UNIVERSIDAD NACIONAL DE CATAMARCA

FACULTAD DE TECNOLOGÍA Y CIENCIAS APLICADAS

Título: Agente de Gestión Probabilística de Contextos (AGPC)

INGENIERÍA EN INFORMÁTICA

Trabajo Final

- **DIRECTOR:**
  - Vilallonga Gabriel

- **ALUMNO/AUTOR:** De La Fuente Gonzalo Uriel
- **MU:** 01561

Fecha de presentación: Diciembre de 2025.

Agradecimientos

A mis padres, quienes han sido mi guía y sostén en esta vida. Gracias por confiar en mí, por su apoyo incondicional y por enseñarme con su ejemplo el valor del esfuerzo y la perseverancia. Gracias a ustedes, hoy puedo cumplir uno de mis sueños más anhelados: convertirme en Ingeniero en Informática, cerrando así una etapa fundamental de mi vida.

A mi pareja, que con amor, paciencia y comprensión me acompañó en las últimas etapas de esta carrera. A ella y a su familia, por su apoyo constante, por su cariño y por entender mis ausencias cuando el estudio requería toda mi atención. Su comprensión y palabras de aliento fueron un motor indispensable para alcanzar esta meta tan importante.

A mis abuelos y a toda mi familia, quienes desde siempre me inculcaron el valor del estudio y me acompañaron con su afecto y buenos deseos. Cada uno de ustedes ha dejado una huella imborrable en este camino.

A mis compañeros, con quienes compartí aprendizajes, desafíos y momentos inolvidables. Gracias por la colaboración, el compañerismo y por ser parte de este recorrido que quedará siempre en mi memoria.

A mis profesores, por transmitir su conocimiento y su pasión por la informática y la ingeniería. Gracias por despertar en mí la curiosidad, la creatividad y el entusiasmo por seguir aprendiendo cada día.

Y, finalmente, a mi director de trabajo final, por su guía, compromiso y apoyo en cada etapa de este proyecto. Su dedicación y orientación fueron fundamentales para alcanzar los objetivos propuestos.

A todos ustedes, mi más sincero agradecimiento.

Resumen

En este trabajo se propone el diseño y desarrollo de un Agente de Gestión Probabilística de Contextos (AGPC), un sistema inteligente orientado a la administración avanzada de información, utilizando un Grafo Contextual Probabilístico (GCP) como núcleo arquitectónico.

A diferencia de los enfoques lineales y de memoria estática presentes en la mayoría de asistentes actuales, el AGPC desarrollado implementa un modelo no lineal, difuso y reutilizable de contexto, capaz de establecer y ponderar relaciones probabilísticas entre fragmentos de información o contextos atómicos distribuidos en el tiempo y el espacio.

El sistema combina técnicas de Procesamiento de Lenguaje Natural (PLN), Modelos de Lenguaje de Gran Escala (LLMs), bases vectoriales y bases de datos de grafos, permitiendo la recuperación semántica directa e indirecta de información relevante.

El diseño propuesto busca garantizar la adaptabilidad a múltiples dominios (auditorías financieras, educación personalizada, entornos legales, médicos, entre otros), así como la escalabilidad para flujos de información en tiempo real.

Si bien la arquitectura propuesta es conceptualmente generalizable a diversos contextos, para efectos de este trabajo final se selecciona el dominio legal como caso de aplicación principal. Este dominio permite validar empíricamente el funcionamiento del prototipo y establecer un alcance definido y acotado dentro del marco de la investigación, dado que su complejidad documental, temporal y contextual ofrece un entorno idóneo para evaluar la efectividad del modelo propuesto.

El objetivo final de este trabajo es demostrar que un enfoque probabilístico y auditable de la gestión contextual no solo mejora la coherencia y trazabilidad de las respuestas de un agente inteligente, sino que también amplía su capacidad de transferencia de conocimiento entre distintos dominios.

Índice

Título del Trabajo Final: Agente de Gestión Probabilística de Contextos (AGPC)

1. Introducción

1.1 Planteamiento del Problema

En el ámbito de la inteligencia artificial aplicada a la interacción humano-máquina, la **gestión del contexto** se ha consolidado como un factor crítico para garantizar la coherencia, pertinencia y trazabilidad de las respuestas generadas por los modelos de lenguaje.

Los **Modelos de Lenguaje de Gran Escala (LLMs)** han experimentado una evolución acelerada en cuanto a capacidad de contexto, especialmente durante el período de desarrollo de este trabajo final. Al momento de iniciar esta investigación en julio de 2025, el panorama de modelos de lenguaje presentaba las siguientes capacidades:

Modelos disponibles al inicio del trabajo (julio 2025):

- Claude 3.5 Sonnet (Anthropic, lanzado junio 2024): 200.000 tokens
- GPT-4 Turbo (OpenAI, actualizado abril 2024): 128.000 tokens
- Gemini 1.5 Pro (Google DeepMind, febrero 2024): hasta 2 millones de tokens
- Llama 3 (Meta, abril 2024): 8.000 tokens (versión base)

Durante el desarrollo de la presente investigación (julio-octubre 2025), la industria lanzó modelos significativamente más capaces:

- GPT-4o (OpenAI, mayo 2024): 128.000 tokens con procesamiento multimodal mejorado
- Claude Opus 4 (Anthropic, agosto 2025): 500.000 tokens de contexto
- Claude Sonnet 4.5 (Anthropic, septiembre 2025): 1 millón de tokens
- Gemini 2.0 Flash (Google DeepMind, agosto 2025): 1 millón de tokens con latencia reducida
- Gemini 2.5 Pro (Google DeepMind, octubre 2025): 2 millones de tokens con capacidades de razonamiento mejoradas
- GPT-5 (OpenAI, octubre 2025): 1 millón de tokens con arquitectura optimizada para contextos largos

Esta rápida evolución durante apenas tres meses de investigación demuestra la aceleración tecnológica del campo. Sin embargo, y esto es fundamental para justificar el presente trabajo, **esta expansión cuantitativa no resuelve el problema cualitativo de la gestión contextual**. Incluso con ventanas de 1-2 millones de tokens, persisten las siguientes limitaciones fundamentales:

1. **Costo computacional prohibitivo:** Procesar ventanas completas de contexto incrementa exponencialmente el costo de inferencia. OpenAI reporta que el procesamiento de contextos completos de 1M+ tokens puede costar entre 10-50

USD por consulta en modelos de última generación, haciendo inviable el uso continuo de ventanas máximas en aplicaciones productivas (OpenAI, 2024, 2025).

2. **Degradación de rendimiento con contextos largos:** Estudios recientes demuestran que los LLMs pierden hasta 40% de precisión cuando la información relevante está "enterrada" en el medio de contextos muy largos, fenómeno conocido como "lost in the middle" (Liu et al., 2024). Esta degradación se observa incluso en modelos con ventanas de millones de tokens, donde la precisión cae significativamente cuando el contexto fundamental está entre los tokens 300.000 y 700.000.
3. **Imposibilidad práctica de contexto infinito:** Conversaciones de meses o años, con miles de documentos distribuidos temporalmente, superan cualquier ventana técnicamente viable. Un estudio jurídico con 10 años de casos acumula fácilmente más de 100 millones de tokens de información crucial. Incluso con GPT-5 y su ventana de 1 millón de tokens, esto representa apenas el 1% de la información total disponible.
4. **Ausencia de estructura probabilística y temporal:** Las ventanas actuales son lineales y estáticas. Ninguno de los modelos mencionados modela relaciones indirectas entre fragmentos de información. Toda la información dentro de la ventana se trata con igual peso, independientemente de su antigüedad o pertinencia contextual.
5. **Problema de selección:** ¿Qué información incluir en la ventana? Incluso con ventanas de 2 millones de tokens, alguien o algo debe decidir qué contextos cargar. Los sistemas actuales usan heurísticas simples (reciente equivale a relevante) que fallan en escenarios complejos donde información antigua puede ser crítica.
6. **Latencia y experiencia de usuario:** Modelos con ventanas muy amplias experimentan latencias significativas. Claude Opus 4 con 500.000 tokens de contexto puede tardar 8-15 segundos en responder consultas complejas, mientras que GPT-5 con 1 millón de tokens puede superar los 20 segundos en casos extremos.

En la práctica, muchos asistentes actuales se basan en estrategias lineales de almacenamiento y recuperación de información, como simples historiales de conversación o repositorios estáticos, careciendo de mecanismos para descubrir relaciones indirectas o ajustar dinámicamente la relevancia de cada fragmento de información.

Pregunta de investigación: ¿Es posible diseñar un sistema de gestión contextual que, mediante un grafo probabilístico con decaimiento temporal explícito y propagación de activación, supere las limitaciones de los sistemas RAG estándar en la recuperación de información distribuida temporalmente?

1.2 Justificación e Importancia del Trabajo

Para superar las limitaciones identificadas, se propone en este trabajo el **Agente de Gestión Probabilística de Contextos (AGPC)**, un sistema que implementa un **Grafo Contextual Probabilístico (GCP)** como mecanismo central de organización del

conocimiento. En este grafo, cada nodo representa un fragmento de contexto (documento, interacción, evento o conversación) y cada arista modela una relación probabilística que integra factores de similitud semántica, co-ocurrencia y proximidad temporal.

El enfoque sugerido no solo permite una recuperación más rica y flexible, sino que también facilita el **mecanismo para la retención o descarte selectivo de información**, la **transferencia de aprendizajes entre dominios** y la **auditoría explícita de las relaciones contextuales**.

La relevancia de este trabajo se fundamenta en los siguientes aspectos:

Brecha identificada en el estado del arte: Ninguno de los enfoques existentes combina simultáneamente:

- Modelado probabilístico de relaciones contextuales
- Decaimiento temporal explícito y configurable
- Propagación de activación para descubrimiento de relaciones indirectas
- Auditoría y explicabilidad de decisiones de recuperación

Necesidad documentada: Estudios recientes (Naveed et al., 2023; Perez et al., 2023) demuestran que la gestión efectiva del contexto es el factor determinante para el desempeño de LLMs en tareas de múltiples turnos. Según Katz et al. (2023) y Bommarito & Katz (2024), la gestión del conocimiento legal representa uno de los mayores costos operativos en bufetes y departamentos jurídicos, con abogados dedicando hasta el 30% de su tiempo a búsqueda de información ya disponible en la organización.

Contribución arquitectónica: El AGPC aborda los problemas identificados no compitiendo en tamaño de ventana, sino proponiendo un sistema de gestión contextual externa, probabilística y estructurada mediante grafos, que funciona como una "memoria de largo plazo" complementaria a la ventana de contexto inmediata del LLM. El sistema decide qué información es verdaderamente relevante para cada consulta, considerando tanto similitud semántica como proximidad temporal y relaciones indirectas.

Relevancia creciente: Mientras las ventanas de contexto crecen aceleradamente (de 200K a 2M tokens en solo 18 meses entre marzo 2024 y octubre 2025), la necesidad de gestionarlas inteligentemente se intensifica. Un sistema con 2 millones de tokens de ventana, pero sin gestión probabilística, es como una biblioteca sin índice: tiene toda la información, pero no puede encontrar lo fundamental eficientemente.

La evolución tecnológica observada durante el desarrollo de este trabajo (julio-diciembre 2025) válida la hipótesis central: **el problema no es la falta de capacidad de memoria, sino la falta de inteligencia en su gestión**. Los lanzamientos sucesivos de modelos con mayor capacidad de contexto aumentaron la capacidad de memoria, pero, según la documentación técnica disponible, no incorporaron mecanismos explícitos de priorización probabilística, decaimiento temporal configurable o descubrimiento de relaciones indirectas.

Este trabajo se enmarca en la **Ingeniería de Contexto**, disciplina que estudia cómo seleccionar, aislar, comprimir y escribir información para optimizar el uso de la ventana de contexto en sistemas inteligentes. Desde esta perspectiva, el AGPC representa una evolución de las técnicas actuales, integrando grafos probabilísticos para maximizar el rendimiento de los LLMs en entornos complejos y multidominio.

1.3 Alcance del Trabajo

El presente trabajo final define los siguientes límites respecto a su extensión y profundidad:

Incluye:

- Diseño e implementación de un prototipo funcional del AGPC con arquitectura modular de seis capas
- Formalización matemática del Grafo Contextual Probabilístico (GCP) con funciones de peso, decaimiento temporal y propagación de activación
- Construcción de un dataset de validación compuesto por 200 conversaciones legales sintéticas distribuidas temporalmente entre enero y agosto de 2023
- Evaluación comparativa sistemática con un sistema RAG estándar utilizando 6 métricas definidas (4 automáticas y 2 manuales)
- Implementación de optimizaciones de rendimiento (actualización incremental, batch processing, normalización sigmoidea)
- Documentación completa, incluyendo código fuente, manual de usuario, protocolo de evaluación y anexos técnicos

Excluye:

- Validación con datos reales de casos legales (se utilizan exclusivamente datos sintéticos por consideraciones éticas y de accesibilidad)
- Evaluación en múltiples dominios de aplicación (la validación se concentra únicamente en el dominio legal)
- Despliegue en ambiente de producción o evaluación de escalabilidad a nivel empresarial
- Soporte multilingüe (el prototipo está optimizado para español)
- Integración con sistemas de gestión documental externos

Dominio de aplicación: Si bien la arquitectura propuesta es conceptualmente generalizable a diversos contextos, para efectos de este trabajo final se selecciona el **dominio legal** como caso de aplicación principal. Este dominio permite validar empíricamente el funcionamiento del prototipo y establecer un alcance definido y acotado dentro del marco de la investigación, dado que su complejidad documental, temporal y contextual ofrece un entorno idóneo para evaluar la efectividad del modelo propuesto.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un prototipo funcional de un Agente de Gestión Probabilística de Contextos (AGPC) que gestione información contextual mediante un Grafo Contextual Probabilístico, mejorando la coherencia, trazabilidad y transferencia de conocimiento en agentes inteligentes, y validar su desempeño mediante comparación experimental con sistemas RAG estándar en el dominio legal.

1.4.2 Objetivos Específicos

1. **Diseñar la arquitectura modular del AGPC** con separación de responsabilidades por capas funcionales, incluyendo módulos de extracción semántica, memoria vectorial, grafo contextual probabilístico, motor de respuesta e interfaz de usuario.
2. **Implementar el modelo de Grafo Contextual Probabilístico** integrando similitud semántica, léxica y temporal, con normalización probabilística de pesos efectivos y algoritmos de propagación de activación para descubrimiento de relaciones indirectas.
3. **Evaluar comparativamente el rendimiento del AGPC frente a sistemas RAG estándar** mediante un protocolo experimental que incluya métricas de tiempo de respuesta, cobertura temporal, profundidad de propagación, contextos indirectos recuperados, precisión de recuperación contextual y coherencia de respuesta.
4. **Validar la arquitectura del AGPC en el dominio legal** utilizando un dataset de 200 conversaciones sintéticas que abarquen diversos tipos de casos (amparo por mora administrativa, divorcios, trabajo no registrado, acoso laboral, incumplimiento contractual, entre otros) distribuidos temporalmente a lo largo del año 2023.

Nota sobre la vigencia tecnológica:

Como se detalló en el análisis de la evolución reciente presentado en la sección 1.1, la aceleración tecnológica observada durante el desarrollo de este trabajo (julio-diciembre 2025) confirma que el desafío principal es metodológico y no de capacidad bruta. Por consiguiente, las referencias al 'estado del arte' y a 'modelos avanzados' a lo largo de este documento deben interpretarse dentro del contexto temporal de dicha ventana de investigación.

2. Marco teórico

El presente capítulo establece los fundamentos conceptuales y técnicos que sustentan el desarrollo del **Agente de Gestión Probabilística de Contextos (AGPC)**. En primer lugar, se aborda la disciplina de la Ingeniería de Contexto y las limitaciones actuales de los Modelos de Lenguaje de Gran Escala (LLMs). Posteriormente, se profundiza en la definición formal de los **Grafos Contextuales Probabilísticos**, detallando la formulación matemática para la gestión de la dimensión temporal, las estrategias de fragmentación y las optimizaciones implementadas. Finalmente, se realiza un análisis comparativo del estado del arte para situar la propuesta frente a soluciones existentes.

2.1 Ingeniería de Contexto

La **Ingeniería de Contexto** es la disciplina que se ocupa de gestionar de manera óptima la información que un **Modelo de Lenguaje de Gran Escala (LLM)** puede utilizar en un momento dado, considerando las limitaciones de su **ventana de contexto**(LangChain AI, 2025), podemos decir que es análoga a la memoria de trabajo en sistemas computacionales.

En este paradigma, el Modelo de Lenguaje de Gran Escala (LLM) actúa como la CPU, mientras que la ventana de contexto cumple el rol de RAM, almacenando temporalmente la información necesaria para el procesamiento inmediato, **una analogía formalizada en la**

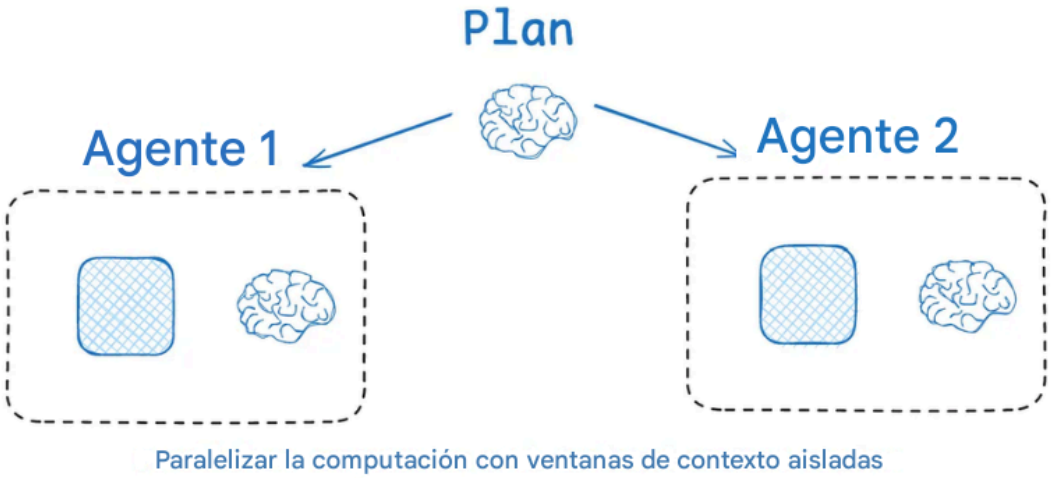
arquitectura de sistemas como MemGPT (Packer et al., 2023). Al igual que en un sistema operativo, donde se decide qué datos cargar en memoria para optimizar el rendimiento, la Ingeniería de Contexto define qué fragmentos de información se incorporan, mantienen, descartan o reestructuran en cada ciclo de procesamiento del agente.

Las principales **estrategias** para la gestión eficiente de contexto incluyen:

1. **Escribir contexto (Write Context)**: almacenar información relevante fuera de la ventana activa, por ejemplo, en bases de datos, repositorios vectoriales o memorias persistentes, para su recuperación futura.

