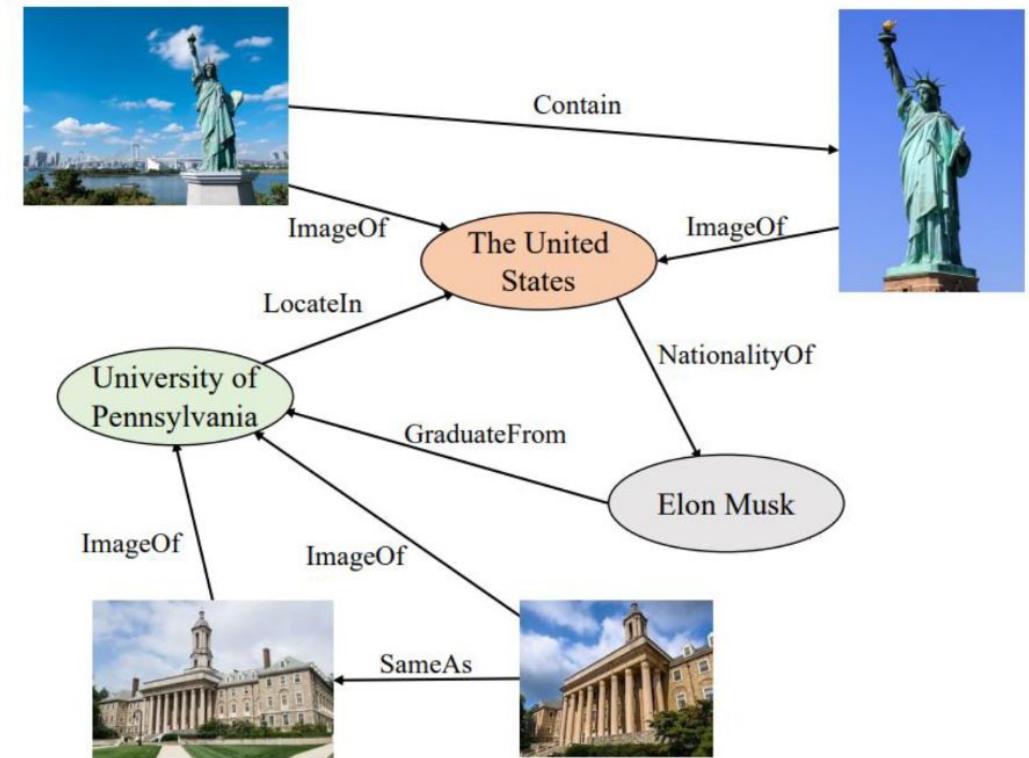


Break: Neurociencia Cognitiva & IA

Survey on Multimodal Knowledge Graphs



(a) Knowledge Graph

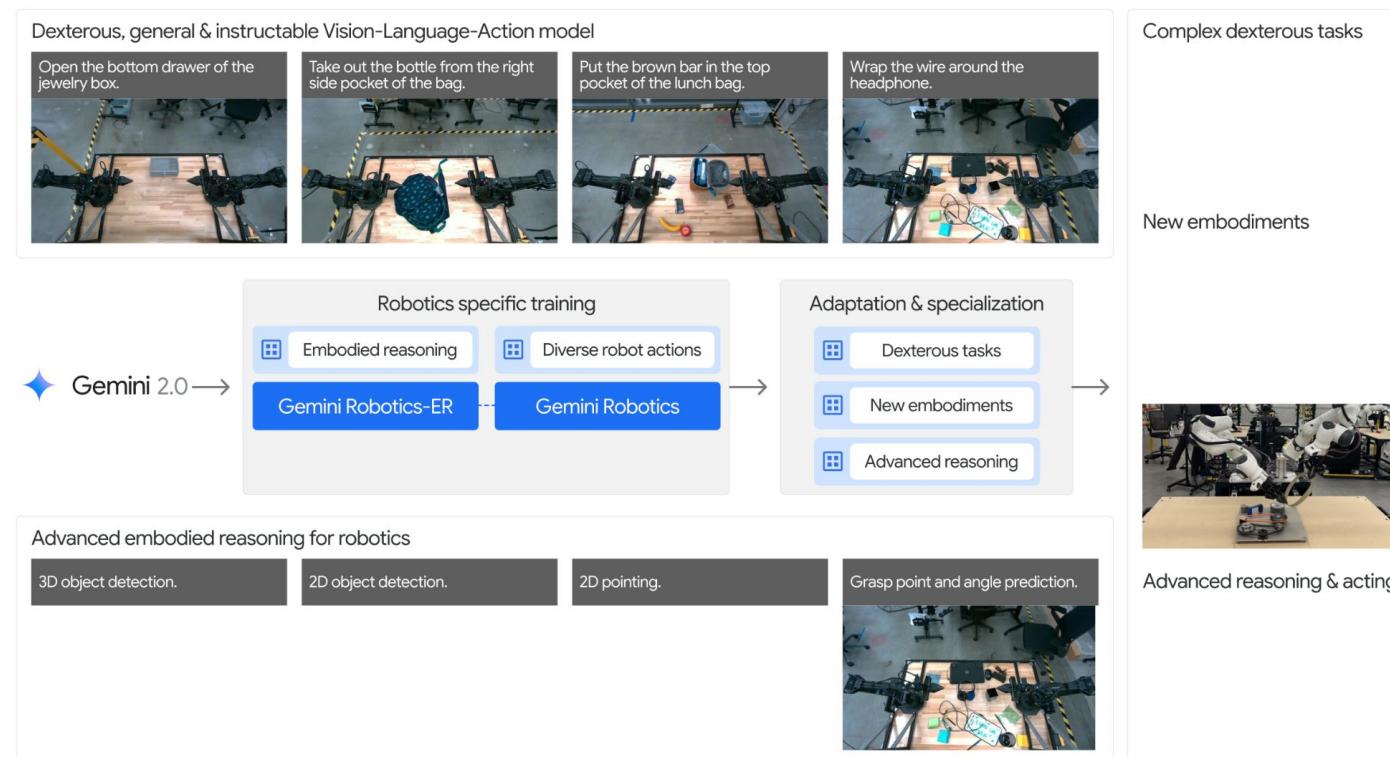


(b) Multimodal Knowledge Graph

Break: Robótica cognitiva

Es fácil deducir que la finalidad de la IA cognitiva es la robótica donde los sistemas de percepción (sensores) ya existen pero es la asociación de la percepción con la cognición el problema real.

Recordando de la materia de **Visión por computadora III**, el paper [RT-2: Vision-Language-Action Models](#) es la demostración de la necesidad y unión entre lo modal y amodal.



Hoy en día existen sistemas de agentes multimodales en robótica como: [Gemini Robotics API](#) [Gemini Robotics Paper](#) [Gemini Robotics 1.5 brings AI agents into the physical world](#)

GraphRAG

Enfocándonos en sistemas de **agentes**, el Knowledge Graph es una gran solución para **brindar contexto correcto en el momento correcto** considerando las relaciones y sentidos sobre el query. Por otro lado Vector RAG consiste en encontrar similitudes semánticas.

GraphRAG es KG aplicado como RAG el cual contrasta con VectorRAG en la habilidad de responder queries que requieren un “sentido” acerca del data corpus.

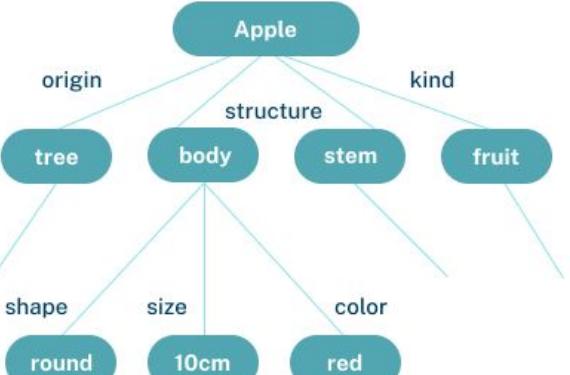
Ejemplo: El query “¿Que capacidades tenes?”, no obtiene ningún documento similar en VectorRAG, pero en GraphRAG **corresponde a acceder a resúmenes sobre la estructura del KG.**

Human View of an Apple



Vector View of an Apple

Knowledge Graph View of an Apple



From Local to Global: A GraphRAG Approach to Query-Focused Summarization

Darren Edge^{1†} **Ha Trinh**^{1†} **Newman Cheng**² **Joshua Bradley**² **Alex Chao**³
Apurva Mody³ **Steven Truitt**² **Dasha Metropolitansky**¹ **Robert Osazuwa Ness**¹

Jonathan Larson¹

¹Microsoft Research
²Microsoft Strategic Missions and Technologies
³Microsoft Office of the CTO

{daedge, trinhha, newmancheng, joshbradley, achao, moapurva, steventruitt, dasham, robertness, jolarso}@microsoft.com

[†]These authors contributed equally to this work.

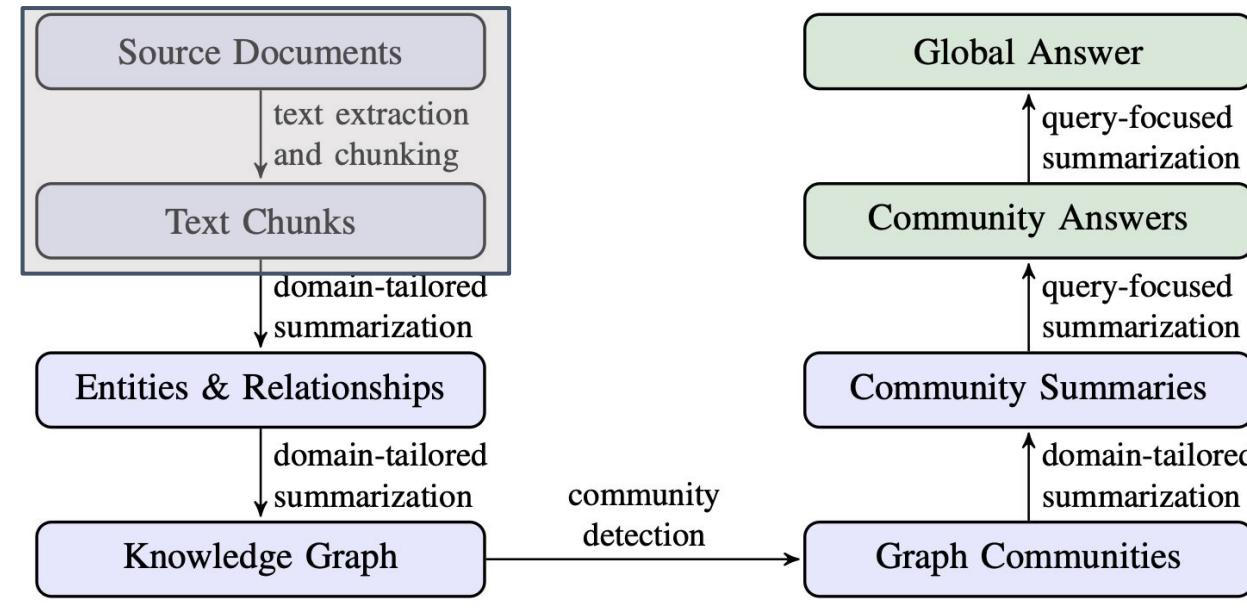
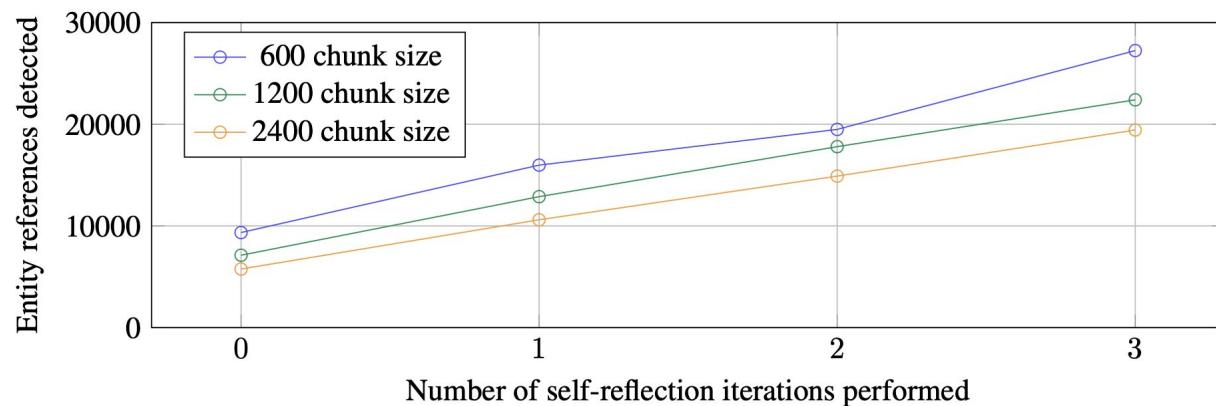
Abstract

The use of retrieval-augmented generation (RAG) to retrieve relevant information from an external knowledge source enables large language models (LLMs) to answer questions over private and/or previously unseen document collections. However, RAG fails on global questions directed at an entire text corpus, such as “What are the main themes in the dataset?”, since this is inherently a query-focused summarization (QFS) task, rather than an explicit retrieval task. Prior QFS methods, meanwhile, do not scale to the quantities of text indexed by typical RAG systems. To combine the strengths of these contrasting methods, we propose *GraphRAG*, a graph-based approach to question answering over private text corpora that scales with both the generality of user questions and the quantity of source text. Our approach uses an LLM to build a graph index in two stages: first, to derive an entity knowledge graph from the source documents, then to pre-generate community summaries for all groups of closely related entities. Given a question, each community summary is used to generate a partial response, before all partial responses are again summarized in a final response to the user. For a class of global sensemaking questions over datasets in the 1 million token range, we show that GraphRAG leads to substantial improvements over a conventional RAG baseline for both the comprehensiveness and diversity of generated answers.

GraphRAG Pipeline: Chunks

Empezando por la creación de chunks, donde se debe considerar el chunkSize, según el paper:

“Chunks más grandes requieren menos llamados al LLM pero sufren de memoria degradada sobre información que aparece al inicio del chunk”.



GraphRAG Pipeline: Entidades y relaciones

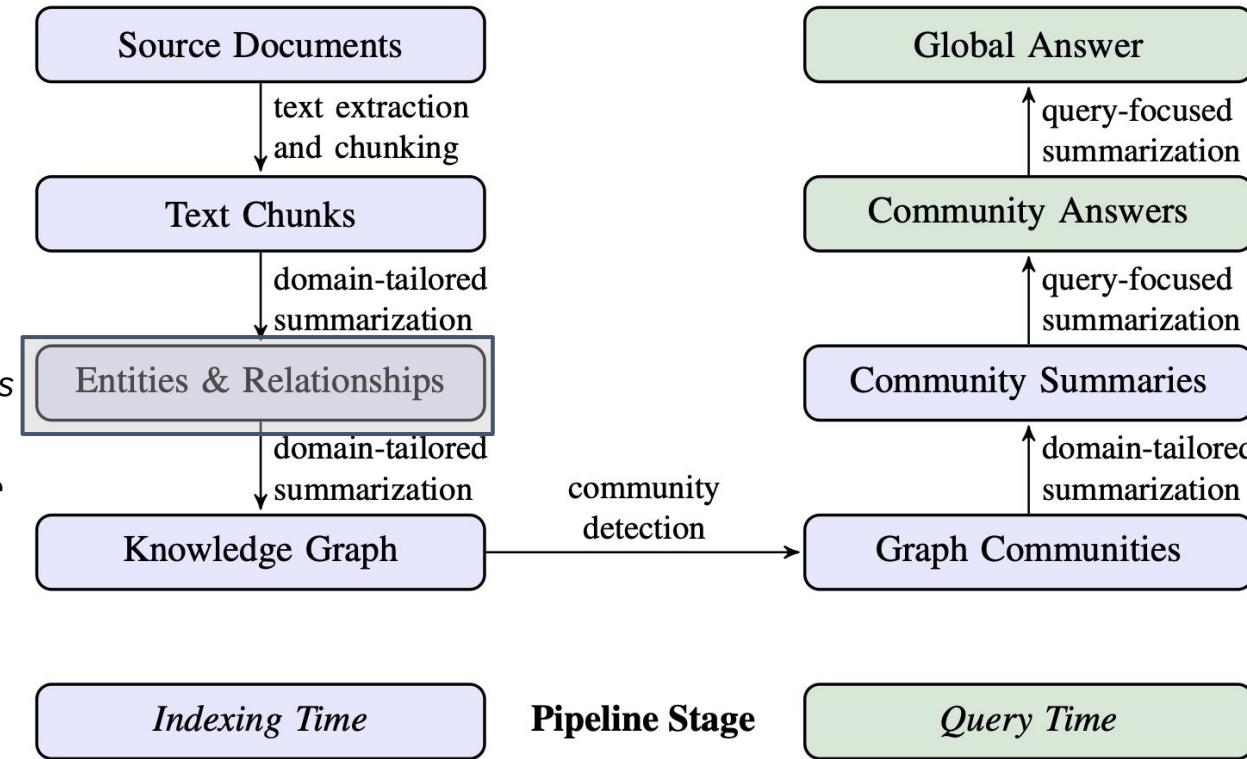
Una vez definidos los chunks, el **LLM es prompted a extraer entidades y relaciones (ER) entre entidades dado un chunk.**

Adicionalmente, el LLM debe generar descripciones cortas para las ER. Ejemplo:

Chunk: “NeoChip’s (NC) shares surged in their first week of trading on the NewTech Exchange. However, market analysts caution that the chipmaker’s public debut may not reflect trends for other technology IPOs. NeoChip, previously a private entity, was acquired by Quantum Systems in 2016. The innovative semiconductor firm specializes in low-power processors for wearables and IoT devices.”

El LLM es prompted dando como resultado:

- The **entity** NeoChip, with description “NeoChip is a publicly traded company specializing in low-power processors for wearables and IoT devices.”
- The **entity** Quantum Systems, with description “Quantum Systems is a firm that **previously** owned NeoChip.”
- A **relationship** between NeoChip and Quantum Systems, with description “Quantum Systems owned NeoChip from 2016 until NeoChip became publicly traded.”