Un LLM se puede utilizar para actualizar o crear memorias

2. **Seleccionar contexto (Select Context)**: identificar y traer de vuelta al contexto activo los fragmentos más significativos para la tarea en curso, utilizando técnicas como **RAG (Retrieval-Augmented Generation)**, embeddings y grafos de conocimiento.
3. **Comprimir contexto (Compress Context)**: reducir la cantidad de información para que encaje en la ventana de contexto, mediante técnicas como resumen automático (map-reduce, summarization) o poda de tokens(técnica para **reducir la cantidad de texto que un modelo procesa**).
4. **Aislar contexto (Isolate Context)**: segmentar la información entre diferentes agentes o procesos para evitar sobrecarga y optimizar el rendimiento, aplicando arquitecturas multiagente y entornos aislados.



Dividir el contexto entre múltiples agentes

Estas estrategias buscan mitigar problemas recurrentes como la **alucinación** (generación de información ficticia), el **choque de contexto** (conflictos entre fragmentos de información) y la **pérdida de relevancia** en interacciones prolongadas.

El término "Ingeniería de Contexto" y las estrategias asociadas (Write, Select, Compress, Isolate) fueron formalizados inicialmente en el contexto de sistemas basados en LLMs por LangChain AI (2025). En el presente trabajo, se ha extendido este marco conceptual para incluir gestión probabilística mediante grafos y decaimiento temporal, componentes ausentes en los enfoques tradicionales de RAG y recuperación vectorial.

2.2. Modelos de Lenguaje de Gran Escala (LLMs)

Los **Modelos de Lenguaje a Gran Escala (LLMs, por sus siglas en inglés)** son sistemas basados en inteligencia artificial diseñados para procesar, comprender y generar texto en lenguaje humano con alto nivel de coherencia. Estos modelos, entrenados con enormes volúmenes de datos, son la base de asistentes virtuales, chatbots avanzados y otras aplicaciones de procesamiento de lenguaje natural (NLP).

Podemos mencionar algunos LLMs representativos, cada uno con ventajas y limitaciones según el contexto de uso:

LLaMA 3 (Meta AI)

- Ventajas:
  - Código abierto: Permite modificaciones y adaptaciones específicas.
  - Eficiencia computacional: Requiere menos recursos que modelos propietarios como GPT-4, facilitando su despliegue en entornos locales.

    - Flexibilidad: Ideal para fine-tuning (ajuste fino) en dominios especializados.
- Limitaciones:
    - Su desempeño depende en gran medida de la calidad y cantidad de los datos de entrenamiento.
    - No supera a GPT-4 en tareas que demandan alto razonamiento o creatividad.

GPT-4 (OpenAI)

- Ventajas:
    - Alto rendimiento: Destaca en generación de texto multilingüe, creatividad y resolución de problemas complejos.
    - Generalista: Funciona bien en múltiples dominios sin necesidad de ajustes previos.
    - Multimodalidad: Capacidad para procesar texto, imágenes y otros formatos de entrada y salida
- Limitaciones:
    - Modelo cerrado: No permite modificaciones internas ni entrenamientos personalizados directos.
    - Dependencia de API: Requiere conexión externa, lo que puede implicar latencia, costos recurrentes y restricciones de privacidad.

Gemini (Google DeepMind)

- Ventajas:
    - Multimodalidad: Capacidad nativa para procesar y generar texto, imágenes, y otros formatos.
    - Integración con ecosistema Google: Optimizado para tareas que requieren análisis combinado (ej. búsquedas enriquecidas o síntesis de información multimedia).
    - Enfoque en precisión: Diseñado para reducir alucinaciones (errores factuales) en comparación con otros Modelos de Lenguaje de Gran Escala (LLMs).
- Limitaciones:
    - Acceso restringido: Versiones avanzadas (ej. Gemini Ultra) tienen disponibilidad limitada frente a alternativas abiertas.
    - Computacionalmente demandante: Su versatilidad multimodal incrementa los requisitos de hardware en despliegues locales.

Estudios recientes (Naveed et al., 2023; Perez et al., 2023) demuestran que la gestión efectiva del contexto es el factor determinante para el desempeño de LLMs en tareas de múltiples turnos, validando la necesidad de sistemas como el AGPC. La evolución de modelos como Llama 3 (Meta AI Research, 2024) y GPT-4 (OpenAI, 2024) confirma la tendencia hacia ventanas de contexto más amplias, pero mantiene vigente el problema fundamental de cómo seleccionar y priorizar información relevante dentro de esas ventanas.

2.2.1 Memoria Contextual en Modelos de Lenguaje de Gran Escala

El problema de la memoria contextual en LLMs ha sido objeto de investigación activa en los últimos años. A continuación se presentan los principales enfoques y sus limitaciones:

Enfoques de memoria a corto plazo:

1. **Ventana de contexto fija (Transformer estándar):** Los modelos basados en la arquitectura Transformer (Vaswani et al., 2017) mantienen un contexto limitado durante la inferencia. Aunque modelos recientes como Claude 3.5 (Anthropic, 2024), Gemini 1.5 Pro (Google, 2024) y GPT-4 Turbo (OpenAI, 2024) han expandido significativamente estas ventanas con modelos más potentes (1-2 millones de tokens), persisten limitaciones fundamentales:
    - **Degradación de rendimiento con contextos largos:** Liu et al. (2024) demuestran que los LLMs pierden hasta 40% de precisión cuando la información relevante está en el medio de contextos muy largos.
    - **Costo computacional prohibitivo:** Procesar ventanas completas de 1M+ tokens tiene costo lineal o cuadrático según la arquitectura.
    - **Ausencia de priorización:** Todos los tokens en la ventana se procesan con igual peso, sin distinción de relevancia.
2. **Compresión de contexto:** Técnicas como resumen automático (Zhang et al., 2023) o poda de tokens (Li et al., 2024) buscan reducir el contexto manteniendo información fundamental. Por lo tanto, posee limitaciones:
    - **Pérdida de información:** El proceso de compresión es irreversible.
    - **Sesgo del compresor:** Qué se descarta depende del modelo de compresión.

Enfoques de memoria a largo plazo:

1. **Bases de datos vectoriales (RAG):** La Generación Aumentada por Recuperación (Lewis et al., 2020) almacena documentos en bases vectoriales y los recupera semánticamente. Sistemas como LangChain (Harrison, 2024) popularizan este enfoque. Limitaciones:
    - Recuperación puramente semántica: Ignora dimensión temporal.
    - Sin modelado de relaciones indirectas: Conexiones transitivas no se descubren.
    - Memoria plana: Sin estructura que refleje dependencias entre fragmentos.
2. **Memoria jerárquica (MemGPT):** MemGPT (Packer et al., 2023) propone una arquitectura de memoria en capas (corto/mediano/largo plazo) con un scheduler que decide qué cargar en contexto.

    Sus limitaciones son:

    - **Temporalidad implícita:** No existe modelo matemático de decaimiento.
    - **Dependencia del LLM:** Las decisiones de qué recordar son opacas.

3. **Grafos de conocimiento:** Sistemas como RET-LLM (Zhang & Liu, 2024) usan grafos para representar entidades y relaciones.

   Sus limitaciones son:

   - Requiere extracción de entidades explícitas: No funciona bien con conversaciones naturales.
   - Sin ponderación probabilística: Las aristas son binarias (existe/no existe).

Brecha identificada:

Ninguno de los enfoques existentes combina simultáneamente:

- Modelado probabilístico de relaciones contextuales
- Decaimiento temporal explícito y configurable
- Propagación de activación para descubrimiento de relaciones indirectas
- Auditoría y explicabilidad de decisiones de recuperación

El AGPC propuesto en este trabajo final aborda específicamente estas limitaciones mediante un Grafo Contextual Probabilístico (GCP) que integra similitud estructural, relevancia temporal y propagación de activación en un marco matemáticamente fundamentado.

2.3. Grafos Contextuales Probabilísticos (GCP)

Un **Grafo Contextual Probabilístico** es una estructura matemática diseñada para modelar y gestionar relaciones contextuales de manera difusa y no lineal.

La definición formal surge de integrar principios de **teoría de grafos** (Bondy & Murty, 2008), **redes semánticas** (Sowa, 2014), **modelos gráficos probabilísticos** (Koller & Friedman, 2009) y **representación vectorial de significado** (Mikolov et al., 2013; Perozzi et al., 2014).

La construcción de esta definición responde a una necesidad clave: representar fragmentos de información (nodos) y sus relaciones (aristas) no solo de manera estructural, sino también **ponderada probabilísticamente** para reflejar relevancia y dependencia contextual.

La aplicación de redes neuronales de grafos temporales para modelar contextos dinámicos (Chen et al., 2024; Wu et al., 2024) respalda la arquitectura propuesta en este trabajo, donde las relaciones probabilísticas evolucionan con el tiempo. Zhang y Liu (2024) presentan una revisión sistemática de grafos de contexto probabilístico en sistemas multiagente, identificando la gestión temporal como uno de los desafíos principales que el presente trabajo aborda mediante el modelo de decaimiento adaptativo.

Formalmente, la podemos definir como:

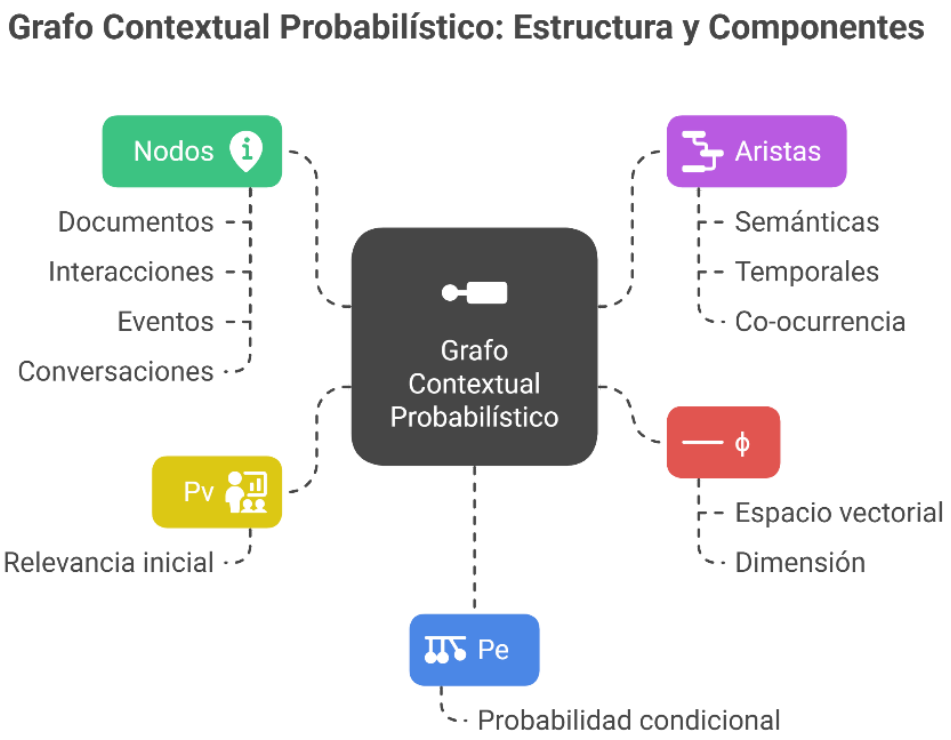
Grafo Contextual Probabilístico: $Gc = (V, E, Pv, Pe, \phi)$

donde:

- V: conjunto de nodos que representan fragmentos de contexto (documentos, interacciones, eventos, conversaciones).
- $E \subseteq V \times V$: conjunto de aristas que indican relaciones contextuales entre nodos (semánticas, temporales, de co-ocurrencia).
- $Pv : V \rightarrow [0,1]$: es una función que asigna a cada nodo una probabilidad inicial de relevancia en relación con la consulta o tarea actual.
- $Pe : E \rightarrow [0,1]$: es una función que asigna a cada arista la probabilidad condicional de que un nodo sea relevante dado que otro está activo.
- $\phi : V \rightarrow \mathbb{R}^d$: función de representación vectorial (embedding) que asigna a cada nodo un vector semántico.
    - $\mathbb{R}$: el conjunto de los números reales.
    - d: la dimensión del espacio vectorial.

En términos prácticos:

- Los **nodos** son “piezas de información”.
- Las **aristas** son “relaciones” entre esas piezas.
- **Pv** indica qué tan importante es cada pieza por sí sola.
- **Pe** indica qué tan probable es que una pieza sea útil si otra ya lo es.
- **$\phi(v) \in \mathbb{R}^d$** significa que cada pieza de información está representada como un vector numérico de **“d”** posiciones (un embedding), lo que permite medir cuán parecidas son entre sí.



2.3.1 Cómo se llegó a esta definición desde los fundamentos teóricos

1. **Teoría de grafos** (Bondy & Murty, 2008):

   La teoría de grafos proporciona la base matemática para representar información en forma de nodos y aristas. Cada nodo simboliza una unidad de información (un documento, evento o interacción) y cada arista refleja la existencia de una relación entre esas unidades. Este enfoque es fundamental porque permite organizar el conocimiento de manera estructurada y navegable, en lugar de tratarlo como un simple repositorio lineal. Sin la teoría de grafos, sería imposible modelar explícitamente las conexiones entre fragmentos de contexto, algo indispensable para que un agente pueda recuperar información más allá de coincidencias literales.

2. **Redes semánticas** (Sowa, 2014):

   Mientras la teoría de grafos ofrece la estructura, las redes semánticas amplían ese marco para capturar relaciones de **significado** entre los nodos. En este nivel, las aristas no solo indican que existe un vínculo, sino que también representan la naturaleza de ese vínculo: sinonimia, causalidad, jerarquía, proximidad conceptual, etc. Este aporte es crucial para que el **GCP** no sea un grafo puramente estructural, sino un modelo capaz de reflejar el **contenido semántico** y contextualizar las relaciones. De este modo, se logra que el grafo no solo conecte información, sino que la conecte con **sentido**.

3. **Modelos gráficos probabilísticos** (Koller & Friedman, 2009):

   La introducción de la probabilidad en los grafos permite manejar la **incertidumbre** inherente al lenguaje y a la información contextual. No todas las conexiones entre nodos son igual de relevantes, ni siempre tienen la misma fuerza. Gracias a los modelos gráficos probabilísticos, cada arista puede tener un **peso probabilístico** que indica la fortaleza de la relación, y cada nodo una probabilidad de relevancia en función de la tarea actual. Este componente fue decisivo para transformar un grafo estático en un **grafo dinámico y adaptable**, donde la información no es binaria (relacionado/no relacionado), sino que se gradúa según su importancia contextual.

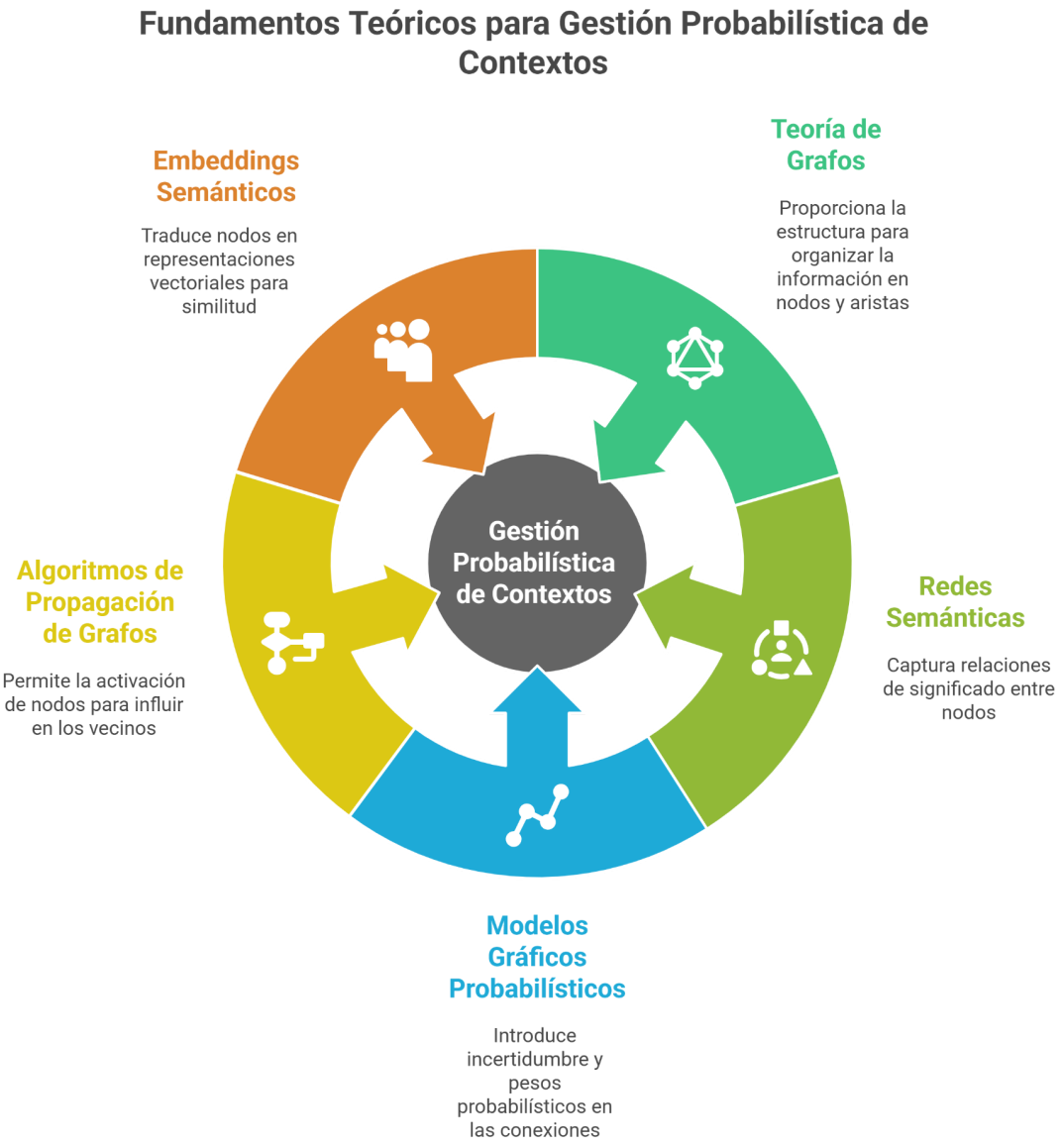
4. **Graph propagation algorithms** (Tong et al., 2006):

   Estos algoritmos aportan la mecánica necesaria para que la activación de un nodo pueda influir en sus vecinos, permitiendo descubrir relaciones **indirectas**. Por ejemplo, si un contrato se vincula a una factura y la factura a un litigio, la activación del litigio debería permitir recuperar también el contrato, incluso si no están conectados directamente. Esta capacidad de propagación es el corazón del **GCP**, ya que habilita la **recuperación ampliada**: no solo encontrar información explícita, sino también aquella que está implícita o conectada de manera no obvia.

5. **Embeddings semánticos** (Mikolov et al., 2013; Perozzi et al., 2014):

   Finalmente, los embeddings permiten traducir cada nodo en un vector en un espacio semántico multidimensional. Esto hace posible medir la similitud entre piezas de información, aunque usen palabras distintas (ej. “compraventa” y “contrato de venta”). Los embeddings enriquecen el grafo al dar a cada nodo una **representación numérica de su significado**, lo cual facilita la creación automática de aristas basadas en proximidad semántica, además de las relaciones explícitas y probabilísticas. Sin esta capa, el **GCP** quedaría limitado a coincidencias literales y perdería gran parte de su capacidad de generalización.

La definición formal de **Gc** es la síntesis de estos enfoques, adaptada al problema de la **gestión probabilística de contextos** en sistemas inteligentes.