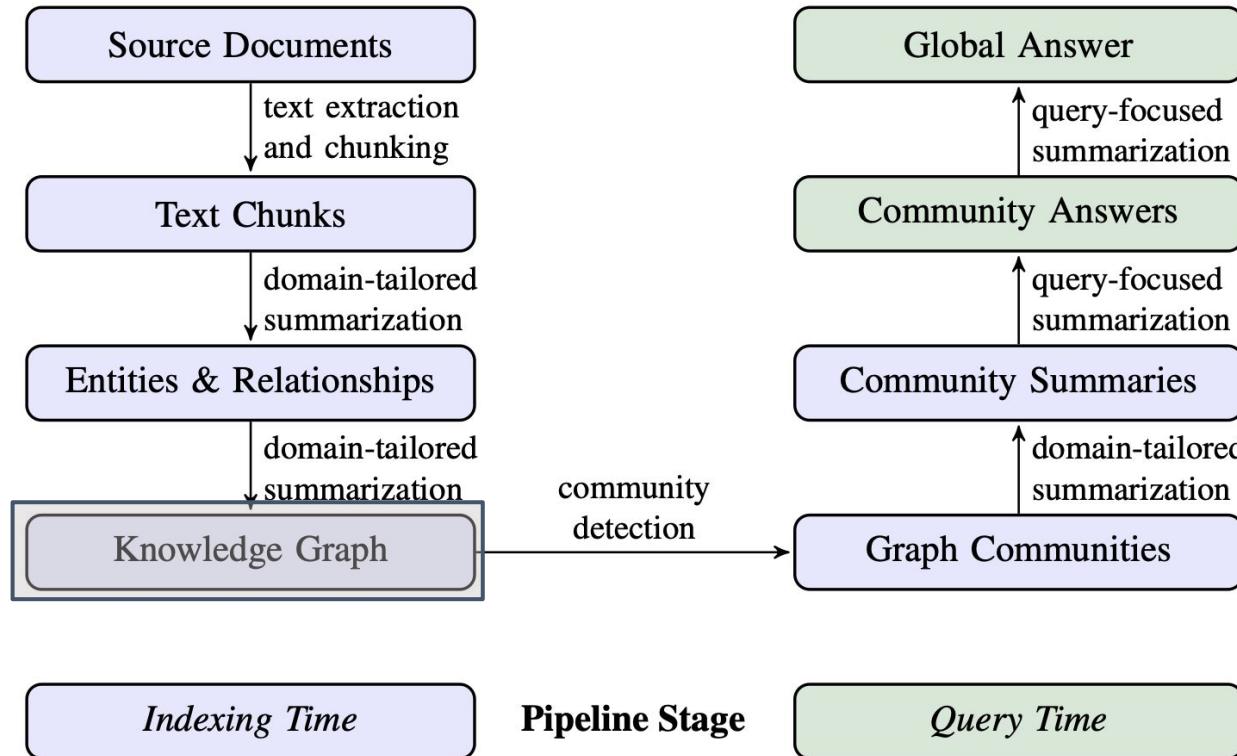
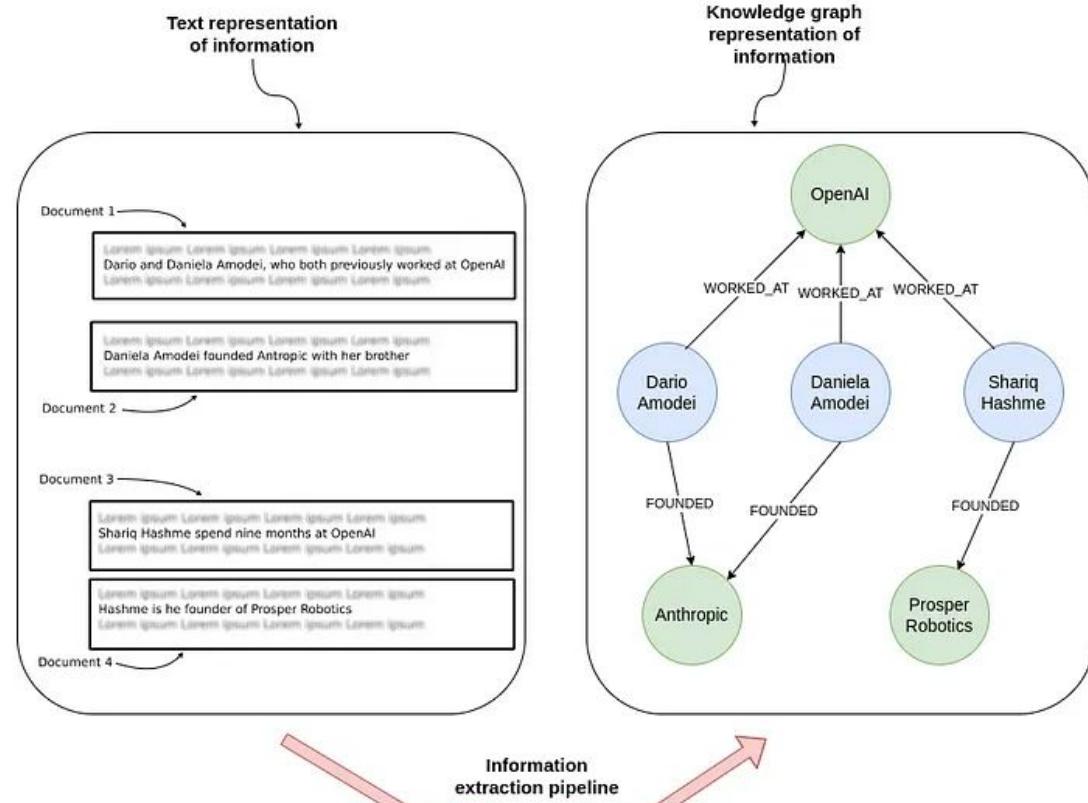
El prompt debe ser ajustado al dominio del corpus.

GraphRAG Pipeline: Knowledge Graph

Los ERs se **convierten en nodos y bordes** individuales del KG.

Las descripciones de las entidades se agregan y resumen para cada nodo y cada borde.

Las relaciones se agrupan en los bordes del grafo, donde el **número de duplicados de una relación dada se convierte en el peso del borde**.

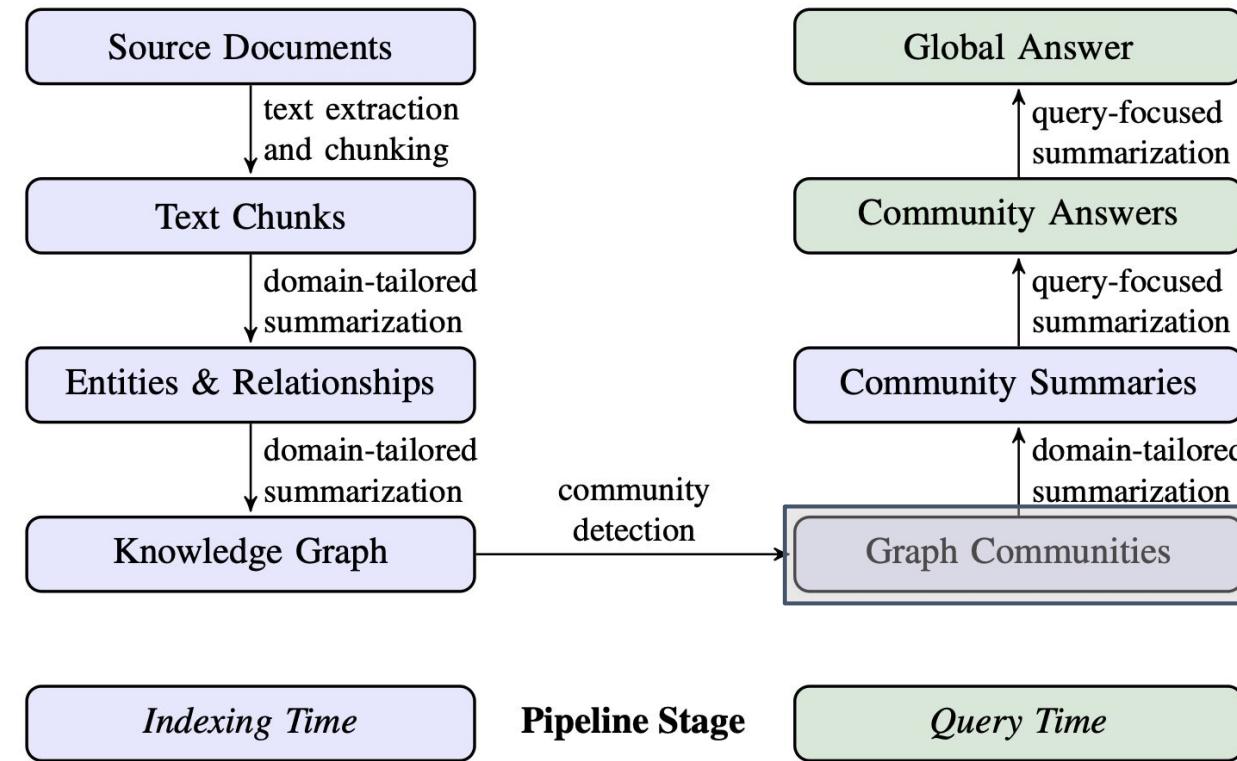
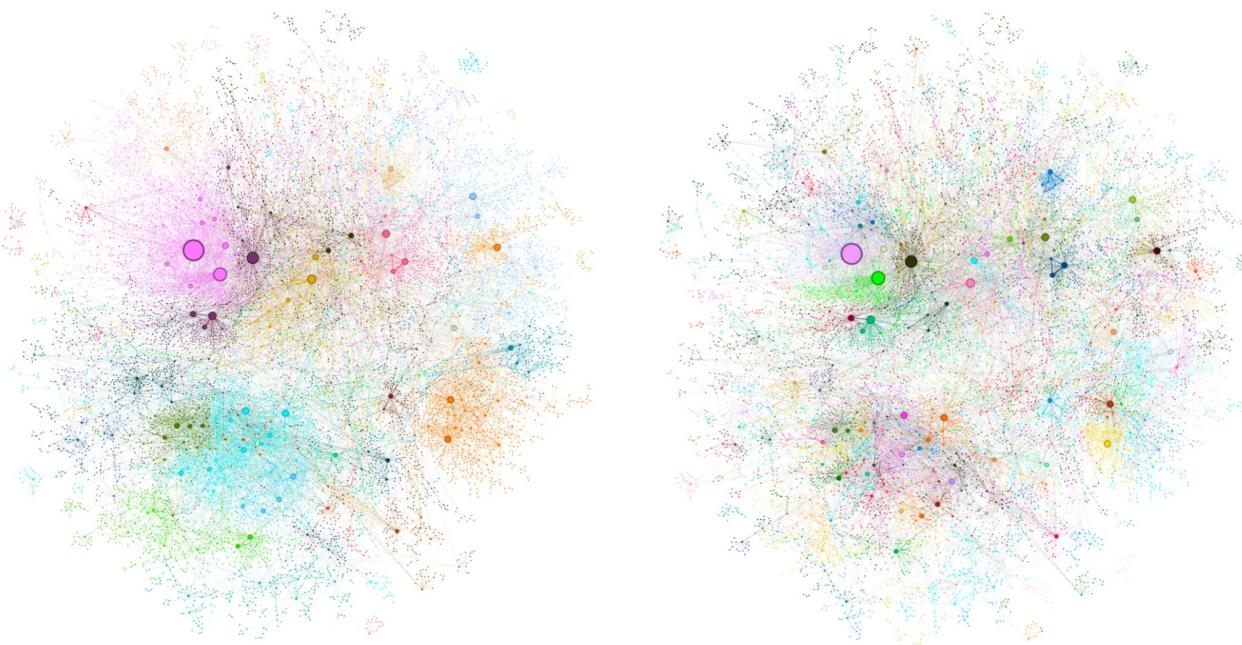


[Building KG with LangChain LLMGraphTransformer](#)
[LLMGraphTransformer Docs](#)

GraphRAG Pipeline: Graph Communities

Dado el índice del grafo generado del paso anterior. Se pueden utilizar algoritmos de **detección de comunidades** para particionar el grafo en comunidades de nodos fuertemente relacionados. En el paper de GraphRAG se utiliza [Leiden Community Detection](#).