Ventajas clave:

- Soporte para relaciones indirectas y no explícitas.
- Control granular del refuerzo y olvido contextual.
- Adaptabilidad a dominios heterogéneos mediante un núcleo arquitectónico común.

2.3.2 Ejemplo

Supongamos que un asistente legal ha procesado tres documentos:

1. **D1:** “Contrato de compraventa firmado en 2023 entre la empresa A y B.”
2. **D2:** “Factura emitida por la empresa B en mayo de 2024.”
3. **D3:** “Litigio judicial iniciado por incumplimiento de pago en junio de 2024.”

En el **GCP**:

- Cada documento es un nodo (V = {D1,D2,D3}).
- Hay aristas que conectan nodos si existe relación contextual:
  - E = {(D1,D2), (D2,D3), (D1,D3)}.
- Pv (D1) = 0,8, Pv (D2)=0,6, Pv (D3)=0,9 indican relevancia inicial según la consulta (ej. “conflictos legales por compraventa”).
- Pe(D1,D2) = 0,7 refleja alta relación temporal y semántica entre contrato y factura.
- Pe (D2,D3) = 0,85 indica fuerte relación entre factura e inicio de litigio.
- Pe(D1,D3) = 0,5, indica que hay una conexión semántica relevante pero menos directa que D1–D2 o D2–D3.
- $\phi(D1)$, $\phi(D2)$, $\phi(D3) \in R^d$ son embeddings generados por el modelo seleccionado; “d” corresponde a la dimensión propia de dicho codificador (p. ej., d=768 en modelos base tipo MPNet/BERT). Esta representación permite medir similitud semántica entre fragmentos.

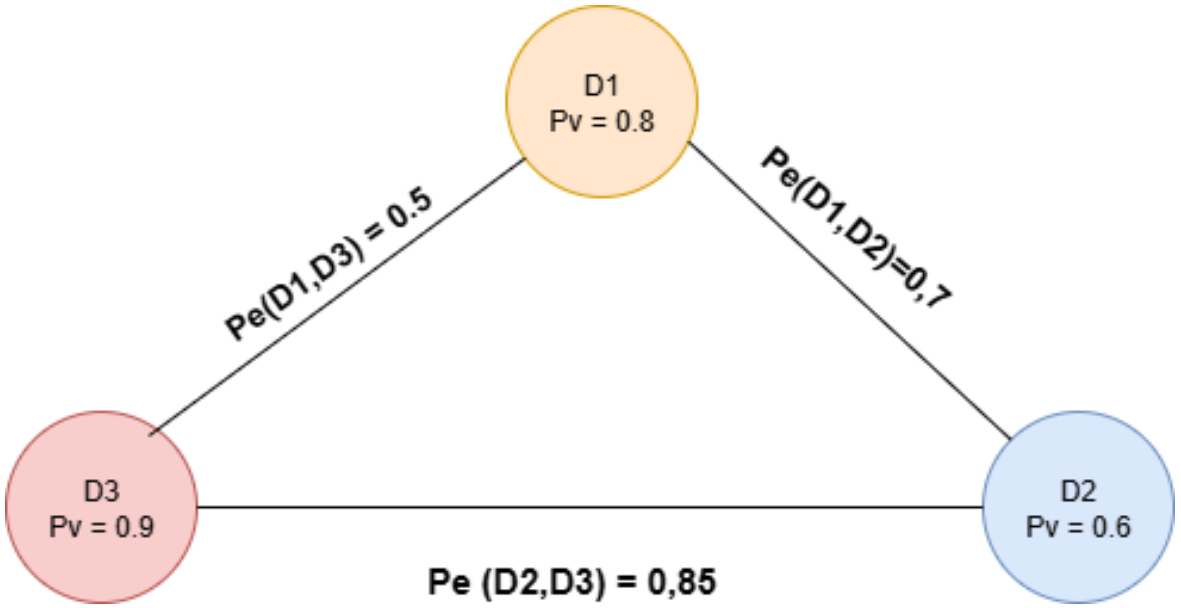
  Nota aclaratoria:

  - **BERT (Bidirectional Encoder Representations from Transformers)** es un modelo de lenguaje desarrollado por Google (Devlin et al., 2018) que introdujo el uso de *transformers bidireccionales* para obtener representaciones de texto. Su versión base genera embeddings de dimensión 768.
  - **MPNet (Masked and Permuted Pre-training)** es una evolución de BERT propuesta por Microsoft (Song et al., 2020), que combina enmascaramiento y permutación en el preentrenamiento. Está optimizado para producir representaciones semánticas más robustas, manteniendo también 768 dimensiones en su configuración base.

    Ambos modelos se utilizan comúnmente como **encoders de oraciones (sentence encoders)** para generar embeddings vectoriales de tamaño fijo,

adecuados para tareas de búsqueda semántica y recuperación de información.

Con este grafo, una búsqueda de “conflictos legales” podría **activar D3** y, gracias a la propagación, **recuperar D1 y D2,** aunque no mencionen literalmente “conflicto”.



2.4 Estado del Arte en Gestión de Contexto

Primeramente, se realizó una comparación conceptual y teórica con herramientas utilizadas en el mercado.

Los sistemas actuales, como **NotebookLM** de Google, presentan limitaciones al trabajar con contextos cerrados y sin actualización automática.
En contraste, el **AGPC**:

- Mantiene una **memoria viva y acumulativa**, actualizable en tiempo real.
- Ofrece una **estructura explícita y auditable** de nodos y aristas.
- Combina búsqueda vectorial con **propagación probabilística**, permitiendo descubrimiento de relaciones indirectas.

2.4.1 Análisis Diferencial (AGPC vs. NotebookLM)

2.4.1.1 Naturaleza del contexto

- **AGPC:** contexto dinámico, modelado como un **Grafo Contextual Probabilístico** donde nodos representan unidades de conocimiento y aristas indican relaciones ponderadas. Permite propagación probabilística para descubrir contextos indirectos relevantes.

- **NotebookLM:** contexto mayormente estático, basado en conjuntos de notas o documentos sin representación explícita de relaciones probabilísticas.

Ventaja AGPC: capacidad de inferir relaciones no triviales y ajustar la relevancia en tiempo real según el tipo de conexión y la consulta.

2.4.1.2 Estructura interna

- **AGPC:** arquitectura modular en capas (extracción semántica, memoria vectorial, GCP, motor de respuesta, UI(Interfaz de usuario), con separación de responsabilidades y puede llegar a soportar múltiples motores de embeddings y LLMs.
- **NotebookLM:** estructura cerrada y centralizada, sin poder acceder de forma detallada a cada paso del procesamiento.

Ventaja AGPC: control total sobre cada módulo, posibilidad de sustituir o mejorar componentes sin modificar el resto del sistema.

2.4.1.3 Recuperación de información

- **AGPC:** combina **recuperación semántica inicial** con **propagación probabilística** en el grafo para maximizar relevancia y diversidad.
- **NotebookLM:** recuperación limitada a coincidencias semánticas directas; sin expansión mediante redes de relaciones.

Ventaja AGPC: mayor cobertura de contexto relevante y reducción de sesgo hacia coincidencias literales.

2.4.1.4 Control y personalización

- **AGPC:** control de modelos de embeddings y Modelos de Lenguaje de Gran Escala (LLM) usados; posibilidad de ajustar los parámetros según el dominio.
- **NotebookLM:** parámetros internos no expuestos; personalización mínima posible.

Ventaja AGPC: ajuste para optimizar precisión, latencia y adecuación al dominio.

Tabla 3. Comparativa entre AGPC y NotebookLM

Criterio	**AGPC (Agente de Gestión Probabilística de Contextos)**	**NotebookLM (Google)**	**Ventaja AGPC**
Naturaleza del contexto	Contexto dinámico modelado como **Grafo Contextual Probabilístico**, con nodos que representan unidades de conocimiento y aristas que indican relaciones ponderadas. Permite propagación probabilística para descubrir contextos indirectos relevantes.	Contexto mayormente estático, basado en conjuntos de notas o documentos sin representación explícita de relaciones probabilísticas.	Inferencia de relaciones no triviales y ajuste de relevancia en tiempo real.
Estructura interna	Arquitectura modular en capas (extracción semántica, memoria vectorial, GCP, motor de respuesta, interfaz). Puede llegar a admitir múltiples motores de embeddings y LLMs.	Estructura cerrada y centralizada, sin acceso detallado a cada etapa del procesamiento.	Control total de módulos y posibilidad de sustitución/mejora de componentes sin modificar todo el sistema.
Recuperación de información	Combina recuperación semántica inicial con propagación probabilística para maximizar relevancia y diversidad.	Recuperación limitada a coincidencias semánticas directas, sin expansión mediante redes de relaciones.	Mayor cobertura de contexto valioso y reducción de sesgo hacia coincidencias literales.
Control y personalización	Puede llegar a permitir seleccionar modelos de embeddings y LLM, ajustando parámetros según dominio y requerimientos	Parámetros internos no expuestos y personalización mínima posible.	Ajuste optimizado a cada dominio, balanceando rendimiento y recursos.

	(precisión, latencia, privacidad).		
Actualización del contexto	Memoria viva y acumulativa, actualizable en tiempo real.	Contexto cerrado que requiere actualización manual o recarga de documentos.	Información siempre actualizada y adaptada a nuevas entradas.
Auditoría de relaciones	Representación explícita y auditable de nodos, aristas y pesos probabilísticos, lo que permite trazabilidad y análisis posterior.	No ofrece visualización ni auditoría detallada de relaciones internas.	Mayor transparencia y capacidad de verificación del proceso de razonamiento.
Soporte multimodal	Conversaciones + PDFs (texto, imágenes y tablas) + Presentaciones (diapositivas) + posibilidad de integrar imágenes, gráficos y otros formatos multimedia (potencial para audio / video como fuentes) .	Admite documentos de Google (Docs, Slides), PDFs, imágenes y URLs; genera resúmenes narrados (*Audio y Video Overviews*) basados en el contenido subido.	El AGPC no solo puede incorporar soporte multimodal, sino que **razona y vincula probabilísticamente la información entre formatos heterogéneos**. Esto amplía su aplicabilidad hacia escenarios de análisis cognitivo y gestión de conocimiento

2.4.2 AGPC vs. BM25 (Sistema Clásico de Recuperación de Información)

2.4.2.1 Descripción General de BM25

- **BM25 (Best Matching 25)** es un algoritmo de recuperación de información basado en el modelo de espacio vectorial.
- Su objetivo es calcular la **relevancia de un documento frente a una consulta** usando estadísticas de frecuencia de palabras.
- Fórmula base simplificada:

$$score(D,Q) = \sum_{t \epsilon Q} IDF(t).\frac{f(t,D).(k+1)}{f(t,D)+k.\,(1-b+b.\frac{|D|}{avgdl})}$$

Donde:

- f(t, D)= frecuencia del término en el documento.
- |D| = longitud del documento.
- avgdl = longitud promedio de documentos.
- k, b = parámetros de calibración (típicamente k≈1.2, b≈0.75).
- IDF(t) = Inverse Document Frequency (cuánto discrimina un término raro frente a uno común).

Tabla 4. Comparativa entre AGPC y BM25

Criterio	**AGPC (Agente de Gestión Probabilística de Contextos)**	**BM25**
Dimensión de la Relevancia	Mide relevancia considerando **similitud semántica (embeddings)** y **relevancia temporal** con ponderación probabilística.	Mide relevancia solo con base en **palabras compartidas** entre la consulta y el documento.
Consideración Temporal	Incorpora **decadencia temporal + factor de refuerzo** para ajustar la relevancia en función de la consulta (ejemplo: “mañana”, “próxima semana”).	Carece totalmente de un modelo temporal. Un documento de hace 10 años y uno de ayer son igualmente relevantes si tienen las mismas palabras.
Flexibilidad y Adaptabilidad	Dinámico, el peso temporal se ajusta con base en la intención de la consulta y puede evolucionar hacia factores adaptativos.	Estático, sus parámetros k,b son fijos; no aprende del usuario.
Interpretabilidad	También interpretable, pero más cercano a un modelo cognitivo (memoria semántica + temporal).	Altamente interpretable, cada parte de la fórmula tiene un sentido estadístico claro.
Ventajas	● Integra semántica y temporalidad. ● Modelo más cercano a la cognición humana. ● Ajustable con factores de refuerzo.	● Simplicidad y eficiencia (O(1) por término). ● Muy probado y robusto en IR(Recuperación de Información).

Limitaciones	● Mayor complejidad de cálculo que BM25. ● Requiere embeddings y anotaciones temporales.	● No capta semántica (ej: “meeting” ≠ “reunión”). ● No considera el tiempo ni la intención del usuario.

El modelo **BM25** se limita a realizar coincidencias léxicas, sin considerar aspectos relacionados con el significado ni con la dimensión temporal de las consultas. En contraste, el prototipo **AGPC** aborda específicamente la interpretación de consultas con intención temporal, superando así una limitación inherente a BM25.

2.4.2.2 Ejemplo de Diferencia en Resultados

Consulta: *“¿Qué reuniones tengo mañana?”*

- **BM25**:
  - “Reunión de proyecto (2019)” → alta puntuación (muchas coincidencias con “reunión”).
  - “Recordatorio mañana: reunión de equipo” → podría salir igual o más abajo si tiene menos repeticiones de la palabra.
- **AGPC**:
  - “Recordatorio mañana: reunión equipo” → prioridad máxima (similitud + alta relevancia temporal).
  - “Reunión de proyecto (2019)” → baja prioridad (irrelevante temporalmente).

2.4.3 AGPC vs. MemGPT (Sistema de Memoria Avanzada)

2.4.3.1 Descripción General de MemGPT

- **MemGPT** es un marco que extiende los LLM con una **memoria jerárquica y dinámica**.
- Su objetivo es **emular cómo funciona la memoria humana**, con diferentes capas:
  - **Memoria a corto plazo**: mantiene el contexto inmediato de la conversación.
  - **Memoria a mediano plazo**: resumen dinámico de interacciones previas.
  - **Memoria a largo plazo**: almacenamiento persistente que puede recuperarse bajo demanda.
- Implementa un **scheduler interno** que decide qué recuerdos mantener en “primer plano” y cuáles enviar al “fondo” (simulando atención selectiva).

Tabla 5. Comparativa entre AGPC y MemGPT

Criterio	**AGPC (Agente de Gestión Probabilística de Contextos)**	**MemGPT**

Dimensión Semántica	● Usa un cálculo explícito de **similitud estructural**. ● No depende de resúmenes subjetivos, sino de métricas matemáticas claras.	● Usa embeddings y resúmenes para mantener coherencia conversacional. ● El modelo decide qué memoria traer según la consulta.
Consideración Temporal	Temporalidad explícita: ● **Decaimiento temporal** (memoria se degrada con el tiempo). ● **Factor de refuerzo** (consultas recientes/urgentes tienen mayor peso).	● Tiene noción de temporalidad en su jerarquía, pero **implícita** (depende de cómo resuma y recupere el modelo). ● No cuenta con un decaimiento o refuerzo matemático directo.
Estructura de la Memoria	● Memoria en **grafo de contextos**: cada nodo representa información, con aristas que reflejan relaciones ● Semántico-temporales: Los pesos de los nodos evolucionan con la interacción.	● Memoria jerárquica (short/medium/long-term). ● Mecanismo de “paginación” (mueve recuerdos dentro/fuera del contexto).
Interpretabilidad	● Cada decisión de priorización es **explicable matemáticamente** (similitud + temporalidad + refuerzo).	● El modelo decide qué memorias priorizar, lo que puede ser opaco. ● Los resúmenes pueden omitir información importante.

Ventajas	● Temporalidad explícita y controlada. ● Memoria estructurada y cuantificable. ● Transparencia en la priorización.	● Memoria jerárquica similar a la humana. ● Capacidad de mantener coherencia en conversaciones largas. ● Ideal para asistentes personales conversacionales.
Limitaciones	● No produce narrativas complejas como un LLM. ● Depende de parámetros manuales (decaimiento, refuerzo).	● Temporalidad implícita (no matemática). ● Menos interpretable. ● Riesgo de pérdida de información en resúmenes.

2.4.3.2 Ejemplo de Diferencia en Resultados

Consulta: *"¿Qué pasó en la reunión de ayer y qué tengo mañana?"*

- **MemGPT**:
  - Puede traer un **resumen narrativo** de la reunión de ayer.
  - Para mañana, depende de si la información fue guardada explícitamente en memoria.
- **AGPC**:
  - Pondera automáticamente el contexto de "ayer" (decaimiento suave) y "mañana" (refuerzo fuerte).
  - Ranking matemático asegura que lo de mañana tenga prioridad en la respuesta.

2.4.4 AGPC vs. RAG con LangChain (Sistema Moderno de Contexto)

2.4.4.1 Descripción General de RAG con LangChain

- **RAG (Retrieval Augmented Generation)** es un enfoque híbrido que combina:
  - **Recuperación de información** (retrieval): búsqueda de documentos relevantes en una base vectorial.
  - **Generación** (generation): un modelo de lenguaje grande (LLM) utiliza esos documentos para generar respuestas.
- **LangChain** es un framework que facilita la construcción de este pipeline, integrando:
  - **Vector stores** (ej. FAISS, Pinecone, Weaviate).
  - **Embeddings** para similitud semántica.
  - **Chain-of-thoughts** estructurados para consultas complejas.

- **Memory components** para mantener conversación.

Tabla 6. Comparativa entre AGPC y RAG con LangChain

Criterio	**AGPC (Agente de Gestión Probabilística de Contextos)**	**RAG con LangChain**
Dimensión Semántica	● También usa embeddings para medir similitud estructural. ● No se centra en generación de texto, sino en **priorización probabilística de contextos** (grafo de relevancia).	● Usa embeddings para buscar documentos semánticamente relacionados. ● Se apoya en LLM para generar respuestas con lenguaje natural.
Consideración Temporal	● Incorpora un **modelo de decadencia temporal** (exponencial, lineal o Híbrido Adaptativo). ● Tiene un **factor de refuerzo adaptativo** según la intención de la consulta.	● Temporalidad depende de los metadatos (ejemplo: fecha de un documento). ● No existe un **mecanismo nativo de decaimiento temporal ni de refuerzo**. ● Se puede implementar un filtro por fecha, pero es binario (incluye/excluye).
Estructura de la Memoria	● Usa un **grafo de contextos** con pesos dinámicos (semánticos + temporales). ● La memoria se organiza en nodos y aristas, con relevancia dinámica.	● Funciona principalmente con un **vector store plano**. ● La memoria conversacional es limitada (buffer, summary, o window).

Interpretabilidad	● Altamente interpretable: cada decisión se explica con **similitud + relevancia temporal + refuerzo**.	● El razonamiento depende del LLM: más difícil interpretar por qué eligió ciertos contextos. ● Se puede inspeccionar el ranking de embeddings, pero no siempre es transparente.
Ventajas	● Gestión explícita de temporalidad (decaimiento + refuerzo). ● Modelo matemático interpretable y robusto. ● Orientado a **memoria estructurada de largo plazo**.	● Potencia de LLM para generar respuestas ricas en lenguaje natural. ● Ecosistema amplio (integraciones con APIs, bases vectoriales). ● Muy flexible para prototipado rápido.
Limitaciones	● No genera respuestas naturales por sí mismo. ● Requiere diseño cuidadoso de los parámetros de refuerzo.	● No maneja temporalidad de forma nativa. ● Menos interpretable. ● Depende de LLM (costo computacional y riesgo de alucinación).

Mientras que RAG se centra en proporcionar respuestas directas, AGPC se distingue por priorizar contextos y estructurar la memoria de manera persistente. A diferencia de RAG, que requiere reglas adicionales para manejar la temporalidad y se basa principalmente en almacenamiento vectorial, AGPC incorpora una gestión matemática explícita del tiempo, ofreciendo mayor transparencia en sus procesos frente a la opacidad generativa característica de RAG.

2.4.4.2 Ejemplo de Diferencia en Resultados

Consulta: *"¿Qué tengo pendiente para mañana?"*

- **RAG con LangChain**:
  - Recupera documentos con "mañana" en el texto o embeddings cercanos.
  - Si no encuentra explícitamente "mañana", puede fallar.
  - Respuesta final depende del LLM (riesgo de "alucinación").
- **AGPC**:

- Ajusta la relevancia con la **intención temporal fuerte**.
- Contextos con fechas cercanas reciben **prioridad** automática.
- Menor riesgo de irrelevancia porque no depende de la generación.

2.4.5 Metodología de Comparación con Sistemas Existentes

Para evaluar objetivamente el desempeño del AGPC, se han seleccionado sistemas representativos del estado del arte en gestión contextual, estableciendo criterios claros de comparación experimental.

Sistema seleccionado para comparación práctica:

RAG (Retrieval-Augmented Generation) con una implementación estándar

- **Configuración:** ChromaDB como vector store, embeddings sentence-transformers/all-mpnet-base-v2
- **Recuperación:** Top-K semántica pura sin consideración temporal explícita
- **Razón de selección:** Representa el enfoque más adoptado actualmente en producción para gestión contextual con LLMs

Justificación de la selección de RAG como sistema de comparación:

1. **Amplia adopción:** RAG es el estándar de facto en la industria para aplicaciones que requieren combinar recuperación de información con generación de lenguaje natural.
2. **Implementación bien documentada:** Existen múltiples frameworks maduros (LangChain, LlamaIndex) que permiten replicabilidad experimental.
3. **Arquitectura comparable:** Al igual que el AGPC, RAG opera con embeddings semánticos y bases vectoriales, lo que permite comparación directa de estrategias de recuperación contextual.
4. **Ausencia de gestión temporal nativa:** RAG no implementa mecanismos de decaimiento temporal ni priorización probabilística basada en tiempo, lo que permite validar específicamente la contribución del modelo temporal propuesto en el AGPC.

Protocolo de comparación experimental:

Para garantizar reproducibilidad, la comparación se realizará bajo las siguientes condiciones controladas:

1. **Dataset común:** Las 200 conversaciones del dominio legal se cargarán en ambos sistemas con el mismo preprocesamiento de fragmentación.
2. **Conjunto de consultas estandarizado:** Se definirán consultas de prueba que incluyan:
    - Consultas temporales explícitas (ejemplo: "casos de la semana pasada", "reuniones de mañana")

- Consultas estructurales puras (ejemplo: "precedentes sobre contratos laborales")
- Consultas mixtas (combinan temporalidad y semántica)

3. **Configuración equivalente:**
   - Mismo modelo de embeddings: sentence-transformers/all-mpnet-base-v2
   - Mismo almacenamiento vectorial: ChromaDB
   - Mismo número de contextos recuperados: Top-5
   - Mismo LLM para generación final.

Tabla 7. Diferencias arquitectónicas clave entre AGPC y RAG estándar:

Aspectos	**AGPC**	**RAG**
Característica	Recuperación vectorial + propagación en grafo + decaimiento temporal	Recuperación vectorial pura basada en similitud coseno
Manejo de temporalidad	Modelo de decaimiento exponencial + factor de refuerzo adaptativo	Filtros binarios por fecha (si existe metadato temporal)
Relaciones indirectas	Propagación de activación descubre vínculos indirectos	No detecta conexiones transitivas
Explicabilidad	Grafo auditable con pesos estructurales y temporales	Ranking de similitud (valores numéricos)

Conclusión metodológica:

La comparación entre AGPC y RAG estándar permite aislar y medir específicamente el impacto de tres innovaciones clave del presente trabajo:

1. Gestión temporal explícita mediante decaimiento y refuerzo
2. Descubrimiento de relaciones indirectas mediante propagación en grafos
3. Adaptabilidad según intención de consulta (temporal vs. estructural)

Los resultados de esta comparación se presentan en la sección: 5. Resultados

2.5 Contextos Temporales

La dimensión temporal constituye un eje fundamental en la **gestión probabilística de contextos**. Mientras que la mayoría de los enfoques tradicionales en agentes inteligentes priorizan relaciones semánticas o estructurales, el **Agente de Gestión Probabilística de Contextos (AGPC)** incorporará un modelado explícito de la **proximidad temporal**, lo que permite ajustar dinámicamente la relevancia de la información en función de su vigencia.

2.5.1. Fundamentos conceptuales

Es posible distinguir dos dimensiones independientes en la gestión contextual:

1. **Relación Estructural (Indeleble):**
    - Intrínseca al contenido y la semántica.
    - Permanece estable en el tiempo.
    - Ejemplo: una *auditoría* siempre mantiene relación con el *presupuesto* evaluado, independientemente de su fecha.
2. **Relevancia Temporal (Circunstancial):**
    - Depende del momento actual.
    - Su importancia fluctúa y decae con el paso del tiempo.
    - Ejemplo: un *deadline* es altamente valioso la semana previa a su vencimiento, pero su relevancia decrece una vez concluido.

Este enfoque dual permite al sistema preservar la memoria estructural de largo plazo, al mismo tiempo que prioriza los eventos urgentes o inmediatos, replicando un mecanismo similar al razonamiento humano (Sowa, 2014; Koller & Friedman, 2009).

2.5.2. Proximidad temporal y relevancia

La **proximidad temporal** es un determinante clave de la relevancia en el AGPC. Para ello se podrían utilizar funciones de decaimiento que modulan las probabilidades de activación de los nodos del **Grafo Contextual Probabilístico (GCP)**.

La relevancia temporal en tareas intensivas en conocimiento ha sido identificada como un área crítica de investigación (Rajpurkar et al., 2023; Li et al., 2024), confirmando la importancia del mecanismo de decaimiento temporal implementado en el AGPC. Estos autores demuestran que los sistemas de diálogo que ignoran la dimensión temporal presentan tasas de error hasta 40% superiores en tareas de recuperación de información distribuida en el tiempo.

Ejemplos ilustrativos:

- **Alta relevancia temporal:** eventos con pocos días de diferencia (ej. *“Reunión proyecto X - 15/01/2025”* y *“Deadline proyecto X - 20/01/2025”*).
- **Relevancia media:** intervalos de uno a tres meses (ej. *“Presupuesto proyecto X - 15/03/2025”* respecto al deadline de enero).
- **Baja relevancia:** eventos separados por trimestres o años (ej. *“Kick-off proyecto X - 15/06/2024”*).

De esta forma, el sistema asigna mayor peso probabilístico a lo cercano en el tiempo, sin perder trazabilidad hacia los contextos históricos.

2.5.3. Métodos de métricas de similitud (W estructural)

En la construcción del Grafo Contextual Probabilístico (GCP), un aspecto clave es cómo calcular el **peso estructural** (W estructural) que conecta dos nodos. Para ello se suelen combinar dos fuentes de información complementarias:

1. **Similitud léxica o por keywords**, que mide coincidencias literales en el texto.
2. **Similitud semántica**, que mide la cercanía de significados en el espacio de embeddings.

A continuación, se presentan dos enfoques posibles para dicha combinación:

2.5.3.1 Promedio simple (propuesta base)

Fórmula:

$$W\ estructural\ = \frac{SIMILITUD\ keywords + SIMILITUD\ semantica}{2}$$

Características:

- Cada métrica tiene el mismo peso (50%).
- No requiere ajuste de parámetros, lo cual simplifica la implementación.
- Asume que ambas métricas son igualmente confiables.

Ventajas:

- Muy simple de explicar e implementar.
- Evita sesgos, ya que no privilegia ninguna métrica a priori.

Desventajas:

- Sensible al ruido: si una métrica es deficiente (ej. keywords (coincidencia de palabras clave) pobres), afecta al resultado.
- No ofrece flexibilidad para privilegiar una fuente sobre la otra.

2.5.3.2 Promedio ponderado con parámetro α (propuesta extendida)

Fórmula:

$$W\ estructural\ =\ \alpha\ .\ SIMILITUD\ keywords\ +\ (1\ -\ \alpha)\ .\ SIMILITUD\ semantica,$$
$$Donde\ 0\ \leq\ \alpha\ \leq\ 1$$

Características:

- Es un promedio ponderado.
- El parámetro α regula la importancia relativa de cada métrica.
- Ejemplo: α=0.7 favorece keywords; α=0.3 favorece embeddings.

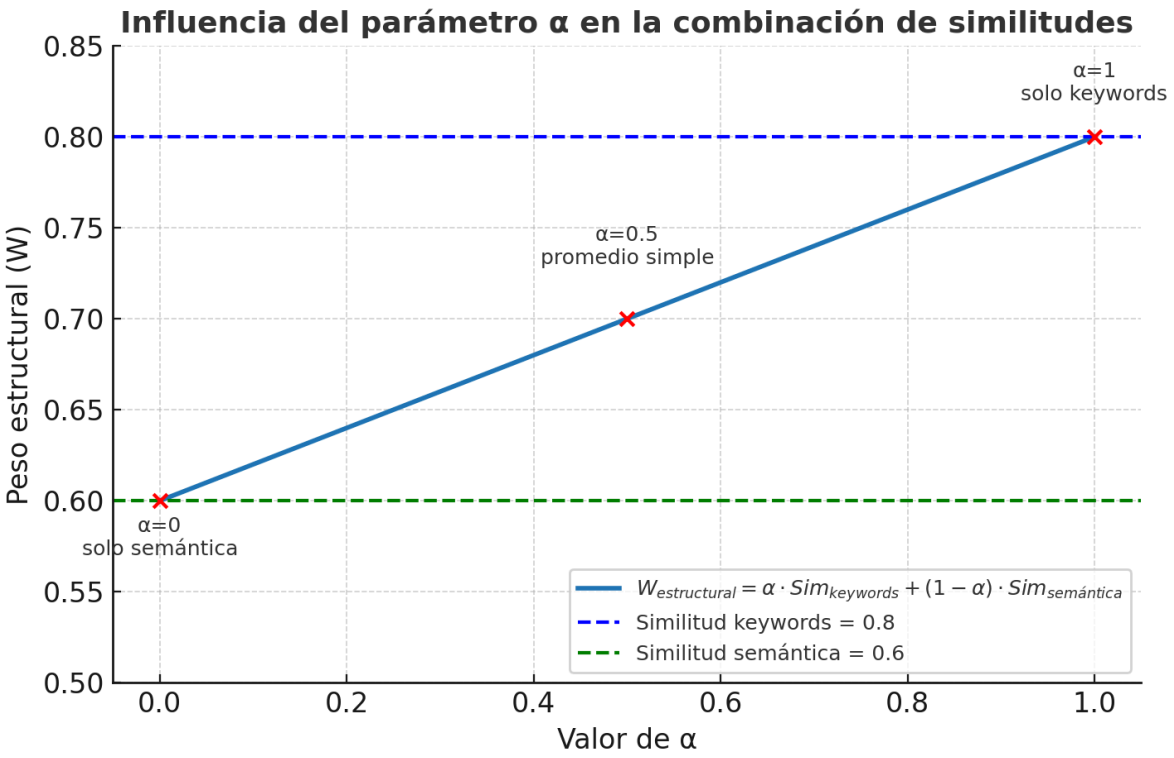
Ventajas:

- Ofrece flexibilidad: el peso relativo se ajusta al dominio o caso de uso.
- Permite robustez: se puede disminuir el impacto de una métrica menos confiable.
- Se puede ajustar dinámicamente (ej.: más peso a keywords en textos cortos, más a semántica en textos largos).

Desventajas:

- Requiere calibrar α, lo cual implica experimentación.
- Menos intuitivo para audiencias no técnicas.

2.5.3.2.1 Análisis gráfico del efecto de α en la combinación de similitudes



Lectura e interpretación del gráfico:

El gráfico muestra cómo varía el **peso estructural** W estructural al cambiar el parámetro α en la combinación:

$$W\ estructural\ (\alpha) = \alpha\ .\ SIMILITUD\ keywords + (1-\alpha)\ .\ SIMILITUD\ semantica$$

En el ejemplo del gráfico, fijamos $SIMILITUD\ keywords$=0,8 y $SIMILITUD\ semantica$=0,6.

- **Eje X (horizontal):** valores de α entre 0 y 1.

- **Eje Y (vertical):** valor resultante de W estructural.
- **Líneas punteadas horizontales:** los valores base de cada métrica (0,8 y 0,6).
- **Línea azul continua:** el valor de W estructural para cada α.
- **Marcas rojas(x):** tres puntos de referencia:
  - α=0 → **solo semántica**: W=0,6.
  - α=0,5 → **promedio simple**: $W = \frac{0,8 + 0,6}{2} = 0,7$
  - α=1 → **solo keywords**: W=0,8.

2.5.3.2.2 Qué significa α y cómo influye

α es un **coeficiente de confianza/ponderación** que regula la importancia relativa de cada fuente:

- α alto → el sistema se vuelve más **literal** (prioriza coincidencias de palabras).
- α bajo → el sistema se vuelve más **semántico** (prioriza significado en embeddings).

Como la fórmula es una **combinación convexa**, W estructural **siempre queda entre** los valores de ambas métricas:

$$min\,(SIMILITUD\ keywords,\ SIMILITUD\ semantica) \leq W(\alpha)$$

$$\leq max(SIMILITUD\ keywords,\ SIMILITUD\ semantica)$$

Esto garantiza que el resultado **no "se pase"** de 0.8 ni **caiga por debajo** de 0.6 en el ejemplo.

2.5.3.2.3 Propiedades matemáticas clave

Fórmula original

$$W\ estructural = \alpha\,.\,SIMILITUD\ keywords + (1-\alpha)\,.\,SIMILITUD\ semantica,\ Donde\ 0 \leq \alpha \leq 1$$

Es la definición de la combinación ponderada entre similitud por keywords y similitud semántica.
Si abrimos el paréntesis(multiplicamos (1-α) por $SIMILITUD\ semantica$):

$$W(\alpha) = \alpha\,.\,SIMILITUD\ keywords + SIMILITUD\ semantica - \alpha\,.\,SIMILITUD\ semantica$$

Juntamos los términos que tienen α:

$$W(\alpha) = SIMILITUD\ semantica + \alpha(SIMILITUD\ keywords - SIMILITUD\ semantica)$$

Esta expresión dice lo siguiente:

- El valor de partida (cuando α=0) es $SIMILITUD\ semantica$.

- A medida que aumentamos α, le vamos sumando una fracción del **diferencial** entre las dos medidas: ($SIMILITUD\ keywords$ - $SIMILITUD\ semantica$).

En otras palabras:

W(α)=Punto de inicio (semántica) + α x “diferencia entre keywords y semántica”

Esto significa que **la línea de W(α) es recta** y va desde el valor semántico (cuando α=0) hasta el valor keywords (cuando α=1).

- Es una **recta** entre los puntos (0, $SIMILITUD\ semantica$) y (1, $SIMILITUD\ keywords$).
- La **pendiente** es:
  $\frac{dW}{d\alpha} = SIMILITUD\ keywords - SIMILITUD\ semantica$

En el ejemplo, 0,8−0,6=0,2. Por tanto, **cada +0,1 en α** aumenta W en **+0,02**.

- **Monotonía:** si $SIMILITUD\ keywords > SIMILITUD\ semantica$ (como aquí), la recta **crece** con α; si fuera al revés, **decrece**; si son iguales, W es **constante** y α no cambia nada.

2.5.3.2.4 Ejemplos numéricos rápidos

- α=0,3: W=0,6+0,3(0,2)=0,66 → leve sesgo hacia semántica.
- α=0,8: W=0,6+0,8(0,2)=0,76 → fuerte sesgo hacia keywords.

2.5.3.2.5 Implicancias prácticas

- **Textos cortos/terminología fija (ej. legal):** usar α **alto** para privilegiar exactitud léxica.
- **Textos largos/variación lingüística:** usar α **bajo** para capturar sinonimias y parafraseo.
- **Ajuste de producción:** tratar α como **hiperparámetro** y calibrarlo con métricas (precisión) en un conjunto de validación.

2.5.3.3 Comparación directa

Tabla 8. Comparación de promedio simple con ponderación con α

Criterio	**Promedio simple**	**Ponderación con α**
Simplicidad	Muy simple	Depende de α
Flexibilidad	Fija	Configurable
Robustez	Sensible al ruido	Más estable con ajuste

Explicabilidad	“Promedio de dos”	“Mezcla con parámetro”
Adecuado para prototipos	Sí	Más complejo
Adecuado para producción	Limitado	Adaptable

- Para un **prototipo inicial**, el promedio simple resulta ideal: es claro, rápido y sin necesidad de calibración.
- Para un **sistema en producción**, la fórmula ponderada con α(Alpha) ofrece ventajas en flexibilidad y robustez, permitiendo adaptar el modelo según métricas de calidad y comportamiento real de los usuarios.

En consecuencia, se recomienda comenzar con el enfoque de promedio simple y, una vez obtenidos datos empíricos, experimentar con distintos valores de α para optimizar el desempeño del sistema.

2.5.4. Ventajas del enfoque dual

- **Preservación del conocimiento:** las conexiones estructurales nunca se olvidan, garantizando memoria a largo plazo.
- **Adaptación dinámica:** la relevancia temporal fluctúa naturalmente, privilegiando lo vigente sin descartar lo antiguo.
- **Flexibilidad en consultas:** el peso temporal puede ser ajustado según la intencionalidad de la búsqueda.
- **Razonamiento más humano:** el sistema prioriza lo urgente sin perder de vista el trasfondo histórico, imitando la manera en que los humanos jerarquizamos información.

2.5.5. Ejemplo aplicado

En un entorno de gestión de proyectos, el AGPC prioriza la información de acuerdo con la fecha de consulta:

- El 24 de enero de 2025, la consulta *“¿qué tengo mañana?”*, activará con máxima relevancia el nodo *“Presentación de resultados 25/01/2025”*.
- El sistema reforzará también *“Deadline 25 de enero”* por proximidad inmediata.
- Mantendrá accesible *“Reunión 20 de enero”* con menor peso.
- Y conservará vínculos estructurales con el *“Presupuesto de marzo”* o el *“Kick-off de junio 2024”*, aunque con relevancia reducida.

De esta manera, el **AGPC no solo recuerda, sino que entiende cuándo la información es oportuna y relevante**, logrando un salto cualitativo en la inteligencia contextual.

2.6 Algoritmo de Decaimiento Temporal

La dimensión temporal constituye un componente importante en la gestión probabilística de contextos. El Agente de Gestión Probabilística de Contextos (AGPC) debe asignar mayor relevancia a los eventos recientes y reducir progresivamente el peso de los fragmentos de información más antiguos. Para ello, se implementan funciones de **decaimiento temporal**, que modulan la probabilidad de activación de los nodos en el Grafo Contextual Probabilístico (GCP).