La partición en grupos permiten que el LLM pueda resumir en paralelo en ambos indexado y query times.



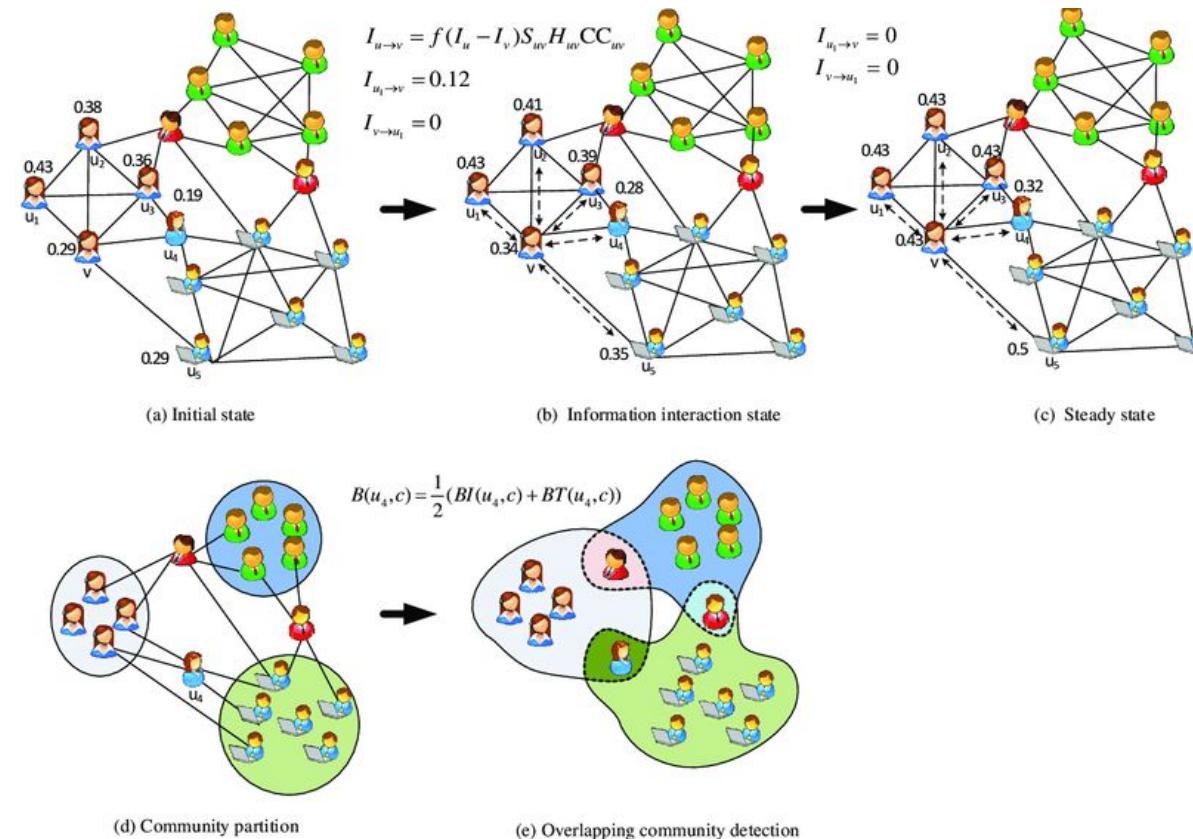
Community Detection ≠ Clusters

En teoría de grafos, **community detection** consiste en **encontrar y agrupar** nodos que están “fuertemente conectados” entre sí dentro de un grafo, de forma **similar a cómo los algoritmos de clustering agrupan puntos similares en espacios vectoriales en ML**.

Ejemplo:

En una red social, podemos buscar identificar comunidades de usuarios con intereses o interacciones similares para recomendar contenido o conexiones relevantes.

*Los algoritmos de detección de comunidades permiten **identificar y filtrar usuarios de interés** dentro del grafo de relaciones.*

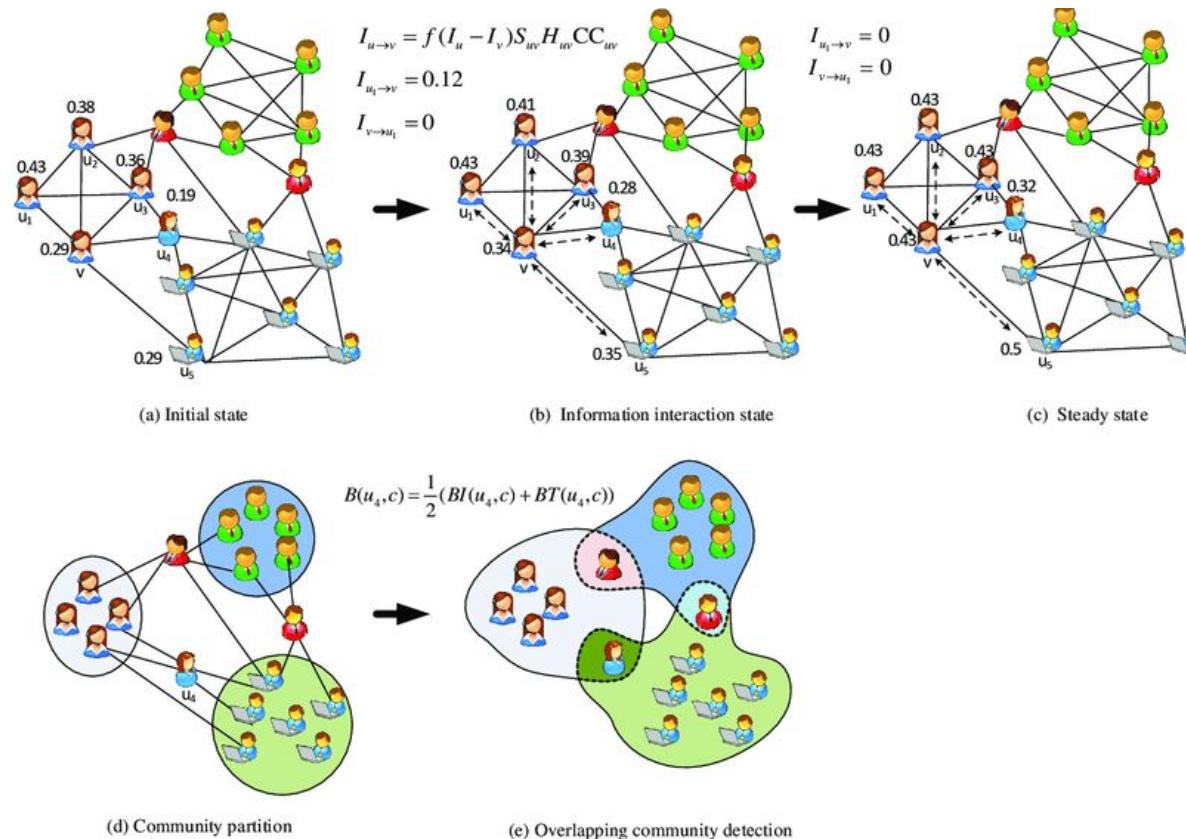


Community Detection ≠ Clusters

Clustering: se realiza sobre espacios de características **vectoriales**, agrupando elementos basados en similitud de sus atributos.

Community detection: se realiza sobre **grafos**, agrupando nodos según patrones de conectividad.

Por otro lado, es posible aplicar clustering a grafos, pero primero se necesita **representar el grafo en un espacio vectorial**, por ejemplo mediante embeddings obtenidos con Graph Neural Networks (GNNs) u otros.



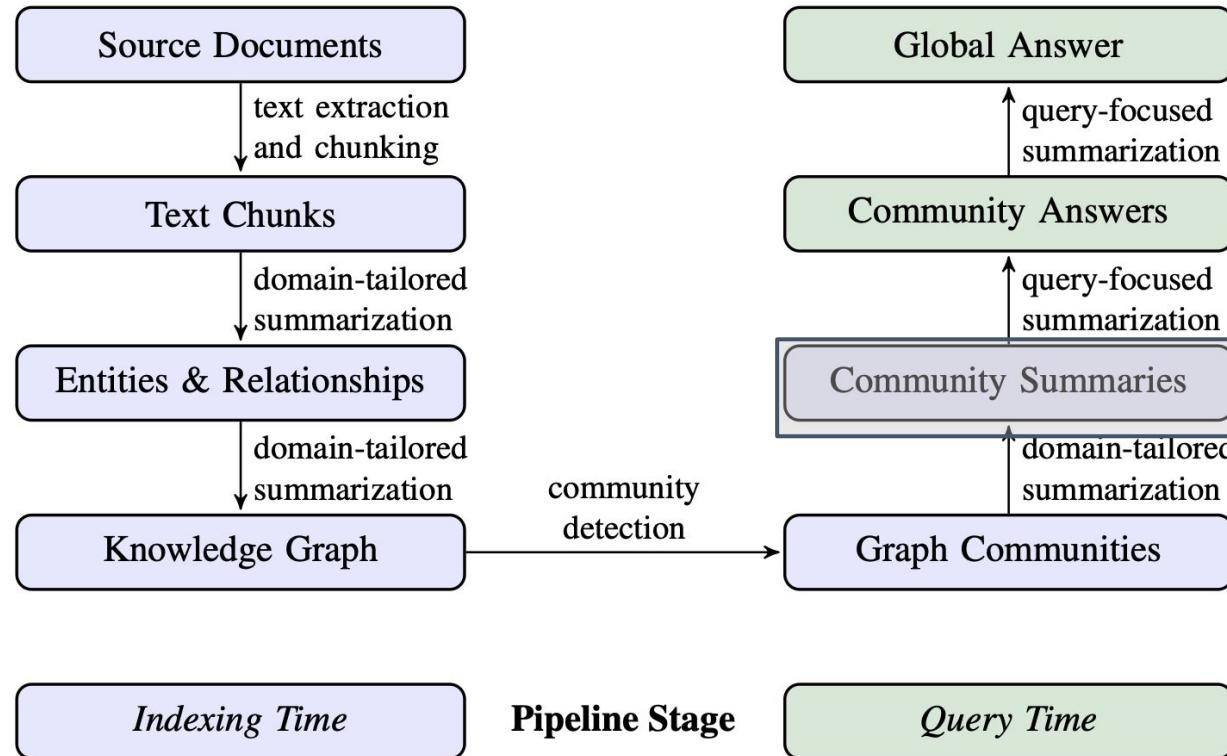
[Neptune.ai](#): GNN & applications
[NVIDIA](#): What are GNN?

GraphRAG Pipeline: Community Summaries

Lo siguiente es crear resúmenes con formato de informe para cada comunidad, utilizando un método diseñado para escalar a conjuntos de datos muy grandes.

Estos resúmenes son útiles de forma independiente como una manera de comprender la estructura y la semántica global del conjunto de datos, y pueden utilizarse por sí mismos para interpretar un corpus en ausencia de una consulta específica.

Por ejemplo, un usuario puede recorrer los resúmenes de comunidades en un cierto nivel buscando temas generales de interés, y luego leer los informes enlazados en un nivel inferior que brindan detalles adicionales sobre cada subtema.

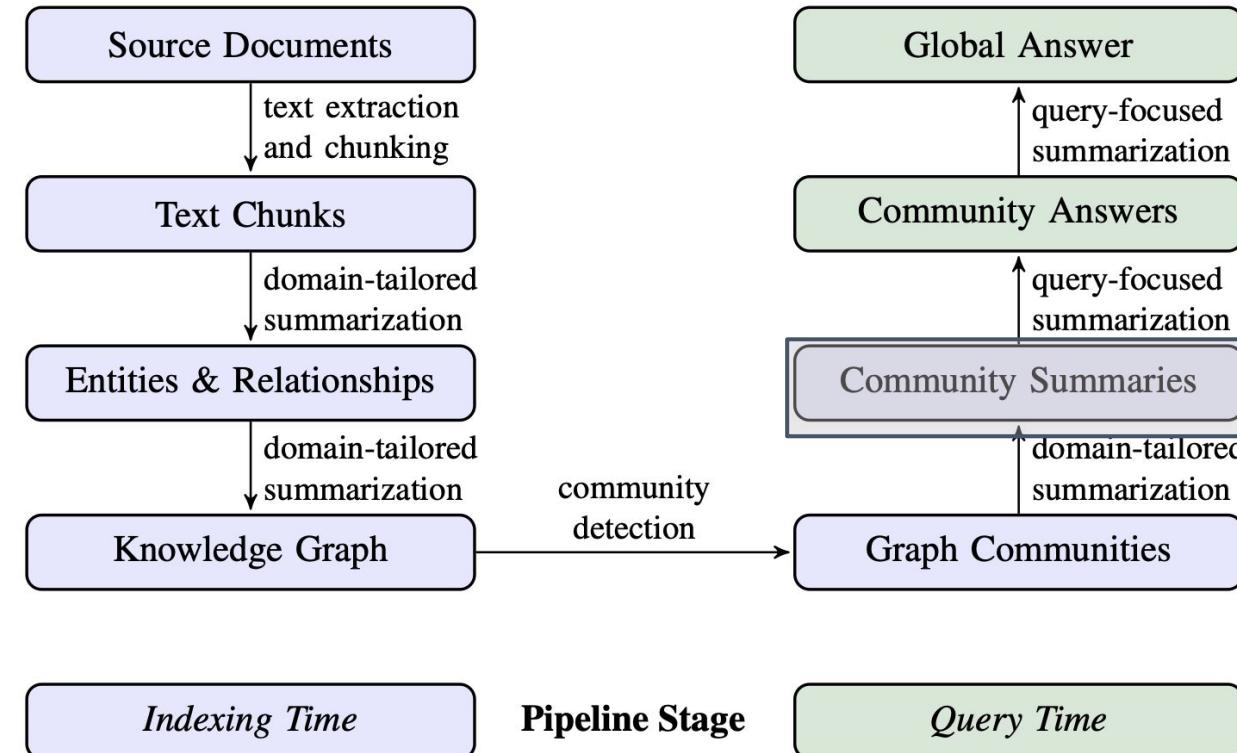


GraphRAG Pipeline: Community Summaries

GraphRAG genera resúmenes de comunidades agregando distintos resúmenes de elementos (para nodos y bordes) a una plantilla de resumen.

Los resúmenes de comunidades de niveles inferiores se utilizan para generar los resúmenes de comunidades de niveles superiores de la siguiente manera:

- **Comunidades de nivel hoja:** Se priorizan los elementos más importantes de la comunidad (según la relevancia de sus nodos y bordes) y se van agregando al contexto del modelo hasta llegar al límite de tokens. Para cada conexión, se incluyen las descripciones de los nodos implicados y la relación entre ellos.
- **Comunidades de nivel superior:** Si los resúmenes de todos los elementos caben en el límite de tokens, se procesan igual que las comunidades de nivel hoja. Si no, se ordenan las subcomunidades por tamaño y se reemplazan sus resúmenes detallados por versiones más cortas hasta que todo quepa en la ventana de contexto.



GraphRAG Pipeline: Community y Global Answers

La estructura jerárquica de las comunidades identificadas significa que las preguntas pueden ser contestadas mediante un resumen de distintos niveles.

La respuesta global a cualquier query se genera de la siguiente manera:

- **Preparar los resúmenes de comunidad:**

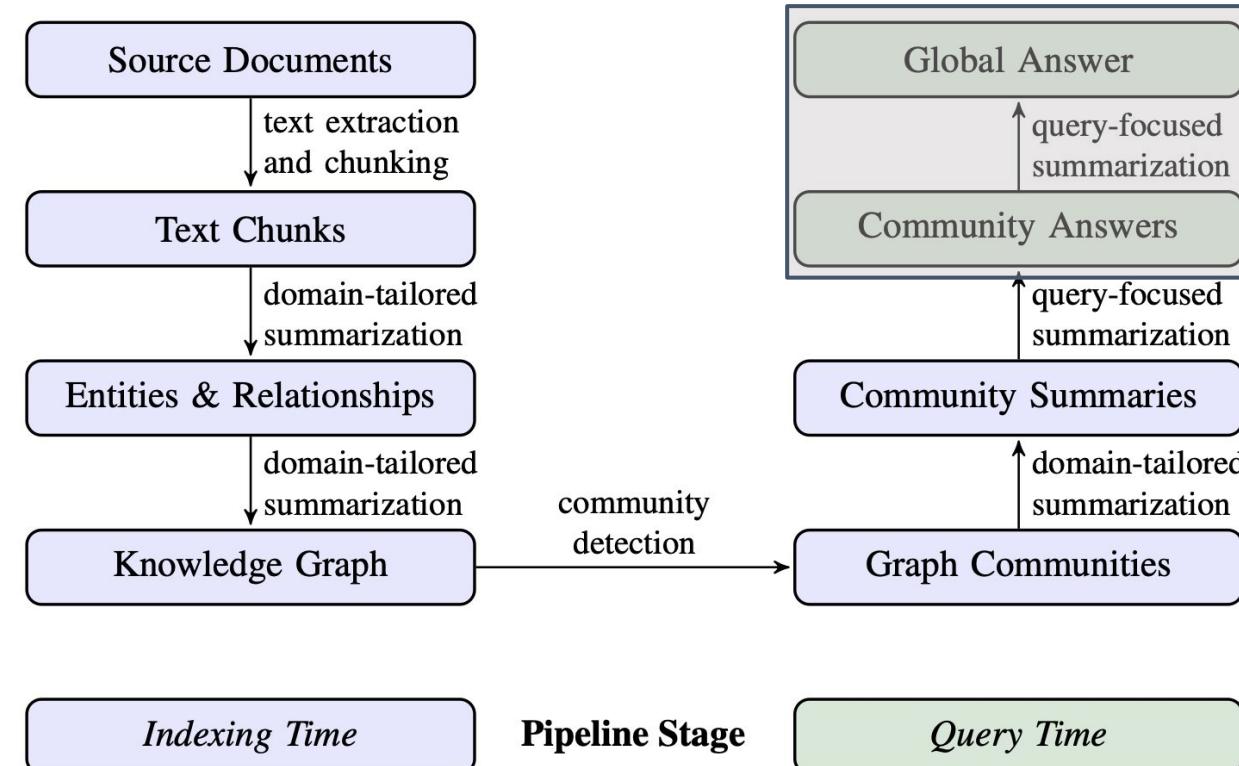
Los resúmenes se mezclan aleatoriamente y se dividen en fragmentos de un tamaño de tokens predefinido. Esto garantiza que la información relevante se distribuya entre los fragmentos, en lugar de concentrarse (y potencialmente perderse) en una sola ventana de contexto.