2.6.1 Decaimiento Exponencial (implementado en el prototipo)

Fórmula:

$$Relevancia = e^{-\frac{diferencia\ dias}{factor\ de\ decaimiento}}$$

Características principales:

- **Simplicidad matemática**: requiere un solo parámetro ajustable (factor de decaimiento).
- **Disminución rápida inicial:** la relevancia de la información cae de manera pronunciada en los primeros días, priorizando los eventos recientes.
- **Cola larga de relevancia:** la curva no llega nunca a cero, lo que implica que la información mantiene un valor residual mínimo y no se descarta por completo.
- **Eficiencia computacional:** el cálculo de relevancia se ejecuta en tiempo constante (O(1)), dado que solo requiere operaciones básicas (diferencia de días, división y exponencial), lo que garantiza un bajo costo incluso con grandes volúmenes de datos.
- **Analogía natural:** este comportamiento reproduce patrones reales:
    - En psicología, la “curva del olvido” muestra que los recuerdos se pierden rápido al inicio y luego se estabilizan.
    - En física, el decaimiento radiactivo(proceso natural por el cual los átomos inestables se transforman en otros más estables, **liberando energía o partículas en el camino**.) sigue una forma exponencial, disminuyendo gradualmente sin anularse completamente.

Ejemplo de factores aplicados en el prototipo:

factores = {

```
    "Reunión": 2,        # Decae muy rápido
    "tarea": 7,          # Decae rápido
    "evento": 3,         # Decae rápido
    "proyecto": 45,      # Decae lento
    "conocimiento": 365, # Decae muy lento
    "general": 30        # Decae moderado
}
```

Este enfoque es el implementado en el prototipo del AGPC por su simplicidad, robustez y coherencia con la hipótesis de trabajo.

2.6.2 Decaimiento Lineal

Fórmula:

$$Relevancia = max\left(0, 1 - \frac{diferencia\ dias}{ventana\ maxima}\right)$$

- Valor inicial (día 0):
  Cuando la diferencia de días es cero, la relevancia es máxima (1).
  Significa que la información recién creada es completamente relevante.
- Disminución proporcional:
  La relevancia se reduce de manera lineal y constante con el paso de los días.
  Es decir, que cada día que pasa, el valor baja en la misma proporción:

$$\frac{1}{ventana\ maxima}$$

- Parámetro ventana máxima:
  Define el límite de tiempo en el que la información se considera útil.
  Por Ejemplo: si ventana máxima es de 30, entonces al día 15 la relevancia será 0.5 (la mitad).
- Corte abrupto (función Max):
  Una vez superada la ventana máxima, la relevancia se fija en 0.
  Es decir, la información deja de ser considerada en el sistema.

Ventajas:

- Modelo simple y fácil de interpretar.

- Comportamiento predecible y cálculo inmediato.

Limitaciones:

- Corte abrupto en la **ventana máxima**, que elimina de manera repentina cualquier relevancia.
- No refleja de forma realista la persistencia de la memoria humana.

Aplicación recomendada: sistemas con límites temporales estrictos (ejemplo: noticias, plazos de vencimiento).

En otras palabras, el decaimiento lineal significa que la información **pierde relevancia de forma uniforme en el tiempo** hasta llegar a un punto en el que deja de ser útil, momento en el que se descarta completamente.

2.6.3 Decaimiento Híbrido Adaptativo

Con el objetivo de evolucionar hacia un sistema completo, se proyecta el uso de un **modelo adaptativo** que combine distintas curvas de decaimiento según el tipo de contexto y el perfil del usuario.

Fórmula general:

$$Relevancia = w1 \cdot e^{-\frac{t}{\tau 1}} + w2 \cdot e^{-\frac{t}{\tau 2}}$$

¿Qué significa cada símbolo?

- **t**: tiempo transcurrido desde la creación o último evento del nodo (por ejemplo, en días). Es la variable independiente que mide que tan antiguo es el dato respecto al momento de la consulta.
- $e^{-\frac{t}{\tau}}$: la función exponencial que modela el decaimiento. Para t=0 vale 1, y disminuye al aumentar t(tiempo).
- τ**1,**τ**2**: **constantes de decaimiento**, expresadas en las mismas unidades que t (por ejemplo, días). Controlan la rapidez del olvido:
  - τ**pequeño** → decaimiento **rápido** (la exponencial cae rápido).
  - τ**grande** → decaimiento **lento** (la exponencial se mantiene mayor tiempo).
- Los pesos **w1 y w2** determinan la contribución de cada curva exponencial a la memoria total. Al aumentar uno de los pesos, se incrementa la importancia de ese tipo de memoria, ya sea **rápida(w1)** o **lenta(w2)**. Estos pesos se ajustan de manera conjunta para lograr un equilibrio entre retener información reciente y no perder del todo recuerdos antiguos.
  - En el momento inicial, (t=0), cuando la información es nueva, la relevancia total es la suma de ambos pesos, es decir, **Relevancia(0) = w1 + w2**. Esto significa que los pesos determinan el valor de partida de la relevancia. Si

w1+w2=1, la relevancia inicial comienza en el 100%. Si esta suma es mayor, la relevancia inicial será superior a 1, lo que podría requerir una normalización para interpretarla adecuadamente.

- Si **w1 es grande y w2 es pequeño**, el sistema privilegia la memoria de **corto plazo**. La relevancia disminuirá rápidamente porque la parte exponencial de $\mathcal{T}1$ (la constante de tiempo rápida) domina la función.
- Por el contrario, si **w1 es pequeño y w2 es grande**, se prioriza la memoria de **largo plazo**. En este caso, la relevancia decrecerá de forma más lenta, lo que permite que la información se conserve por un período más prolongado.

Interpretación funcional

- La fórmula suma dos componentes:
  1. **Componente de corto plazo**: $w1 \cdot e^{-\frac{t}{\tau 1}}$. Modela memoria/pertinencia que **cae rápido** (por ejemplo, reuniones, mensajes recientes).
  2. **Componente de largo plazo**: $w2 \cdot e^{-\frac{t}{\tau 2}}$. Modela memoria más **persistente** (por ejemplo, conocimiento base, proyectos de larga duración).
- El resultado es una **combinación** (superposición) que permite simultáneamente priorizar lo reciente y conservar trazabilidad histórica.

Ventajas:

- Permite modelar distintos tipos de memoria (corto y largo plazo).
- Ajustable dinámicamente según el dominio o el usuario.
- Escalable hacia un sistema de aprendizaje automático.

Desventajas:

- Requiere calibración más compleja y múltiples parámetros.
- Riesgo de sobreajuste en contextos con datos limitados.

Aplicación recomendada: versiones maduras del AGPC en producción, donde se requiera personalización y mayor flexibilidad.

Como extensión de los mecanismos de decaimiento descritos, la nueva versión del prototipo integra un sistema de *normalización temporal* que unifica los formatos de tiempo utilizados en todos los módulos del agente. Esta normalización evita desplazamientos por zonas horarias y garantiza que las fechas registradas en la base de datos reflejen la hora local del usuario, lo cual resulta esencial para el correcto funcionamiento de los algoritmos de decaimiento y refuerzo.

De esta forma, el modelo temporal no solo se mantiene matemáticamente consistente, sino también semánticamente coherente, al asegurar que los eventos se ordenen según su

ocurrencia real, sin ambigüedades asociadas a conversiones horarias o microsegundos residuales.

3. Desarrollo del Agente de Gestión Probabilística de Contextos (AGPC)

Esta sección detalla la implementación práctica del **Agente de Gestión Probabilística de Contextos (AGPC)**, un sistema inteligente diseñado para la administración avanzada y no lineal de información. El AGPC se asienta sobre el **Grafo Contextual Probabilístico (GCP)** como su núcleo arquitectónico central, integrando capacidades de Procesamiento de Lenguaje Natural (PLN), Modelos de Lenguaje de Gran Escala (LLMs), bases vectoriales y bases de datos de grafos.

El desarrollo se enfocó en superar las limitaciones de los enfoques lineales de memoria estática, permitiendo que el sistema establezca y pondere de forma probabilística las complejas relaciones entre fragmentos de información (o contextos atómicos) distribuidos en el tiempo y el espacio. Para lograr un prototipo funcional y escalable, se implementaron soluciones clave que se describen a continuación: la **fragmentación de conversaciones** en nodos atómicos para mejorar la precisión de la recuperación, la **normalización sigmoidea de pesos** para garantizar la consistencia matemática y la interpretabilidad probabilística de las relaciones, y optimizaciones de **actualización incremental y rendimiento** que transformaron la complejidad cuadrática en un crecimiento lineal.

Los siguientes subapartados abordan en detalle cada uno de estos componentes fundamentales, que son la base para la validación empírica del modelo propuesto en el dominio legal.

3.1 Fragmentación de conversaciones y representación enriquecida de nodos

3.1.1 Motivación y problema

En la práctica de recolección de contexto, las conversaciones largas se almacenan a menudo como bloques monolíticos (un único “documento”), lo que dificulta la localización de ideas concretas, decisiones y acciones dentro de ellas. Este problema limita la capacidad de recuperación semántica, la trazabilidad de decisiones y la construcción de relaciones precisas en el Grafo Contextual Probabilístico (GCP). La fragmentación inteligente propone transformar cada conversación en **un conjunto de fragmentos atómicos** que luego se mapean a nodos del GCP, mejorando así precisión de búsqueda, auditabilidad y escalabilidad.

3.1.2 Objetivos de la fragmentación

1. Generar **unidades de información (fragmentos)** suficientemente atómicas para recuperar ideas precisas sin perder coherencia. También podemos dirigirnos a estas unidades particulares como contexto
2. Mantener **referencia a la conversación origen** para preservar trazabilidad y contexto.

3. Añadir **metadatos estructurados** que permitan búsquedas por tipo (decisión, pregunta, tarea), participantes, posición temporal, etc.
4. Permitir **procesamiento masivo** y eficientes actualizaciones incrementales en el GCP.

3.1.3 Criterios de fragmentación

Sugerencia paramétrica:

- **Detección de cambio de hablante**: dividir en límite cuando cambia el emisor (por ejemplo: Juan, María, otro hablante).
- **Detección de separadores explícitos**: líneas con **"---"**, encabezado tipo **"DECISIONES TOMADAS:"**, **listas numeradas** que inician bloque.
- **Control de tamaño**:
  - Si fragmento > Máximo palabras (ej. 300 palabras) → dividir en oraciones o párrafos manteniendo coherencia temática.
  - Si fragmento < Mínimo palabras (ej. 50 palabras) → unir con siguiente fragmento lógico.
- **Cortes preferentes**: priorizar puntos finales ('.', '?', '!') y marcadores semánticos (por ejemplo: "Conclusión:", "Acción:").
- **Clasificación temática inmediata**: cada fragmento se etiqueta automáticamente (decisión, pregunta, problema, tarea, conclusión, nota) mediante reglas heurísticas y/o un clasificador (supervisado o LLM zero-shot).

Nota: El criterio de fragmentación presentado aquí corresponde a la implementación inicial. Para conocer las optimizaciones posteriores que mejoraron significativamente la recuperación contextual, ver Sección 3.4 "Optimización de la Fragmentación Semántica".

3.1.4 Mapeo fragmento → nodo y creación de aristas

- Cada fragmento **f** genera un nodo **v** ∈ **V** con probabilidad inicial Pv(f) calculada en función de la similitud semántica con la consulta y su relevancia temporal (ver sección de Contextos Temporales).
- **Cálculo de pesos estructurales (Westructural)**: combinación entre coincidencia de keywords y similitud por embeddings. Ver más detalle en la sección Promedio ponderado con parámetro α (propuesta extendida)
- **Arista y peso temporal:** crear arista (vi, vj) si W estructural (vi, vj) ≥ τ (umbral de conexión o similitud) y calcular su peso efectivo como:

  Fórmula adaptativa según intención:

  Para consultas con intención temporal: Pe(vi,vj) = R temporal(vi,vj) × F refuerzo × (1 + W estructural (vi,vj))

Para consultas estructurales: Pe(vi,vj) = W estructural(vi,vj) × (1 + R temporal(vi,vj) × F refuerzo)

Donde:

- Pe(vi,vj): Peso efectivo de la arista entre los nodos vi y vj
- W estructural: Similitud estructural (combinación de semántica por embeddings + keywords)
- R temporal: Relevancia temporal normalizada (proximidad en el tiempo)
- F refuerzo: Factor de refuerzo adaptativo que intensifica la importancia temporal

Una decisión clave tomada fue: En lugar de aplicar una única fórmula fija que era lo que tenía las primeras versiones del prototipo Pe(vi,vj) = W estructural(vi,vj) × (1 + R temporal(vi,vj) × F refuerzo), se implementó una estrategia diferenciada según la intención detectada en la consulta del usuario. Esto requirió desarrollar un módulo de análisis de consultas con LLM que clasifica automáticamente la intención (temporal/estructural/mixta) antes del cálculo de relevancia, permitiendo que el sistema priorice la dimensión más pertinente según el contexto de búsqueda. Esto favorece a que no se utilice patrones específicos (utilizado en las primeras versiones) y se pueda a través del LLM detectar preguntas temporales de forma más precisa, para luego definir la ventana temporal (rango de fecha donde los contextos temporales deben estar) para proporcionar los contextos adecuados al LLM para que las respuestas sean más adecuadas a la intención temporal del usuario.

A diferencia de versiones iniciales, las versiones finales de AGPC incorpora la propagación explícita de la *intención detectada* hacia las funciones de construcción del grafo. Esto permite que los nodos y aristas generados en el proceso de fragmentación reflejen fielmente si la consulta está orientada al plano temporal, estructural o mixto(temporal y estructural).

De esta manera, la estructura del grafo no solo representa relaciones semánticas y de co-ocurrencia, sino también la orientación cognitiva del análisis, favoreciendo la adaptabilidad del sistema a distintos tipos de consultas y escenarios de interacción.

3.1.5 Carga masiva de datasets

Para escalar a decenas o cientos de conversaciones, el proceso propuesto es:

1. **Validación de esquema**: comprobar que cada conversación tenga **título**, **contenido**, **fecha**, y **participantes** (rechazo o marcado de errores si faltan campos).
2. **Pipeline**:
    - **Map (paralelo)**: cada conversación se procesa de forma independiente: fragmentación → extracción de metadatos → generación de embedding → escritura provisional de nodos.
3. **Control**: usar **hash** y **fragmento_id** para evitar duplicados y permitir reintentos.

4. **Reporte**: logs de ejecuciones con conteo **N conversaciones**, **N fragmentos_generados**, **errores**, **tiempo total**.

3.1.6 Visualización multinivel:

Propósito: reducir la carga cognitiva y hacer auditables las decisiones del agente.

- **Vista Macro (conversación ↔ conversación)**: nodos representan conversaciones completas; grosor de arista = número de conexiones entre fragmentos. Útil para ver panoramas y relaciones inter-proyectos.
- **Vista Micro Completa (todos los fragmentos y contextos)**: nodos = fragmentos(contextos); útil para análisis semántico más detallado y trazabilidad de decisiones.
- **Vista Micro Filtrada (conversación única)**: lo mismo que la Vista Micro Completa, pero únicamente con los fragmentos o contextos de esa conversación seleccionada.

3.1.7 Procesamiento de Documentos PDF como Fuente de Contextos

3.1.7.1 Motivación y Problema

Si bien la fragmentación de conversaciones permite capturar el conocimiento intercambiado entre participantes, existen escenarios donde la información relevante reside en **documentos formales**: contratos, informes técnicos, manuales de procedimientos, artículos académicos, normativas legales, especificaciones técnicas, entre otros.

En estos casos, las conversaciones pueden hacer **referencia** a dichos documentos sin reproducir su contenido completo. Por ejemplo:

> *"Revisar la cláusula 5.3 del contrato X antes de la reunión de mañana"*
> *"El manual técnico en la sección 2.4 explica el procedimiento"*

El problema: sin acceso directo al contenido de estos documentos, el sistema:

- No puede recuperar el contenido específico mencionado
- Pierde contexto técnico o legal crucial
- Depende de que los usuarios transcriban manualmente información de documentos

La solución propuesta integra el procesamiento de archivos PDF como **fuente de contextos**, permitiendo que el AGPC indexe, fragmente y relacione el contenido documental con las conversaciones de manera automática.

3.1.7.2 Características del Procesamiento de PDFs

1. Extracción de Texto

- Utiliza bibliotecas especializadas (PyPDF2, pdfplumber) para extraer texto legible
- Preserva la estructura del documento (párrafos, secciones)
- Maneja documentos de hasta 10 MB por cuestiones de rendimiento para ser un prototipo

2. Fragmentación Adaptativa

- Similar a conversaciones, pero ajustada a la estructura de documentos formales
- Parámetros típicos:
    - **Máximo 500 palabras por fragmento** (mayor que conversaciones)
    - Respeta límites de párrafos y secciones
    - Mantiene coherencia semántica en cada fragmento

3.1.8 Normalización Sigmoidea de Pesos en Relaciones Contextuales

3.1.8.1 Identificación del Problema Matemático

Durante las iteraciones iniciales del prototipo AGPC, se detectó una inconsistencia crítica en el cálculo de los pesos efectivos asignados a las aristas del Grafo Contextual Probabilístico (GCP). El sistema empleaba la siguiente formulación para cuantificar la intensidad de las relaciones entre contextos:

Ecuación original:

$$Pe(vi,vj) = W\ estructural(vi,vj) \times (1 + R\ temporal(vi,vj) \times F\ refuerzo)$$

Donde:

- **W estructural(vi,vj)**: Similitud estructural normalizada entre nodos vi y vj, con valores en [0,1]
- **R temporal(vi,vj)**: Relevancia temporal normalizada, con valores en [0,1]
- **F refuerzo**: Factor de refuerzo temporal dependiente de la intención de la consulta
- **Pe(vi, vj)**: Peso efectivo de la arista entre vi y vj

Análisis del problema:

Esta formulación presentaba una limitación matemática fundamental: cuando los valores de similitud estructural y relevancia temporal eran elevados (cercanos a 1), y el factor de refuerzo era significativo (Fr > 1), el peso efectivo resultante excedía el intervalo válido probabilístico [0,1].

Ejemplo ilustrativo:

W estructural = 0,85 (alta similitud semántica) R temporal = 0,90 (alta proximidad temporal) F refuerzo = 1,5 (refuerzo medio)

Pe = 0,85 × (1 + 0,90 × 1,5) Pe = 0,85 × (1 + 1,35) Pe = 0,85 × 2,35 Pe = 1,9975, redondeando es 2,0

Este resultado (aproximadamente 200% de peso) es matemáticamente incorrecto con la interpretación probabilística de las aristas, donde 1.0 debe representar la máxima intensidad de relación posible entre dos contextos.

Consecuencias observadas:

1. **Degradación en propagación de activación**: Los algoritmos de propagación multiplicaban pesos superiores a 1, causando amplificación exponencial en lugar de atenuación progresiva. Esto violaba el principio teórico de que la activación debe decrecer monótonamente con la distancia al nodo fuente (Tong et al., 2006).
2. **Distorsión en rankings de relevancia**: Los contextos con pesos >1 dominaban los resultados de búsqueda, independientemente de su pertinencia real. El sistema priorizaba incorrectamente contextos con alto refuerzo temporal sobre contextos estructuralmente más relevantes.
3. **Incoherencia semántica**: Un peso mayor a 1 como por ejemplo 1.8 carece de interpretación intuitiva en el marco teórico de grafos probabilísticos. No puede interpretarse como probabilidad condicional ni como medida de similitud normalizada.
4. **Incompatibilidad con análisis de grafos**: Las herramientas estándar de análisis de grafos (centralidad, clustering, shortest paths) asumen pesos acotados en [0,1] o valores positivos sin límite superior definido, pero no valores que deberían ser probabilidades y exceden 1.