- **Mapear las respuestas de comunidad:**

Se generan respuestas intermedias en paralelo. Además, el LLM asigna una puntuación del 0 al 100 indicando qué tan útil es la respuesta generada. Las respuestas con puntuación 0 se descartan.

- **Reducir a una respuesta global:**

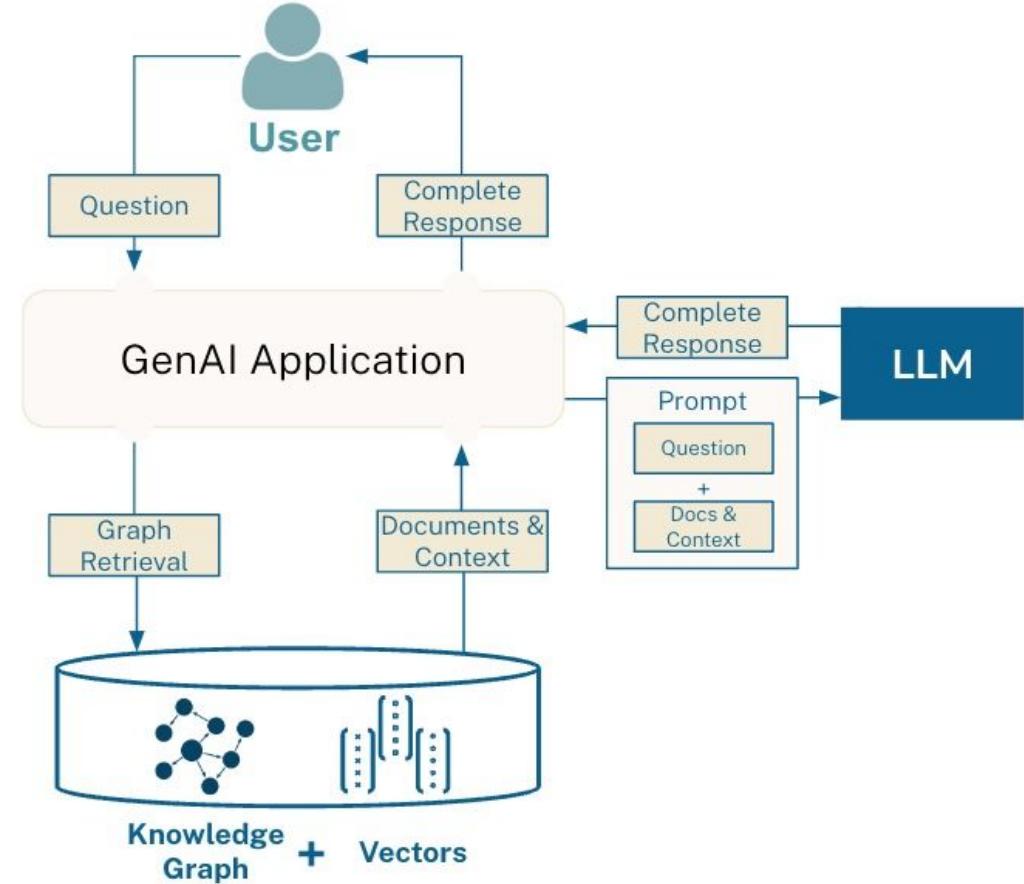
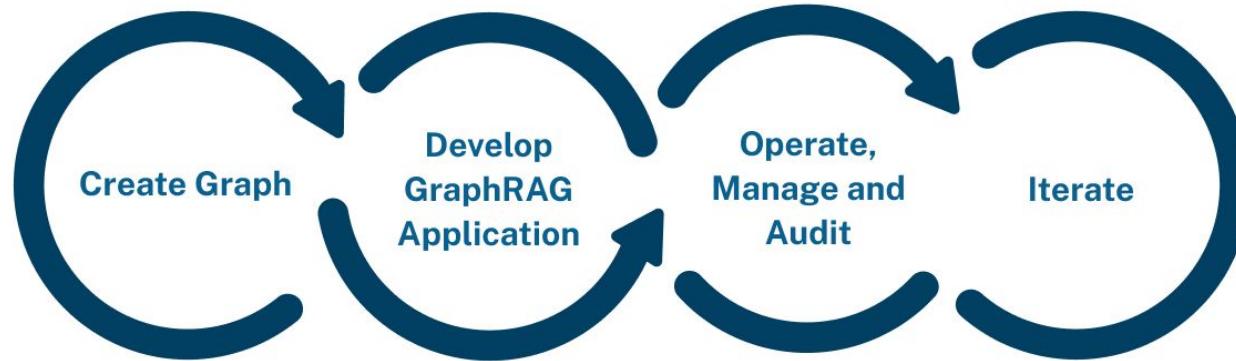
Las respuestas intermedias de las comunidades son ordenadas de manera descendente según su puntuación de utilidad y se agregan iterativamente a una nueva ventana de contexto hasta alcanzar el límite de tokens. Este contexto final se utiliza para generar la respuesta global.



GraphRAG no sustituye al RAG

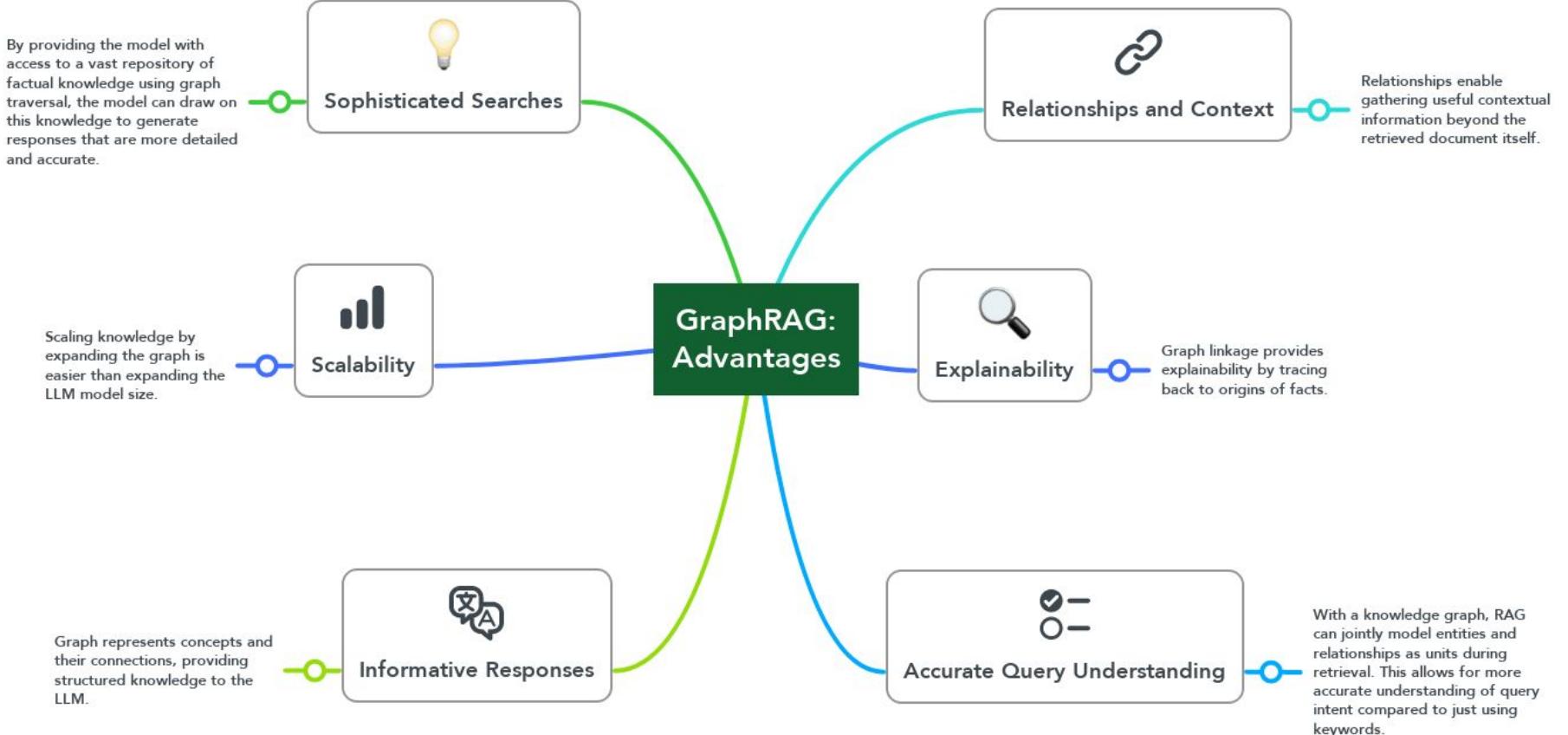
GraphRAG si bien implica obtener respuestas más robustas y cercas de la realidad, realmente en producción se integran los dos como contexto. En resumen el **retriever = KG + VectorRAG**.