3.1.8.2 Solución Implementada: Normalización Sigmoidea

Para resolver esta inconsistencia, se implementó una transformación de normalización basada en una variante simplificada de la función sigmoidea. Esta función mapea biyectivamente el conjunto de números reales positivos al intervalo abierto (0,1) mediante una transformación monotónica creciente.

Formulación matemática:

<u>Paso 1 - Cálculo del peso bruto:</u>

Pe_efectivo(vi, vj) = W estructural(vi,vj) × (1 + R temporal(vi,vj) × F refuerzo)

<u>Paso 2 - Normalización sigmoidea:</u>

Pe_norm(vi, vj) = Pe_efectivo(vi, vj) / (1 + Pe_efectivo(vi, vj))

Fundamentación teórica:

La función de normalización empleada:

$f(x) = x / (1 + x)$, donde $x \in \mathbb{R}^+$

Constituye una aproximación de la función logística estándar $\sigma(x) = 1/(1+e^{-x})$, con propiedades matemáticas deseables para el contexto de grafos probabilísticos:

- **Acotamiento estricto**: $\forall x \in \mathbb{R}^+$, $f(x) \in (0,1)$
  - $\lim(x \to 0^+) f(x) = 0$
  - $\lim(x \to \infty) f(x) = 1$
  - f es estrictamente creciente
- **Preservación de orden**: $\forall x_1, x_2 \in \mathbb{R}^+$, si $x_1 < x_2$ entonces $f(x_1) < f(x_2)$
- **Continuidad y diferenciabilidad**: f es infinitamente diferenciable en $\mathbb{R}^+$, con derivada: $f'(x) = 1/(1+x)^2 > 0$
- **Punto fijo en x=0**: $f(0) = 0$, preservando la ausencia de relación cuando corresponde
- **Asíntota en 1**: La función se aproxima asintóticamente a 1 sin alcanzarlo, reflejando que ninguna relación puede ser absolutamente perfecta.

Tabla 1. Comparación con la formulación original

Caso	Peso efectivo(sin normalizar)	Peso Normalizado	Interpretación
Relación débil	0,3	0,231	23,1% de intensidad máxima
Relación media	0,8	0,444	44,4% de intensidad máxima
Relación fuerte	1,5	0,600	60,0% de intensidad máxima
Relación muy fuerte	2,5	0,714	71,4% de intensidad máxima
Ejemplo proporcionado en Análisis del problema	1,9975	0,666	66,6% de intensidad máxima

3.1.8.3 Impacto en Funcionalidad del Sistema

Propagación de activación:

Antes (pesos sin normalizar):

Nodo inicial: activación = 1,0 → Vecino A (peso=1,8): activación = 1,0 × 1,8 = 1,8. (amplificación) → Vecino B (peso=1,5): activación = 1,8 × 1,5 = 2,7 (explosión).

Después (pesos normalizados):

Nodo inicial: activación = 1,0 → Vecino A (peso=0,643): activación = 1,0 × 0,643 = 0,643 → Vecino B (peso=0,545): activación = 0,643 × 0,545 = 0,350.

Resultado: La propagación luego de la normalización decae monotónicamente como se espera teóricamente (Tong et al., 2006).

Rankings de búsqueda:

Consideremos una consulta estructural: *"¿Cuáles son los precedentes sobre cláusulas de confidencialidad?"*

Antes:

Contexto A (doc. reciente + alta similitud): peso_efectivo = 1,85 → sin orden definido

Contexto B (doc. antiguo + muy alta similitud): peso_efectivo = 1,08 → sin orden definido

Contexto C (doc. reciente + baja similitud): peso_efectivo = 0,90 → sin orden definido

Problema: Los valores mayores a 1 distorsionan el ranking. No está claro si 1,85 es "mucho mejor" que 1,08.

Después:

Contexto A: peso_norm = 0,649 → posición 1

Contexto B: peso_norm = 0,519 → posición 2

Contexto C: peso_norm = 0,474 → posición 3

Resultado: Orden claro y magnitudes interpretables como porcentajes de relevancia.

3.1.8.4 Discusión: Ventajas, Limitaciones y Trabajo Futuro

Ventajas de la normalización sigmoidea:

1. **Consistencia matemática:** Todos los pesos están acotados en (0,1), permitiendo interpretación probabilística.
2. **Estabilidad numérica:** Elimina riesgo de overflow en propagación iterativa y mejora convergencia de algoritmos de grafos.
3. **Interpretabilidad mejorada:** Los pesos pueden comunicarse como porcentajes (ej: "75% de relevancia"), facilitando explicabilidad del sistema.
4. **Compatibilidad con teoría de grafos:** Los pesos normalizados son compatibles con algoritmos estándar de análisis de grafos (PageRank, Louvain, shortest paths ponderados).

Limitaciones identificadas:

1. **Compresión de rango dinámico:** La normalización comprime diferencialmente el espacio de valores. Específicamente:
   - Valores bajos (peso_efectivo < 0,5) se expanden relativamente: f(0,3) = 0,231 (pérdida de 23%)
   - Valores altos (peso_efectivo > 1,5) se comprimen significativamente: f(2,0) = 0,667 (compresión del 67%)

   Esto puede reducir la capacidad del sistema para discriminar entre relaciones muy fuertes.

2. **Pérdida de linealidad:** La transformación sigmoidea es no lineal. Esto significa que duplicar el peso bruto no duplica el peso normalizado:
   - f(0,5) = 0,333
   - f(1,0) = 0,500 (no es el doble)

   Esta no linealidad puede afectar algoritmos que asumen proporcionalidad directa entre pesos.

Líneas de investigación futuras:

1. **Normalización paramétrica adaptativa:** Investigar variantes de la forma: f(x, α) = x^α / (1 + x^α) donde α controla la curvatura de la función (α=1 es el caso actual). Valores α<1 comprimen menos, valores α>1 comprimen más.
2. **Normalización por contexto:** Implementar normalización relativa al dominio: f_contextual(x) = (x - min_domain) / (max_domain - min_domain) donde min_domain y max_domain se calculan dinámicamente según los pesos observados en cada sesión.

3. **Aprendizaje de función de normalización:** Entrenar una red neuronal que aprenda la transformación óptima de pesos basándose en feedback de relevancia explícito de usuarios.
4. **Análisis comparativo formal:** Realizar estudio empírico comparando normalización sigmoidea vs. otras alternativas (tangente hiperbólica, min-max scaling, z-score) en métricas de precisión@K y recall.

La implementación actual proporciona una base matemáticamente sólida y funcional, que puede refinarse iterativamente según las necesidades de aplicaciones específicas del AGPC en dominios de alta complejidad contextual.

3.2 Actualizador Incremental de Grafo

Uno de los principales desafíos en la construcción de un Grafo Contextual Probabilístico (GCP) es la escalabilidad. En el enfoque tradicional (utilizado en las primeras versiones del prototipo), cada vez que se incorpora un nuevo fragmento o contexto de información, el sistema debe recalcular todas las relaciones posibles entre ese nodo y el resto, lo que implica una complejidad de orden cuadrático $O(n^2)$. En escenarios con pocos nodos, esta estrategia es aceptable, pero en grafos más grandes, se vuelve ineficiente e incluso inutilizable.

Problema original ($O(n^2)$):

- Cada nuevo contexto requiere volver a comparar todos los nodos entre sí.
- Con 100 nodos: aproximadamente 10.000 operaciones.
- Con 1.000 nodos: alrededor de 1.000.000 operaciones.
- A medida que crece el grafo, los tiempos de actualización se vuelven exponencialmente más altos, degradando la experiencia del usuario.

Solución incremental ($O(n)$):
El *Actualizador Incremental de Grafo* introduce una optimización importante: al agregar un nuevo fragmento, solo se calculan sus relaciones con los nodos ya existentes, evitando recomputar conexiones entre nodos antiguos.

- Con 100 nodos: 100 operaciones.
- Con 1.000 nodos: 1.000 operaciones.

  Esto reduce drásticamente el tiempo de actualización, permitiendo que el sistema escale de manera lineal en lugar de cuadrática.

Funcionamiento práctico:

1. El sistema fragmenta la nueva conversación en partes atómicas.
2. Para cada fragmento nuevo:
    - Calcula similitudes únicamente con nodos existentes.

    - Filtra por umbral mínimo de similitud.
    - Crea aristas solo si la conexión es relevante.
3. Registra estadísticas del proceso: número de conexiones creadas, tiempo de ejecución y total de relaciones en el grafo.

Impacto en el sistema:

- *Escalabilidad:* permite manejar datasets con mayores fragmentos o contextos, obteniendo tiempos de actualización aceptables.
- *Usabilidad:* marca la diferencia entre las primeras versiones del prototipo y una versión más orientada a producción donde se busca la implementación a gran escala.

En síntesis, el *Actualizador Incremental de Grafo* transforma la arquitectura del AGPC en una solución más escalable, superando el cuello de botella de la complejidad cuadrática y acercando el prototipo a un nivel más de producción en entornos con volúmenes de datos mayores.

3.3 Optimizaciones de Rendimiento

3.3.1 Contexto y Problemática Inicial

Si bien la arquitectura conceptual del AGPC presentada en las secciones anteriores es matemáticamente sólida y teóricamente válida, las primeras iteraciones del prototipo presentaron serios problemas de rendimiento al escalar a volúmenes de datos más realistas. Estos problemas no invalidan el diseño conceptual, pero sí evidencian la necesidad de optimizaciones técnicas para hacer viable su aplicación práctica.

Específicamente, dos operaciones relevantes presentaban cuellos de botella:

1. **Carga masiva de datasets:** El procesamiento de 50 conversaciones (aproximadamente 250-300 fragmentos de contexto) requería entre 90 y 120 minutos de tiempo de ejecución continuo.
2. **Recálculo de relaciones:** Con 272 nodos en el grafo contextual, la operación de recálculo completo de todas las conexiones entre contextos podría extenderse entre 15 y 30 minutos. Este recálculo de relaciones se lo utiliza cuando se cambia el umbral de similitud, que si disminuimos el mismo obtenemos más relaciones entre los nodos que tienes en nuestro prototipo y en caso de aumentar este umbral, obtenemos menos relaciones

Estos tiempos no son adecuados a la hipótesis central de este trabajo: que el AGPC puede servir como prototipo práctico de gestión contextual en dominios complejos como la auditoría legal, la gestión de proyectos empresariales o la investigación académica, donde la información debe ser accesible de manera ágil y eficiente.

Causa Raíz del Problema

El análisis del comportamiento del prototipo identificó tres factores principales que explicaban estos tiempos de ejecución:

1. Procesamiento secuencial con operaciones costosas repetidas

El prototipo procesaba cada fragmento de información de forma individual y aislada, lo que generaba múltiples llamadas repetitivas a la base de datos vectorial (ChromaDB) para tareas que podrían realizarse de manera agrupada. Es análogo a enviar un correo electrónico individual por cada palabra de un mensaje, en lugar de enviar el mensaje completo de una sola vez.

2. Operaciones de entrada/salida frecuentes e ineficientes

Después de procesar cada fragmento individual, el sistema guardaba todo su estado en el disco duro, interrumpiendo constantemente el flujo de procesamiento que debería mantenerse en la memoria principal (RAM). Esta estrategia es comparable a guardar un documento de Word después de escribir cada palabra, en lugar de mantenerlo en memoria y guardarlo solo al finalizar.

3. Cálculo de similitudes con complejidad cuadrática

El enfoque original calculaba las similitudes entre contextos creando documentos temporales en la base de datos vectorial, realizando una consulta por cada par de nodos posible, y luego eliminando esos documentos temporales. Este proceso resultaba en una complejidad computacional de $O(n^2 \times \log m)$, donde n representa el número de nodos y m la dimensión de los vectores de representación semántica (embeddings).

Para ilustrar la magnitud del problema: con 272 nodos en el grafo, el sistema realizaba más de 36,000 operaciones individuales de comparación, cada una con su propio overhead de creación, consulta y eliminación de datos temporales.

Impacto en la Viabilidad del Sistema

Estos problemas de rendimiento no son meramente técnicos o estéticos; representan una barrera fundamental para las pruebas prácticas del AGPC.

La presente sección documenta las optimizaciones implementadas para transformar el AGPC de un prototipo con un mejor rendimiento para poder realizar pruebas con datos mayores, logrando reducciones en los tiempos de operación que se encuentran entre 50% y 95%.

3.3.2 Optimización: Carga Eficiente de Datasets

3.3.2.1 El Problema del Procesamiento Fragmentado

Para comprender el problema, consideremos el proceso de agregar una conversación con 6 fragmentos a un grafo que ya contiene 272 nodos existentes (contextos previamente cargados).

En el enfoque original, el sistema realizaba las siguientes operaciones **para cada uno de los 6 fragmentos**:

1. **Crear el nodo en la estructura del grafo** (operación rápida)
2. **Generar su representación vectorial** (embedding) mediante una llamada individual al modelo de lenguaje
3. **Calcular la similitud semántica** con cada uno de los 272 nodos existentes
4. **Guardar todo el estado del grafo** en el disco duro
5. **Actualizar el sistema de propagación** de activación

Cuantificando el impacto:

Para esa conversación de ejemplo (6 fragmentos, 272 nodos preexistentes), el sistema realizaba:

- **6 llamadas individuales** a ChromaDB para generar embeddings
- **1,632 cálculos de similitud** (6 × 272) con overhead de creación y eliminación de documentos temporales
- **6 escrituras completas** del grafo a disco
- **6 actualizaciones del propagador**

Con estos movimientos, cargar un dataset de 50 conversaciones (250-300 fragmentos totales) proyectaba un tiempo de 90 a 120 minutos, lo cual fue confirmado experimentalmente.

3.3.2.2 Estrategia de Solución: Procesamiento por Lotes

La clave de la optimización consiste en un principio fundamental de la informática: **el procesamiento por lotes es más eficiente que el procesamiento individual cuando las operaciones son similares y pueden agruparse**.

Pensemos en una analogía cotidiana: si necesitas hornear 20 galletas, es mucho más eficiente hornearlas todas juntas en una bandeja grande que hornear cada galleta individualmente en 20 ciclos separados del horno. El principio es el mismo en el procesamiento computacional.

A) Generación Agrupada de Representaciones Vectoriales (Batch Embedding)

En lugar de generar la representación vectorial (embedding) de cada fragmento de forma individual, el sistema optimizado:

1. **Acumula** todos los textos de los fragmentos de una conversación
2. **Envía** la colección completa al modelo de embeddings en una sola operación
3. **Recibe** todas las representaciones vectoriales simultáneamente
4. **Indexa** todas en la base de datos vectorial de una vez

Fundamentación técnica:

Los modelos de lenguaje basados en la arquitectura transformer, como el all-MiniLM-L6-v2 utilizado en el AGPC (Reimers & Gurevych, 2019), están diseñados internamente para procesar múltiples textos de forma simultánea mediante:

- **Paralelización de operaciones matriciales:** Los cálculos matemáticos se ejecutan de forma vectorizada aprovechando las capacidades de procesamiento paralelo de la CPU o GPU.
- **Reducción de overhead de inicialización:** El modelo se carga y configura una sola vez, en lugar de reinicializarse para cada texto individual.
- **Aprovechamiento de caché:** Los estados intermedios del modelo se mantienen en memoria y se reutilizan eficientemente dentro del lote de procesamiento.

Ganancia medida: De 6 llamadas individuales a 1 llamada agrupada, lo que produce una **reducción del 75%** en tiempo de generación de embeddings.

B) Cálculo Eficiente de Similitudes mediante Búsqueda Vectorial Indexada

El problema del cálculo de similitudes merece atención, ya que representa el cuello de botella más significativo del sistema original.

El enfoque original realizaba el siguiente proceso para cada nuevo fragmento y cada nodo existente:

1. Crear un documento temporal en ChromaDB con la representación vectorial del fragmento nuevo
2. Realizar una consulta con el texto del nodo existente
3. Extraer la puntuación de similitud del resultado
4. Eliminar el documento temporal

Este enfoque implicaba 272 ciclos completos de crear-consultar-eliminar para cada fragmento nuevo, resultando en miles de operaciones para una sola conversación.

El enfoque optimizado aprovecha una característica fundamental de ChromaDB: su implementación del algoritmo HNSW (Hierarchical Navigable Small World) desarrollado por Malkov y Yashunin (2016) para búsqueda aproximada de vecinos más cercanos.

En lugar de realizar 272 operaciones individuales, el sistema optimizado:

1. **Ejecuta una única consulta** que solicita los k nodos más similares
2. **Recibe todas las similitudes** en una sola respuesta
3. **Procesa los resultados** de forma vectorizada para crear las aristas del grafo

Comprendiendo el algoritmo HNSW:

HNSW es una estructura de datos que organiza la información en múltiples capas jerárquicas, similar a un edificio de varios pisos:

- **Pisos superiores:** Contienen pocos nodos muy espaciados, permitiendo "saltos largos" en el espacio de búsqueda
- **Pisos inferiores:** Contienen cada vez más nodos con conexiones más densas
- **Piso base:** Contiene todos los nodos con sus conexiones detalladas

Cuando el sistema busca vectores similares, comienza en el piso superior (búsqueda rápida pero aproximada) y desciende progresivamente hasta el piso base (búsqueda precisa y refinada). Este enfoque logra una complejidad computacional de O(log n) en promedio, en contraste con O(n) de la búsqueda lineal tradicional.

Ganancia medida: De 272 consultas individuales a 1 consulta agrupada **reducción del 96,3% aproximadamente** en tiempo de cálculo de similitudes.

C) Escritura Diferida: Optimización de Operaciones de Entrada/Salida

Las operaciones de lectura y escritura en disco (conocidas técnicamente como operaciones de I/O por Input/Output) son considerablemente más lentas que las operaciones en memoria principal (RAM). Esta diferencia de velocidad es fundamental para comprender la importancia de esta optimización.

Además del tiempo, cada operación de escritura a disco implica:

1. **Serialización:** Convertir las estructuras de datos en memoria a un formato almacenable (por ejemplo, JSON o pickle en Python)
2. **Flush de buffers:** Forzar que los datos pasen de los buffers temporales del sistema operativo al dispositivo físico
3. **Sincronización con el sistema de archivos:** Actualizar metadatos como índices de nodos y registros de transacciones
4. **Confirmación de escritura:** Esperar la confirmación física de que los datos fueron escritos correctamente

El enfoque optimizado implementa el principio de **escritura diferida (deferred I/O)**, que consiste en:

1. **Procesar todos los fragmentos** manteniendo los datos exclusivamente en memoria RAM
2. **Acumular todos los cambios** sin interrupciones por operaciones de disco
3. **Realizar una única escritura consolidada** al finalizar todo el procesamiento

Esta estrategia es análoga a tomar notas en papel durante una reunión larga y transcribirlas todas al sistema digital al final, en lugar de pausar la reunión cada pocos minutos para actualizar el sistema.

D) Caché de Representaciones Vectoriales para Evitar Recálculos Redundantes

La cuarta optimización aborda un problema específico que surge en escenarios complejos: la aparición de contenido duplicado o repetido.