Neo4j: Vectors + KG



GraphRAG

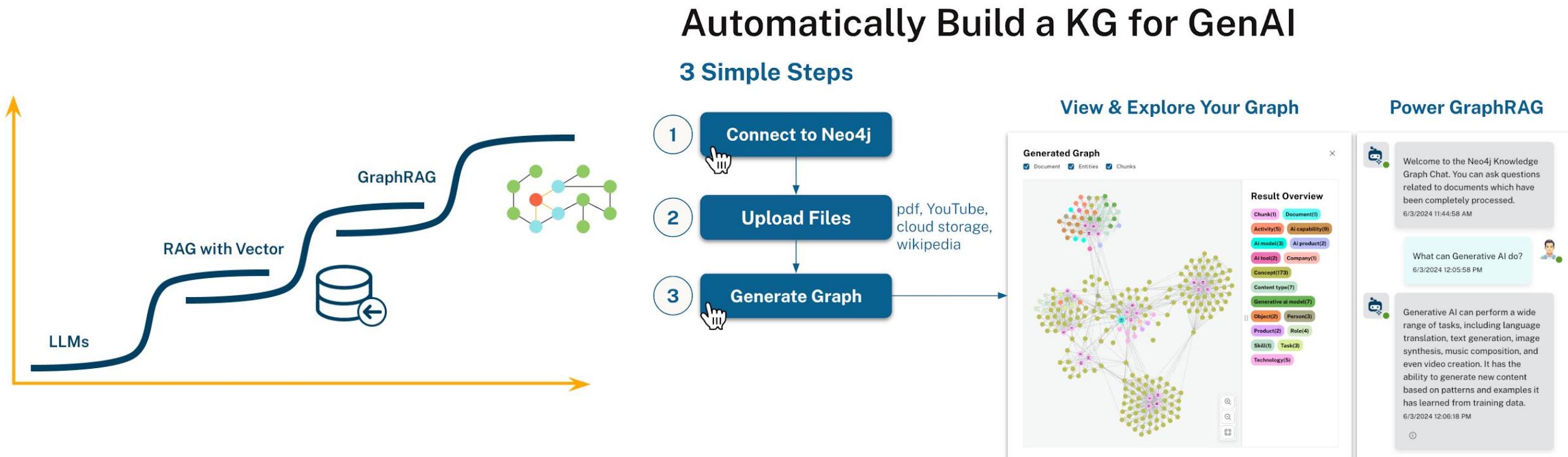
- [GraphRAG: Design Patterns, Challenges, Recommendations](#)
- [HybridRAG](#)
- [MedicalGraphRAG](#)
- [Microsoft GraphRAG](#)



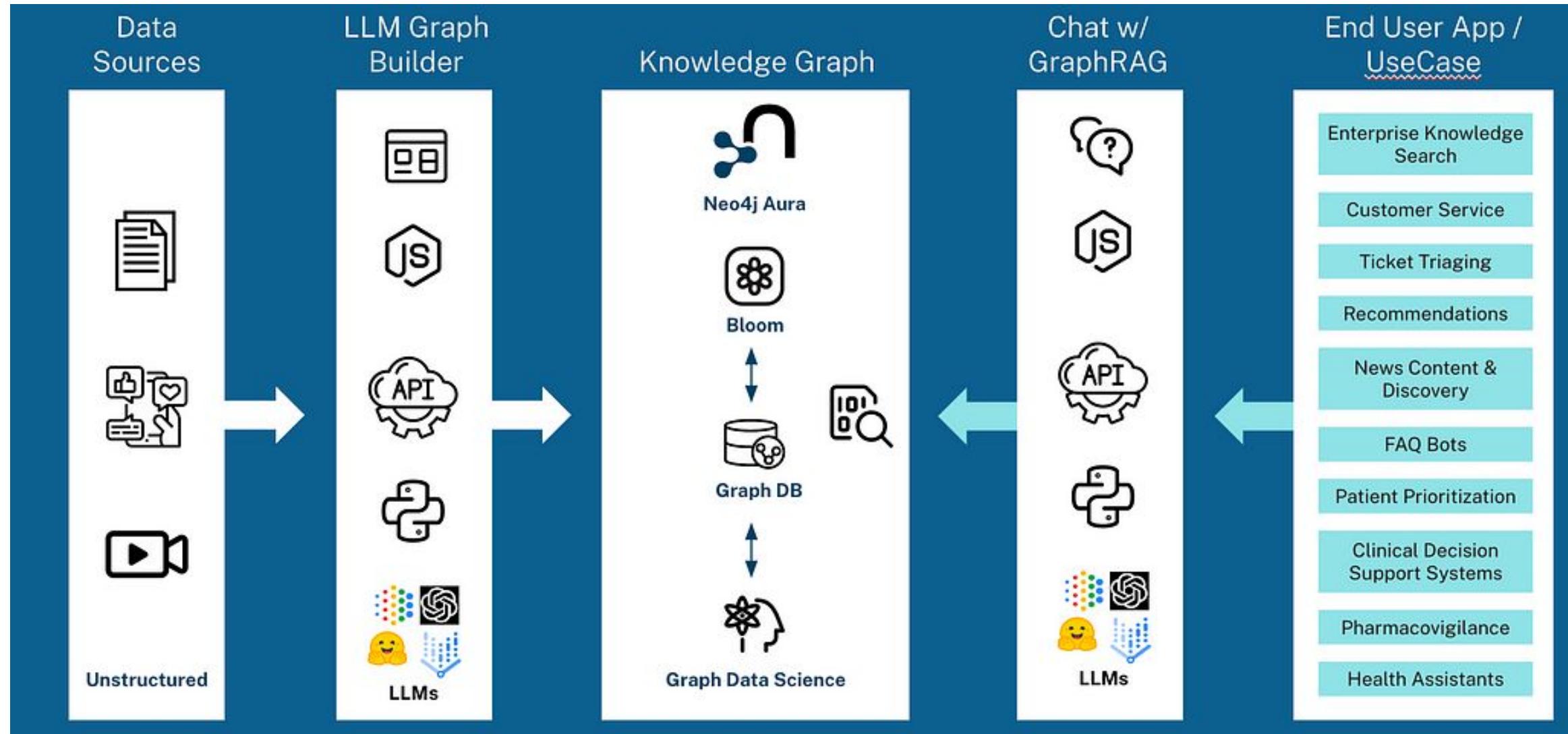
GraphRAG: Neo4J

Neo4J provee herramientas y [guías](#) para crear aplicaciones GraphRAG de manera bastante simple.

[The GraphRAG Manifesto: Adding Knowledge to GenAI](#)



GraphRAG: Neo4J



GraphRAG: AWS Neptune

[AWS](#) también ofrece servicios para crear aplicaciones GraphRAG de manera fácil.

GraphRAG

Existen variantes del GraphRAG como:

[Fast Think-on-Graph \(FToG\)](#)

[FastGraphRAG](#)

[LightRAG](#)

[FastGraphRAG Benchmarks](#)

También en el espacio multimodal:

[MMGraphRAG: Bridging Vision and Language with Interpretable Multimodal Knowledge Graphs](#)

[mKG-RAG: Multimodal Knowledge Graph-Enhanced RAG for Visual Question Answering](#)

[Neo4J: Advancing GraphRAG for multimodal intelligence](#)

Sistemas Multiagentes

Legacy y Modernos

Sistemas Multiagentes (MAS)

Un **sistema multiagente (MAS)** es un conjunto de **agentes autónomos** que interactúan dentro de un mismo entorno para **colaborar, coordinar o competir** con el fin de alcanzar metas individuales o colectivas.

A diferencia de los sistemas centralizados, los MAS se basan en **control y toma de decisiones distribuidos**, lo que les otorga **mayor precisión, adaptabilidad y escalabilidad**. Este enfoque representa un cambio de paradigma en la IA:

Pasar de soluciones únicas a **redes descentralizadas de agentes inteligentes** que trabajan en conjunto para resolver problemas complejos.

Los **sistemas multiagente (MAS)** funcionan distribuyendo tareas entre agentes autónomos que colaboran en un entorno común.

Cada agente **percibe** su entorno, **razona y decide**. Hoy en día el **LLM** funciona como “cerebro”, **actúa** según su plan y **se comunica o coordina** con otros agentes.

Los MAS modernos emplean **orquestación**, dividiendo tareas complejas en flujos estructurados donde un **orquestador** o **grafo** gestiona el orden, la comunicación y el cumplimiento de los objetivos, como en frameworks actuales tipo CrewAI o *LangGraph*.

Sistemas Multiagentes (MAS)

Para colaborar, los agentes necesitan **comunicarse** entre sí mediante **protocolos de comunicación**, que definen cómo se intercambia la información: el formato de los mensajes (por ejemplo, JSON, JSON-RPC, XML, etc.) y el método de envío (como HTTP, MQTT, gRPC, etc.).

Los **lenguajes de comunicación de agentes (ACL)**, como **FIPA ACL** y **KQML (Legacy), A2A, etc.** Ofrecen un estándar para que los agentes interactúen y compartan información detallada.

Por ejemplo, **FIPA ACL** se basa en la comunicación humana, usando “acciones” específicas como *request* o *inform*, e incluye campos para remitente, destinatario, acción y contenido, lo que hace la comunicación clara y estructurada.

Los **mecanismos de coordinación** permiten a los agentes resolver conflictos, alinearse en objetivos y trabajar en equipo.
Ejemplos:

- **subastas de tareas, votaciones**
- sistemas tipo **contract nets**.

Sistemas Multiagentes: Beneficios

Los sistemas multiagente ofrecen varias ventajas frente a los sistemas tradicionales o de agente único:

- **Mejor resolución de problemas:** múltiples agentes especializados colaboran, combinando habilidades y perspectivas distintas para abordar desafíos más complejos.
- **Escalabilidad:** se pueden agregar más agentes sin afectar el rendimiento, permitiendo manejar mayores volúmenes de trabajo o datos.
- **Robustez y fiabilidad:** si un agente falla, los demás continúan funcionando, lo que hace al sistema más confiable en contextos críticos.
- **Flexibilidad y adaptabilidad:** los MAS pueden ajustarse ante nueva información o imprevistos sin requerir intervención humana constante.
- **Mayor velocidad y eficiencia:** al dividir tareas entre agentes que trabajan en paralelo, los MAS resuelven problemas más rápido y aprovechan mejor los recursos.
- **Aprendizaje colectivo:** los agentes comparten conocimientos y mejoran juntos, fortaleciendo la capacidad del sistema para evolucionar y optimizar su desempeño con el tiempo.

Sistemas Multiagentes: Desafíos

Difícil de gestionar: coordinar múltiples agentes independientes sin conflictos es complejo, y la dificultad crece a medida que se suman más.