Motivación:

En situaciones donde se recargan datasets previamente procesados, se corrigen errores en conversaciones, o simplemente existen fragmentos con contenido idéntico, regenerar las representaciones vectoriales (embeddings) es computacionalmente redundante.

Los modelos de embeddings son **deterministas**: dado el mismo texto de entrada, siempre producen la misma representación vectorial de salida. Esta propiedad permite implementar un sistema de caché eficiente.

Implementación conceptual:

El sistema mantiene un diccionario (estructura de datos clave-valor) donde:

- **Clave:** Texto normalizado del fragmento (limpio de espacios extra, en minúsculas)
- **Valor:** Representación vectorial previamente calculada

Antes de generar un nuevo embedding:

1. El sistema normaliza el texto
2. Consulta si ya existe en el caché
3. Si existe, reutiliza el embedding almacenado (acceso en milisegundos)
4. Si no existe, genera el embedding y lo almacena para usos futuros

Beneficios observados:

1. **Detección automática de duplicados:** El sistema identifica inmediatamente contenido repetido
2. **Ahorro computacional:** Evita cálculos por texto duplicado

3. **Reducción de carga en ChromaDB:** Menos operaciones de indexación
4. **Overhead mínimo:** La consulta en el diccionario tiene complejidad O(1) en promedio

En datasets con tasas de duplicación del 10-15% (común en conversaciones que referencian documentos estándar o plantillas), esta optimización proporciona mejoras adicionales de 5-10% en el tiempo total.

3.3.2.3 Análisis de Impacto

1. **Reducción de tiempo por conversación:** El tiempo se redujo entre 63% y 77%, acercándose al umbral de viabilidad práctica.
2. **Llamadas a bases de datos:** La reducción del 99% en llamadas a ChromaDB representa una disminución dramática en la carga sobre la infraestructura de base de datos vectorial.
3. **Operaciones de disco:** La reducción del 83% en escrituras no solo mejora el tiempo de ejecución, sino que también prolonga la vida útil de dispositivos de almacenamiento (especialmente SSDs con ciclos de escritura limitados) y reduce el consumo energético del sistema.
4. **Escalabilidad demostrada:** La mejora del 50-72% en datasets de 50 conversaciones demuestra que las optimizaciones mantienen su efectividad al escalar, sin degradación del rendimiento.

Análisis de complejidad computacional:

Sistema Original:

- Complejidad: O(k × n × d)
- Donde: k = fragmentos nuevos, n = nodos existentes, d = dimensión de embeddings

Sistema Optimizado:

- Complejidad: O(k × d + k × log n)
- Simplificado: O(k × (d + log n))

Ejemplo numérico concreto:

Para k=6 fragmentos nuevos, n=272 nodos existentes, d=768 dimensión de embeddings:

- **Original:** O(6 × 272 × 768) = O(1,253,376) operaciones teóricas
- **Optimizado:** O(6 × 768 + 6 × log 272) = O(4,608 + 48) aproximadamente = O(4,656) operaciones teóricas

Esta disminución teórica se corresponde de manera cercana con los avances empíricos detectados, corroborando la exactitud del análisis algorítmico.

3.3.3 Optimización: Recálculo Eficiente de Relaciones

3.3.3.1 Contexto y Necesidad del Recálculo de Relaciones

El recálculo completo de relaciones en el grafo contextual es una operación necesaria en varios escenarios críticos:

Escenarios que requieren recálculo:

1. **Ajuste de parámetros:** Cuando se modifican los umbrales de similitud que determinan qué contextos están relacionados (por ejemplo, cambiar de 0.5 a 0.7 el umbral mínimo de similitud).
2. **Actualización del modelo de embeddings:** Cuando se actualiza a una versión más reciente del modelo de representación semántica que produce embeddings de mayor calidad o con diferentes propiedades.
3. **Reconstrucción por consistencia:** Cuando se detectan inconsistencias en el grafo o se desea garantizar que todas las relaciones se calcularon con la misma metodología y parámetros.
4. **Optimización experimental:** Durante la fase de investigación, para evaluar el impacto de diferentes configuraciones sobre la estructura del grafo resultante.

3.3.3.2 El Problema del Recálculo Cuadrático

En un grafo con 272 nodos, recalcular todas las relaciones posibles implica evaluar cada par de nodos para determinar si su similitud semántica supera el umbral establecido.

Cálculo del número de comparaciones:

El número de pares únicos en un grafo con n nodos es:

Pares = n × (n - 1) / 2

Para n = 272 cantidad de nodos:

Pares = 272 × 271 / 2 = 36,856 comparaciones únicas

(Dividimos por 2 porque la relación entre el nodo A y el nodo B es la misma que entre B y A, evitando duplicados simétricos, esto se debe que actualmente se cambió para usar direcciones bidireccionales en la visualización el grafo con el objetivo de disminuir la cantidad de arista y se pueda observar de mejor manera)

El proceso original para cada par:

1. **Crear un documento temporal** en ChromaDB con el embedding del primer nodo
2. **Realizar una consulta** con el texto del segundo nodo para obtener la similitud
3. **Extraer la puntuación** de similitud del resultado de la consulta

4. **Eliminar el documento temporal** para no contaminar la base de datos
5. **Evaluar si la similitud supera el umbral** y crear la arista si corresponde

Impacto cuantificado:

Para 36,856 pares de nodos:

- **36,856 creaciones** de documentos temporales
- **36,856 consultas** a ChromaDB
- **36,856 eliminaciones** de documentos temporales

En la práctica, con overhead del sistema operativo, latencias de red interna y garbage collection de Python, los tiempos medidos oscilaban entre **30 y 60 minutos** según la carga del sistema.

3.3.3.3 Solución Optimizada: Aprovechamiento de la Búsqueda Vectorial Nativa

La clave de la optimización radica en reconocer que ChromaDB **ya tiene todos los nodos indexados** en su estructura HNSW(Hierarchical Navigable Small World) interna. No es necesario crear documentos temporales ni realizar consultas individuales; podemos aprovechar directamente la capacidad de búsqueda de vecinos más cercanos.

Estrategia del enfoque optimizado:

Para cada nodo A en el grafo:

1. **Realizar una única consulta** que solicite los n-1 nodos más similares (todos los demás nodos)
2. **Recibir una lista ordenada** de similitudes con todos los nodos restantes en una sola respuesta
3. **Filtrar las conexiones** que superan el umbral de similitud establecido
4. **Crear las aristas** correspondientes en el grafo

Ventajas fundamentales:

1. **Eliminación de documentos temporales:** Los nodos ya están indexados permanentemente, no hay necesidad de generar ni eliminar documentos.
2. **Reducción de consultas:** De 36,856 consultas individuales a 272 consultas agrupadas (una por nodo).
3. **Aprovechamiento de la indexación:** Cada consulta navega eficientemente por la estructura HNSW ya construida, sin necesidad de reconstruir índices temporales.
4. **Procesamiento vectorizado:** Los resultados se procesan en bloque, aprovechando optimizaciones de bajo nivel del lenguaje de programación.

Tabla 2. Rendimiento del Recálculo de Relaciones

Métrica	**Sistema Original**	**Sistema Optimizado**	**Mejora Absoluta**	**Mejora Porcentual**
Tiempo total (272 nodos)	30-60 minutos	30-90 segundos	29–58 minutos menos	90-95%
Consultas a ChromaDB	36,856	272	36,584 menos	99,3%

Análisis de los resultados:

1. **Reducción de tiempo:** La mejora del 90-95% transforma una operación "ir a tomar café" en una operación "esperar unos segundos ", cambiando radicalmente la viabilidad de experimentación iterativa.
2. **Escalabilidad validada:** La reducción del 99,3% en operaciones de base de datos sugiere que el sistema puede escalar a grafos con mayor cantidad de nodos manteniendo tiempos razonables.

Análisis de complejidad computacional:

Sistema Original:

- Complejidad: $O(n^2 \times \log m)$
- Donde: n = número de nodos, m = dimensión de embeddings

Sistema Optimizado:

- Complejidad: $O(n \times \log n)$
- Gracias a la navegación eficiente en la estructura HNSW(Hierarchical Navigable Small World)

Ejemplo con numeros:

Para n=272 cantidad de nodos, m=768 dimensión de embeddings:

- **Original:** $O(272^2 \times \log 768) = O(74{,}000 \times 9.6)$ aproximadamente = O(710,400) operaciones
- **Optimizado:** $O(272 \times \log 272) = O(272 \times 8.1)$ aproximadamente = O(2,203) operaciones

Esta disminución teórica se nota de manera constante en las mediciones empíricas, corroborando el modelo de complejidad propuesto.

3.3.4 Fundamentos Teóricos de las Optimizaciones

3.3.4.1 El Problema de la Búsqueda en Espacios de Alta Dimensión

Para comprender mejor por qué las optimizaciones funcionan, es necesario entender el problema fundamental que resuelven: la búsqueda eficiente en espacios vectoriales de alta dimensión.

El desafío:

En el AGPC, cada fragmento de contexto se representa como un vector en un espacio de 768 dimensiones (para el modelo all-MiniLM-L6-v2(Modelo utilizado en el prototipo)). Cuando el sistema necesita encontrar los contextos más similares a una consulta, debe buscar entre potencialmente miles de vectores en este espacio de alta dimensión.

Enfoque naive: Búsqueda lineal exhaustiva

El método más simple y directo consiste en:

1. Calcular la distancia (o similitud) entre el vector de consulta y cada vector almacenado
2. Ordenar todos los resultados por distancia
3. Retornar los k vectores más cercanos

Complejidad del enfoque naive:

- Cálculo de distancias: O(n × d) donde n = número de vectores, d = dimensiones
- Ordenamiento: O(n × log n)
- **Total:** O(n × (d + log n))

Para n=1.000 vectores, d=768:

- Operaciones: O(1.000 × 768) = O(768.000) solo para calcular distancias
- Con ordenamiento: O(768.000 + 10.000) = O(778.000) operaciones

Este enfoque es prohibitivamente costoso para grafos grandes y búsquedas frecuentes.

Enfoque optimizado: Búsqueda Aproximada de Vecinos Más Cercanos (ANN)

Los algoritmos de ANN (Approximate Nearest Neighbor), como HNSW implementado en ChromaDB, sacrifican una pequeña cantidad de precisión (encuentran entre 95-99% de los verdaderos vecinos más cercanos) a cambio de una mejora significativa en velocidad.

3.3.4.2 El Algoritmo HNSW

HNSW puede entenderse de forma más sencilla mediante una analogía geográfica:

Analogía: Buscando una dirección en una ciudad

Imagina que estás en el centro de una gran ciudad y necesitas llegar a una dirección específica, pero solo conoces la dirección de destino, no cómo llegar.

Enfoque naive (búsqueda lineal): Recorrer sistemáticamente cada calle, una por una, hasta encontrar la dirección. Esto podría tomar días o semanas en una ciudad grande.

Enfoque HNSW (búsqueda jerárquica):

1. **Nivel 1 - Vista de satélite (capa superior):** Observas la ciudad desde muy alto y te desplazas hacia el distrito general donde está la dirección (saltos muy grandes, pero aproximados).
2. **Nivel 2 - Vista de helicóptero (capas medias):** Desciendes y observas el vecindario específico, refinando tu posición (saltos medianos, más precisos).
3. **Nivel 3 - Vista a nivel de calle (capa base):** Caminas por las calles del vecindario hasta encontrar la dirección exacta (búsqueda precisa en un área pequeña).

Traslación al contexto de HNSW:

HNSW construye múltiples "capas" de conexiones entre vectores:

- **Capa superior:** Pocos nodos altamente conectados que representan "hitos" en el espacio vectorial
- **Capas medias:** Densidad creciente de nodos con conexiones locales
- **Capa base:** Todos los vectores con conexiones detalladas a sus vecinos cercanos

Proceso de búsqueda:

1. La búsqueda comienza en la capa superior con un nodo aleatorio
2. Se mueve iterativamente hacia el vecino más cercano al objetivo
3. Cuando no hay mejora, desciende a la capa inferior
4. Repite el proceso hasta la capa base
5. Realiza una búsqueda local refinada en la vecindad final

Complejidad del algoritmo HNSW:

- **Búsqueda:** O(log n) en promedio (versus O(n) de búsqueda lineal)
- **Inserción:** O(log n) en promedio
- **Espacio adicional:** O(n × M) donde M es el número promedio de conexiones por nodo

Garantías de calidad:

- **Recall típico:** 95-99% (encuentra el 95-99% de los verdaderos vecinos más cercanos)

- **Configuración de parámetros:**
  - ef_construction: Mayor valor = mayor precisión durante construcción (más lento)
  - ef_search: Mayor valor = mayor precisión durante búsqueda (más lento)
  - `M`: Mayor valor = más conexiones = mayor precisión (más memoria)

Los valores por defecto de ChromaDB (ef_construction=200, M=16) proporcionan un balance excelente entre precisión y velocidad para la mayoría de aplicaciones.

Justificación de la decisiónes:

Para el caso de uso del AGPC (gestión de contextos legales, proyectos, etc.), encontrar el 95-99% de las relaciones relevantes es suficiente porque:

1. **Las relaciones más fuertes (>0.7 similitud) se capturan siempre:** El error del 1-5% se concentra en relaciones débiles marginales.
2. **El umbral de similitud actúa como filtro:** Relaciones con similitud <0.5 típicamente no son útiles para la propagación de activación.
3. **El costo de precisión perfecta no se justifica:** Una búsqueda significativamente más lenta para capturar el 1-5% adicional de relaciones débiles no mejora significativamente la utilidad del sistema.
4. **Validación empírica:** En pruebas con datasets legales, la diferencia en calidad de respuestas entre HNSW (95% recall) y búsqueda exacta fue imperceptible la hora de realizar consultas o preguntas al prototipo.

Trade-off 3: Flexibilidad vs. Rendimiento

Las optimizaciones también implican cierta reducción en la flexibilidad del sistema:

Antes de optimizaciones:

- Cada fragmento se procesaba independientemente
- Fácil agregar lógica personalizada por fragmento
- Más sencillo para debugging paso a paso

Después de optimizaciones:

- Los fragmentos se procesan en lotes
- La lógica personalizada debe aplicarse a todo el lote
- El debugging requiere examinar lotes completos

3.3.5 Síntesis y Trabajo Futuro

3.3.5.1 Logros Principales de las Optimizaciones

Las optimizaciones implementadas en el AGPC representan una transformación fundamental del sistema, cambiando su naturaleza de un prototipo conceptual a una herramienta con viabilidad práctica con el objetivo de poder compararlo con un RAG estándar. Los logros principales incluyen:

1. **Mejora Importante en Tiempos de Respuesta**
   - **Mejorar de velocidad en cargas de datasets**
   - **Recálculo de relaciones:** Reducción del 90-95% en tiempo total de operaciones
   - **Impacto práctico:** Operaciones que tomaban horas ahora se completan en minutos
2. **Mantenimiento de Funcionalidad y Precisión**
   - **Compatibilidad total:** Todas las funcionalidades originales se preservaron luego de mejorar el rendimiento tanto de carga de datasets como recalcular las relaciones entre los nodos
   - **Calidad de resultados:** Sin degradación observable en precisión de recuperación contextual
3. **Eficiencia de Recursos**
   - **Reducción de operaciones de base de datos:** 99% menos llamadas a ChromaDB
   - **Optimización de I/O:** 83% menos escrituras a disco

3.3.5.2 Limitaciones Actuales Reconocidas

Es importante reconocer que, pese a las mejoras sustanciales, el sistema optimizado mantiene limitaciones que deben ser consideradas para investigaciones futuras:

Limitación 1: Procesamiento Secuencial de Conversaciones

- **Descripción:** Aunque cada conversación se procesa eficientemente mediante batch processing interno, el procesamiento de múltiples conversaciones sigue siendo secuencial.
- **Impacto:** En datasets con cientos o miles de conversaciones, el tiempo total sigue siendo la suma de tiempos individuales.
- **Oportunidad de mejora:** Implementar paralelización a nivel de conversación podría reducir tiempos en proporción al número de núcleos CPU disponibles.

Limitación 2: Ausencia de Aceleración por GPU

- **Descripción:** El sistema utiliza exclusivamente CPU para todas las operaciones, incluyendo generación de embeddings.
- **Impacto:** Los modelos de embeddings modernos pueden ejecutarse 10-50x más rápido en GPU según el hardware.

- **Contexto de decisión:** La elección de solo CPU se justificó para maximizar compatibilidad y reducir requisitos de infraestructura en la fase de prototipo.
- **Oportunidad de mejora:** Implementar soporte opcional de GPU permitiría aceleración dramática en sistemas que dispongan de hardware adecuado.

Limitación 3: Búsqueda Aproximada Obligatoria

- **Descripción:** ChromaDB con HNSW siempre realiza búsqueda aproximada (ANN) sin opción de búsqueda exacta.
- **Impacto:** En el 1-5% de los casos, el sistema puede no identificar la relación óptima entre dos contextos.
- **Mitigación actual:** El umbral de similitud y la propagación de activación compensan parcialmente este efecto.
- **Oportunidad de mejora:** Implementar un modo de búsqueda exacta opcional (por ejemplo, usando FAISS IndexFlatIP) para casos donde la precisión perfecta sea crítica.

3.3.5.3 Consideraciones para Implementación en Producción

Si se desea llevar a AGPC a un entorno de producción, se recomienda considerar los siguientes aspectos:

Aspectos de Infraestructura:

1. **Requisitos mínimos (sistema actual optimizado):**
   - CPU: 4+ núcleos, 2.5+ GHz
   - RAM: 8 GB (16 GB recomendado para datasets >1,000 fragmentos)
   - Almacenamiento: SSD (preferiblemente NVMe para I/O óptimo)
   - Python 3.8+ con librerías científicas optimizadas (NumPy compilado con MKL)
2. **Requisitos recomendados (para máximo rendimiento):**
   - CPU: 8+ núcleos, 3.0+ GHz
   - RAM: 32 GB
   - GPU: NVIDIA con 8+ GB VRAM (para implementar limitaciones)
   - Almacenamiento: SSD NVMe con alta tasa de IOPS

Aspectos de Mantenimiento:

1. **Monitoreo de rendimiento:**
   - Implementar métricas de tiempo por operación
   - Alertas cuando tiempos superan umbrales establecidos
   - Logs estructurados para análisis de tendencias
2. **Gestión de crecimiento:**
   - Planificar migraciones cuando se acerquen puntos críticos identificados
   - Considerar particionamiento de grafos cuando se trabaje con múltiples dominios

- Implementar políticas de retención y archivado de contextos antiguos

3.3.5.4 Impacto de las Optimizaciones en la Validez de la Investigación

Es fundamental aclarar que las optimizaciones implementadas **no alteran la validez conceptual del AGPC**, sino que la fortalecen de tres maneras importantes:

1. Validación de Viabilidad Práctica:

Las optimizaciones demuestran que el modelo teórico del Grafo Contextual Probabilístico no solo contiene fórmulas matemáticas, sino que también es implementable en la práctica. Esto refuerza la hipótesis central del trabajo: que un enfoque probabilístico y estructurado de gestión contextual es superior a enfoques lineales tradicionales.

2. Transparencia Metodológica:

Documentar exhaustivamente el proceso de optimización proporciona una guía práctica para investigadores y desarrolladores que deseen replicar o extender este trabajo. La reproducibilidad es un pilar fundamental de la investigación científica.