Sobrecarga de comunicación: más agentes implican más mensajes, lo que puede ralentizar el sistema si no hay protocolos claros y eficientes.

Acciones inesperadas: la interacción entre agentes puede generar resultados imprevistos difíciles de anticipar y probar.

Riesgos de seguridad: en entornos con información sensible, agentes maliciosos podrían manipular datos, negarse a cooperar o filtrar información.

Complejidad de diseño y uso: requieren planificación cuidadosa y conocimiento en IA distribuida y protocolos de comunicación sólidos.

Alto costo operativo: el uso intensivo de LLMs (generalmente vía APIs) puede generar costos computacionales elevados al escalar.

Alucinaciones y falta de anclaje factual: los agentes basados en LLM pueden producir información incorrecta pero verosímil, lo que exige mecanismos de verificación de datos.

Depuración y evaluación complejas: el comportamiento no determinista y emergente dificulta rastrear errores y requiere herramientas avanzadas de *logging* y monitoreo.

MAS: Implementacion

Definir el problema y los objetivos	Especificar requerimientos del sistema como cada agente individual.
Diseñar los agentes	<ul style="list-style-type: none"> • Roles: definir las tareas de cada agente. • Capacidades: determinar qué pueden percibir, qué acciones pueden ejecutar y cómo toman decisiones. • Independencia: establecer el grado de autonomía de cada agente.
Modelar el entorno	Crear el espacio compartido donde los agentes operarán, incluyendo sus características, recursos y reglas.
Métodos de comunicación	<ul style="list-style-type: none"> • Lenguaje: Formato como <i>FIPA ACL (Legacy)</i>, A2A, etc. • Reglas: Diseñar como los agentes se comunican ya sea por mensajes directos, Pub/Sub, memoria compartida, etc.
Establecer estrategias de coordinación	Definir mecanismos para asegurar la colaboración y manejo de conflictos, ya sea mediante un agente principal, reglas de negociación o cooperación natural.
Pruebas y validación	Verificar que los agentes se comporten como se espera, cooperen correctamente y cumplan los objetivos generales, prestando atención a comportamientos imprevistos.
Desplegar y monitorear	Poner el sistema en producción y establecer monitoreo continuo para detectar fallos, medir desempeño y mantener su funcionamiento óptimo.

MAS: antes de las LLMs

Los sistemas MAS como concepto, preceden a las LLMs, el libro [MULTIAGENT SYSTEMS: Algorithmic, Game-Theoretic, and Logical Foundations](#) (Shoham & Leyton-Brown, 2009) muestra los fundamentos de sistemas MAS, medios de comunicación, etc.

Si bien estos sistemas son completamente diferentes gracias a las LLMs, fundamentalmente el concepto se mantiene.

En 2001, el Java Framework [JADE](#) para la creación de sistemas MAS fue publicado. Su arquitectura es muy similar conceptualmente a lo que hoy encontramos en aplicaciones de agentes con LLMs. Su comunicación basada en FIPA ACL.

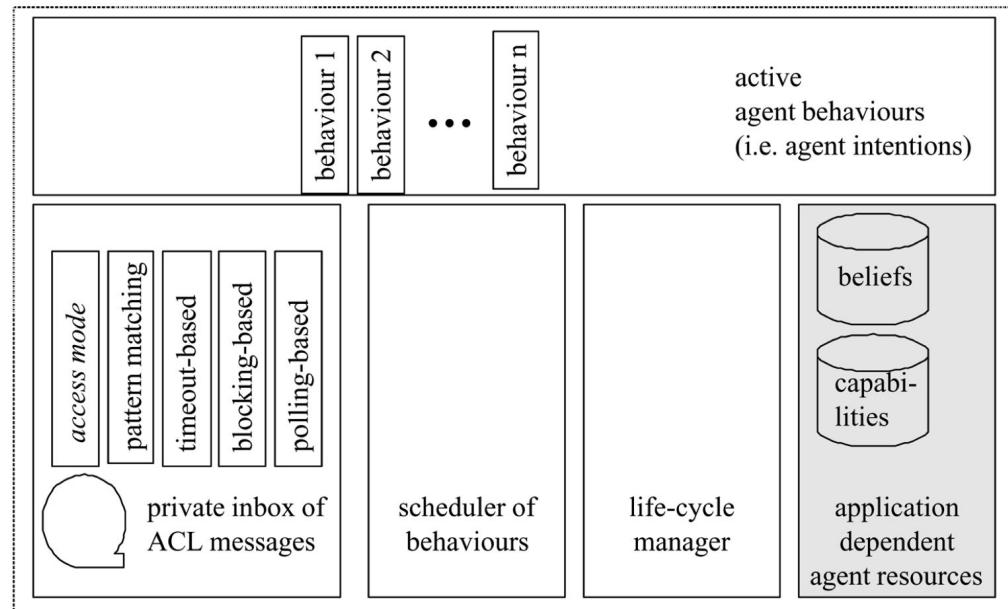


Figure 3. JADE agent architecture.

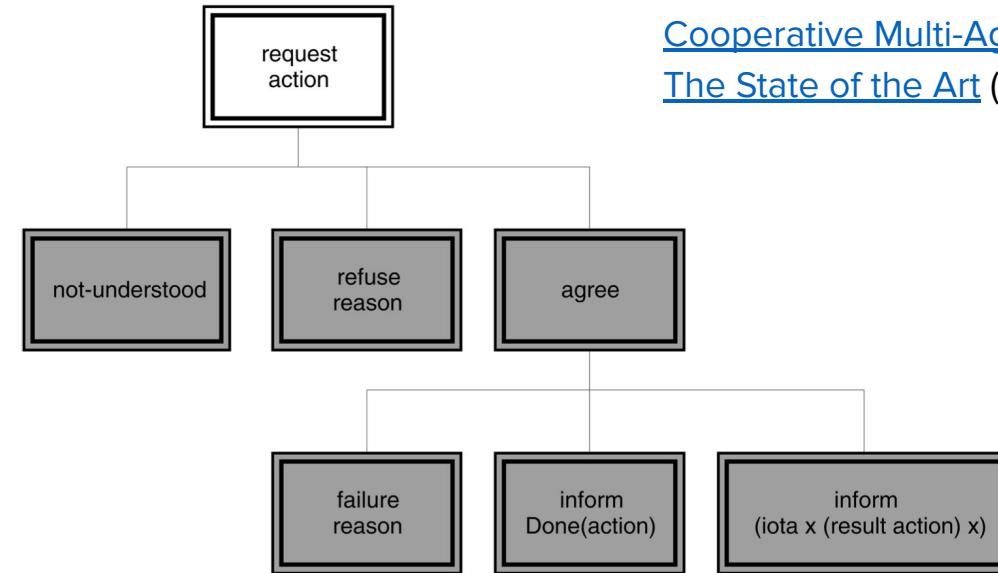


Figure 5. FIPA request interaction protocol.

MAS: antes de las LLMs

[Cooperative Multi-Agent Learning: The State of the Art](#) (2005)

[Stanford: CS 224M](#)

MAS: Actualidad

Las LLMs cambiaron a los sistemas MAS sin embargo, aún se encuentra en investigación, ejemplo:

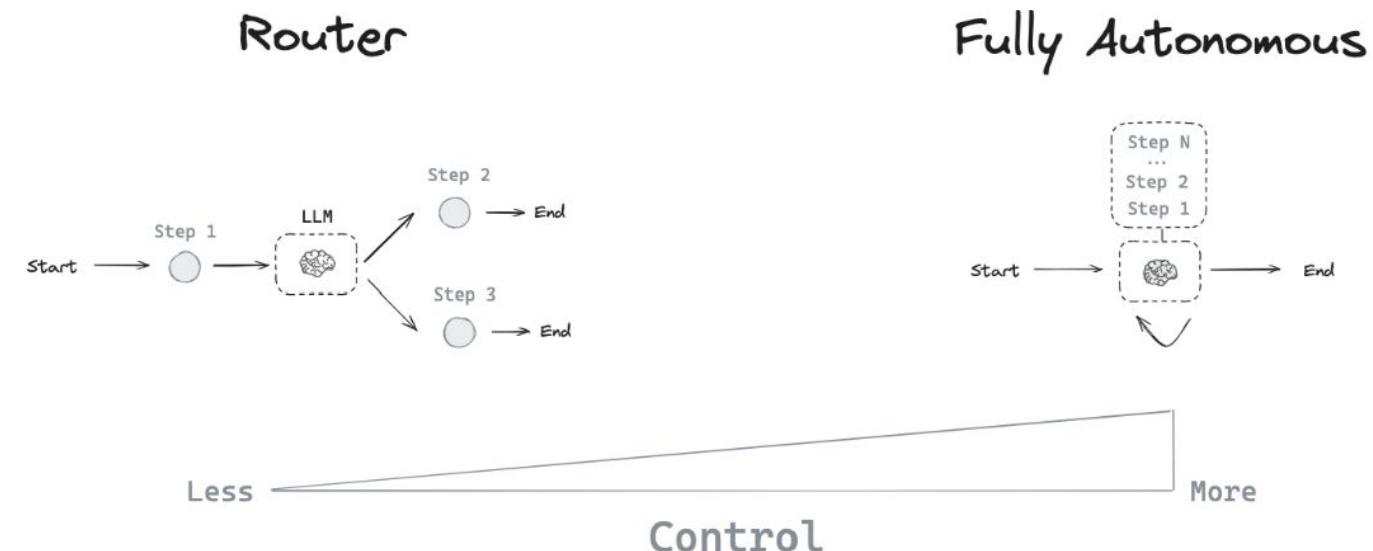
[Multi-Agent MicroServices \(MAMS\)](#)

Frameworks como [CAMEL](#) (competencia de LangGraph)

[CAMEL Github](#)

[MASTER: A Multi-Agent System with LLM Specialized MCTS](#)

[SiriuS: Self-improving Multi-agent Systems via Bootstrapped Reasoning](#)



Preguntas?