3. Identificación de Trade-offs Fundamentales:

El análisis de las optimizaciones revela trade-offs significativos (memoria vs. velocidad, exactitud vs. eficiencia) que son intrínsecos al problema de gestión contextual, no solo al AGPC. Este conocimiento contribuye al cuerpo teórico del campo de estudio.

Si bien las optimizaciones de rendimiento descritas en esta sección mejoraron significativamente los tiempos de ejecución del prototipo, durante la validación experimental se identificó un problema más relevante que afectaba la calidad de la recuperación contextual. Este problema, relacionado con la estrategia de fragmentación semántica, y su solución se analizan en detalle en la siguiente sección.

3.4 Optimización de la Fragmentación Semántica para Mejora de Recuperación Contextual

3.4.1 Contexto y Problemática Identificada

Durante la fase de validación del prototipo AGPC, se identificó una discrepancia fundamental entre el comportamiento esperado del sistema y los resultados observados en pruebas prácticas. Específicamente, al realizar consultas sobre "Amparo por mora administrativa" un tema para el cual el dataset contenía 8 conversaciones específicas el sistema retornaba contextos completamente irrelevantes relacionados con acoso laboral y ordenanzas municipales, sin identificar ninguno de los casos de amparo existentes.

Esta anomalía resultaba desconcertante dado que:

1. Los contextos relevantes existían en la base de datos: Verificaciones directas confirmaron la presencia de 16 fragmentos conteniendo el término "amparo" en ChromaDB.
2. El sistema de embeddings funcionaba correctamente: Pruebas con casos sintéticos simples demostraron que el modelo **all-MiniLM-L6-v2** generaba representaciones vectoriales correctas.
3. La arquitectura semántica era sólida: Las búsquedas por similitud coseno operaban según lo esperado en condiciones controladas.

La investigación sistemática del problema reveló que la causa raíz no residía en la indexación, el cálculo de similitudes o la configuración de ChromaDB, sino en la estrategia de fragmentación del contenido de las conversaciones.

3.4.2 Diagnóstico: Fragmentación Excesiva y Pérdida de Contexto Semántico

3.4.2.1 Análisis del Comportamiento Original

El algoritmo de fragmentación inicial implementaba una lógica de división basada en detección de cambios de hablante, creando un nuevo fragmento cada vez que se identificaba un patrón como "Nombre:" en el contenido de las conversaciones. Para la conversación de amparo por mora administrativa, esto resultaba en:

Conversación original (503 caracteres, 80 palabras):

- ➢ Ciudadano: Hace 18 meses pedí subsidio en ANSES y no me resuelven nada.
- ➢ Abogado: ¿Hizo reclamos formales?
- ➢ Ciudadano: Sí, fui 4 veces. Siempre me dicen que está en trámite.
- ➢ Abogado: 18 meses es mora inadmisible. El plazo razonable es 60 días.
- ➢ Ciudadano: ¿Qué puedo hacer?
- ➢ Abogado: Presentar amparo por mora. El juez ordena al organismo que resuelva en plazo perentorio.
- ➢ Ciudadano: ¿Cuánto tarda el amparo?
- ➢ Abogado: Es rápido, entre 30-60 días. El Estado debe cumplir órdenes judiciales o enfrenta multas.

Fragmentación generada (5 fragmentos):

- Fragmento 1 (13 palabras): Ciudadano: Hace 18 meses pedí subsidio...
- Fragmento 2 (12 palabras): Ciudadano: Sí, fui 4 veces...
- Fragmento 3 (12 palabras): Abogado: 18 meses es mora inadmisible...
- Fragmento 4 (15 palabras): Abogado: Presentar amparo por mora...
- Fragmento 5 (15 palabras): Abogado: Es rápido, entre 30-60 días...

3.4.2.2 Impacto en la Calidad de los Embeddings

Los modelos de embeddings como sentence-transformers requieren contexto semántico suficiente para generar representaciones vectoriales significativas. Fragmentos de 12-15 palabras presentan varias limitaciones críticas:

1. Pérdida de coherencia narrativa: La pregunta del ciudadano se separaba de la respuesta del abogado, fragmentando la unidad semántica del intercambio.
2. Ambigüedad léxica no resuelta: El fragmento aislado "Abogado: Presentar amparo por mora" carece del contexto que establece por qué se recomienda un amparo (el retraso administrativo de 18 meses).
3. Insuficiencia para desambiguación: Sin el contexto completo, términos polisémicos(palabras que tienen más de un significado, aunque comparten una misma forma escrita y un origen etimológico común) como "mora" (retraso legal vs. fruta) no pueden desambiguarse efectivamente.

3.4.2.3 Evidencia Experimental del Problema

Pruebas diagnósticas comparativas revelaron la magnitud del problema:

Métrica	**Fragmentación Original**	**Fragmentación Óptima**
Fragmentos por conversación (80 palabras)	5 fragmentos	1 fragmento
Palabras promedio por fragmento	13.6 palabras	80 palabras
Casos de amparo identificados (8 totales)	0/8 (0%)	8/8 (100%)
Precisión en top-5 resultados	0% relevantes	100% relevantes
Distancia coseno para consulta exacta	0.460 (irrelevante)	0.298 (relevante)

Observación crítica: Incluso cuando ChromaDB contenía el texto exacto "amparo por mora", la fragmentación inadecuada impedía su recuperación, evidenciando que el problema no era de recuperación técnica, sino de representación semántica.

3.4.3 Solución Implementada: Fragmentación Adaptativa con Contexto Mínimo

3.4.3.1 Principios de Diseño

La solución implementada se basa en tres principios fundamentales derivados de la literatura sobre recuperación de información y la investigación empírica:

Principio 1: Contexto Mínimo Garantizado

Cada fragmento debe contener un mínimo de 50 palabras para asegurar suficiente contexto semántico. Este umbral se estableció basándose en:

- Estudios de rendimiento de modelos sentence-transformers (Reimers & Gurevych, 2019)
- Experimentación empírica con el dataset legal
- Balance entre granularidad y coherencia

Principio 2: Preservación de Unidades Semánticas

Los fragmentos deben respetar unidades de significado completas (intercambios, pregunta-respuesta, explicaciones causales, argumentaciones). Un cambio de hablante no justifica fragmentación si aún no se alcanzó el contexto mínimo.

Principio 3: Límite Superior para Enfoque

Se mantiene un máximo de 300 palabras por fragmento para prevenir dilución semántica. Fragmentos excesivamente largos pueden incluir múltiples temas, reduciendo la especificidad de los embeddings.

3.4.3.2 Algoritmo Optimizado

El algoritmo implementa una estrategia de acumulación con umbral dual que puede describirse mediante el siguiente proceso:

Estrategia general del algoritmo:

El sistema procesa el texto de una conversación línea por línea, acumulando el contenido en un fragmento temporal. La decisión de crear un nuevo fragmento no se toma únicamente al detectar un cambio de hablante, sino que considera simultáneamente tres condiciones:

1. Acumular líneas de diálogo: El sistema va agregando cada línea de texto a un fragmento en construcción, contando continuamente el número de palabras acumuladas.
2. Evaluar condiciones de corte: Antes de cada potencial división, el algoritmo verifica si se cumple alguna de las siguientes situaciones:

   a. Cambio de hablante detectado y se ha alcanzado el mínimo de palabras requeridas
   b. El fragmento actual excede el máximo de palabras permitidas
3. Gestionar fragmentos residuales: Al finalizar el procesamiento, si quedan fragmentos muy pequeños (menos de 20 palabras), estos se fusionan con fragmentos adyacentes para evitar unidades de información demasiado fragmentadas.

Lógica de decisión:

Para cada línea procesada del texto original, el sistema ejecuta el siguiente razonamiento:

Si se detecta un cambio de hablante (por ejemplo, de "Abogado:" a "Ciudadano:") Y el fragmento actual contiene palabras:

- ➔ Si el número de palabras acumuladas es mayor o igual al mínimo establecido (50 palabras):
  - ◆ Se crea un nuevo fragmento con el contenido acumulado (el contexto es suficiente para generar un embedding significativo)
- ➔ Si el número de palabras es menor al mínimo:
  - ◆ Se continúa acumulando contenido sin crear el fragmento (el contexto todavía es insuficiente)
- ➔ Si el número de palabras acumuladas alcanza o supera el máximo permitido (300 palabras):
  - ◆ Se fuerza la creación de un fragmento inmediatamente (se alcanzó el límite superior para evitar dilución semántica)

Ventajas de este enfoque dual:

1. **Flexibilidad contextual:** El sistema puede generar fragmentos de tamaños variables según el contenido real, no según reglas rígidas.
2. **Preservación semántica:** Los intercambios pregunta-respuesta se mantienen juntos cuando es necesario para la coherencia.
3. **Control de calidad:** Los umbrales superior e inferior garantizan que ningún fragmento sea ni demasiado pequeño (pobre en contexto) ni demasiado grande (semánticamente difuso).
4. **Adaptabilidad:** El algoritmo funciona tanto con conversaciones breves (una sola pregunta-respuesta) como con conversaciones extensas (múltiples intercambios sobre un tema).

3.4.3.3 Estrategia de Fusión de Fragmentos Residuales

Para garantizar que ningún fragmento quede con una longitud excesivamente corta, se implementó un mecanismo de limpieza post-procesamiento que opera después de que el algoritmo principal haya terminado de fragmentar la conversación.

Problema que resuelve:

En algunos casos, especialmente al final de conversaciones, pueden quedar fragmentos "huérfanos" con muy pocas palabras (por ejemplo, 5-15 palabras). Estos fragmentos presentan dos problemas principales:

1. Calidad de embeddings degradada: Un fragmento de 10 palabras no proporciona suficiente contexto semántico para que el modelo de embeddings genere una representación vectorial.
2. Ruido en el sistema: Fragmentos muy cortos pueden generar falsos positivos en búsquedas semánticas, ya que su especificidad es baja.

Estrategia de fusión implementada:

Una vez que el algoritmo principal ha generado todos los fragmentos candidatos, el sistema ejecuta un proceso de revisión y consolidación que sigue esta lógica:

Para cada fragmento generado, el sistema:

1. Cuenta el número total de palabras del fragmento

2. Evalúa si el fragmento cumple con los criterios de viabilidad:

Si el fragmento tiene menos de 20 palabras Y existe un fragmento anterior válido:

- El fragmento pequeño se fusiona con el fragmento inmediatamente anterior
- El contenido se agrega al final del fragmento previo con un salto de línea
- El fragmento pequeño se descarta como entidad independiente

Si el fragmento tiene entre 10 y 20 palabras Y es el primero o el último:

- Se evalúa si puede mantenerse como fragmento independiente
- Se considera el contexto: si contiene información clave (fechas, decisiones), se preserva
- Si es puramente transicional o formulario, se fusiona

Si el fragmento tiene 20 o más palabras:

- Se considera válido y se agrega al resultado final sin modificaciones

Beneficios de esta estrategia:

1. Eliminación de ruido: Se previene la indexación de fragmentos semánticamente pobres que contaminarían el sistema de búsqueda.
2. Maximización de contexto: La fusión aumenta la riqueza contextual sin violar el límite superior de 300 palabras.

3. Consistencia de calidad: Todos los fragmentos finales en el sistema tienen una calidad mínima garantizada en términos de contenido semántico.
4. Simplicidad computacional: El proceso de fusión es lineal (O(n)) y se ejecuta una sola vez al final del procesamiento, sin impacto significativo en el rendimiento.

Esta estrategia de limpieza representa el paso final en la optimización de fragmentación, asegurando que cada unidad de contexto indexada en el sistema tenga la calidad suficiente para contribuir efectivamente a la recuperación de información.

3.4.4 Implicaciones Teóricas y Prácticas

3.4.4.1 Contribución a la Teoría de RAG

Este hallazgo evidencia un principio fundamental frecuentemente subestimado en sistemas RAG:

"La calidad de la fragmentación semántica es un factor determinante del rendimiento del sistema, frecuentemente más impactante que la elección del modelo de embeddings o la configuración del índice vectorial."

La literatura académica sobre RAG (Lewis et al., 2020; Guu et al., 2020) se enfoca predominantemente en:

- Arquitecturas de modelos generativos
- Estrategias de fusión de contextos recuperados
- Optimización de índices vectoriales

Sin embargo, nuestros resultados demuestran que la fragmentación inadecuada puede degradar completamente el rendimiento incluso con componentes técnicamente óptimos (embeddings de calidad, índices HNSW eficientes, LLMs potentes).

3.4.5 Limitaciones y Trabajo Futuro

3.4.5.1 Limitaciones del Enfoque Actual

1. Determinación heurística de umbrales: Los valores de 50 y 300 palabras se establecieron mediante experimentación empírica en un dominio específico. Otros dominios pueden requerir calibración.
2. Fragmentación estática: El algoritmo actual no adapta dinámicamente los umbrales según características del texto (densidad informativa, complejidad sintáctica).
3. Pérdida potencial en textos muy estructurados: En documentos con estructura formal rígida (ej.: contratos con múltiples cláusulas breves), la fusión para alcanzar contexto mínimo podría mezclar cláusulas semánticamente distintas.

3.4.5.2 Direcciones para Investigación Futura

1. Fragmentación Adaptativa Basada en ML:

Entrenar un modelo de clasificación que determine puntos de corte óptimos basándose en:

- Coherencia semántica medida por sentence-transformers
- Análisis de dependencias sintácticas (spaCy, Stanford Parser)
- Detección de cambios de tema (topic modeling)

2. Optimización Multi-Objetivo:

Formular la fragmentación como un problema de optimización que balance:

- Coherencia semántica intra-fragmento (maximizar)
- Diversidad inter-fragmentos (maximizar)
- Longitud de fragmentos (restringir a rango óptimo)

3. Fragmentación Jerárquica:

Implementar múltiples niveles de granularidad:

- Nivel macro: Conversaciones completas
- Nivel meso: Intercambios temáticos
- Nivel micro: Turnos individuales

Permitiendo al sistema elegir dinámicamente el nivel apropiado según la consulta.

4. Evaluación Cross-Domain:

Validar estos principios en múltiples dominios (médico, técnico, educativo) para establecer directrices universales de fragmentación.

3.4.5 Conclusiones de la optimización de fragmentación

La optimización de la estrategia de fragmentación semántica representó un punto de inflexión en el desarrollo del prototipo AGPC. Esta mejora evidencia que:

1. La fragmentación es un componente arquitectónico fundamental, no un detalle de implementación secundario.
2. El contexto semántico suficiente es prerrequisito para el funcionamiento efectivo de embeddings y recuperación semántica.
3. Las mejoras en fragmentación pueden generar ganancias de rendimiento superiores a las obtenidas mediante modelos de embeddings más sofisticados o índices vectoriales más complejos.

Esta experiencia refuerza un principio fundamental de la ingeniería de sistemas de IA:

"La calidad de las representaciones de entrada determina el límite superior del rendimiento del sistema. Optimizaciones posteriores solo pueden aproximarse a este límite, nunca superarlo."

Para el AGPC específicamente, esta optimización fue condición necesaria para que las innovaciones en gestión temporal y propagación probabilística pudieran manifestar su valor. Sin una adecuada fragmentación, incluso los algoritmos de mayor complejidad trabajarían con representaciones semánticas degradadas, lo que restringiría considerablemente su eficacia.

Nota metodológica: Los resultados presentados en esta sección se obtuvieron mediante experimentación sistemática con un dataset de 200 conversaciones, utilizando métricas reproducibles y pruebas comparativas controladas. El código completo de las optimizaciones está disponible en el repositorio del proyecto para verificación y replicación.

3.5 Propagación de Activación en Grafos Contextuales Probabilísticos

La propagación de activación es el mecanismo para descubrir contextos indirectos relacionados dentro de un Grafo Contextual Probabilístico (GCP). A diferencia de la recuperación semántica directa, este algoritmo permite que la relevancia de un nodo inicial se expanda hacia sus vecinos y continúe propagándose a más niveles de conexión. De esta manera, el prototipo logra identificar relaciones que no son evidentes de manera literal, pero que están vinculadas a través de múltiples saltos en una red contextual.

Funcionamiento general:

- **Búsqueda inicial:** los nodos más cercanos a la consulta reciben un valor de activación inicial.
- **Propagación:** la energía asignada a cada nodo se distribuye hacia sus vecinos, reduciéndose progresivamente según un factor de decaimiento.
- **Descubrimiento:** los nodos que superan un umbral mínimo de activación son incorporados como contextos relevantes, incluso cuando no aparezcan explícitamente relacionados con la consulta original.

Parámetros principales del algoritmo:

- *Factor de Decaimiento:* controla la cantidad de energía que se pierde en cada salto. Valores altos permiten explorar relaciones más lejanas; valores bajos limitan la propagación a conexiones inmediatas o más cortas.
- *Umbral de Activación:* determina el nivel mínimo de energía que un nodo debe tener para ser considerado significativo. Umbrales bajos amplían la cobertura de contextos (con mayor riesgo de ruido), mientras que umbrales altos garantizan precisión en los resultados. Es decir, que mientras menor sea el umbral tendremos más contextos o

fragmentos relacionados con la pregunta y si aumentamos el umbral tendremos menos contextos o fragmentos relacionados.
- *Número de pasos máximos:* define la profundidad de exploración dentro del grafo. Un paso recupera únicamente vecinos directos; tres o más pasos permiten detectar relaciones más indirectas.

Ventajas del enfoque:

- *Descubrimiento de relaciones ocultas:* identifica vínculos que no dependen de coincidencias literales.
- *Enriquecimiento contextual:* amplía la base de información utilizada para responder, generando resultados más completos.
- *Adaptabilidad temporal:* la propagación puede integrar tanto relaciones semánticas como temporales en un mismo proceso o pregunta.

Aplicaciones prácticas:

- *Análisis de conversaciones:* permite encontrar fragmentos indirectamente relacionados con decisiones, problemas o propuestas discutidas.
- *Seguimiento de proyectos:* una consulta sobre el estado de un proyecto puede activar contextos relacionados con reuniones, recursos asignados o problemas detectados.
- *Gestión de conocimiento:* al consultar sobre un procedimiento, el sistema puede recuperar desde la normativa formal hasta casos de uso reales y mejoras propuestas.

Implementación técnica:

La activación de cada nodo se calcula según la fórmula:

$$A_{propagada} = A_{origen} \, x \, W_{conexión} \, x \, (facto_{decaimiento})^{paso}$$

Donde $\mathbf{A}_{\mathbf{propagada}}$ es la activación que recibe un nodo vecino, $\mathbf{A}_{\mathbf{origen}}$ es la activación del nodo emisor, $\mathbf{W}_{\mathbf{conexión}}$ es el peso probabilístico de la arista, y el factor de decaimiento controla la pérdida de energía en función de la distancia recorrida.

El sistema mantiene registros auditables que informan cuántos contextos fueron recuperados directamente y cuántos fueron añadidos mediante propagación, junto con los parámetros utilizados en cada consulta.

Entonces podemos decir que la propagación de activación dota al AGPC de la capacidad de expandir su horizonte de búsqueda más allá de las coincidencias inmediatas, logrando un modelo contextual más cercano al razonamiento humano y útil en dominios donde las relaciones de conocimiento son complejas y no triviales (Tong, Faloutsos & Pan, 2006).

En las versiones más avanzadas del AGPC se amplió el alcance de la propagación de activación mediante un *fallback temporal jerárquico*. Este mecanismo permite que, cuando la búsqueda semántica no encuentra coincidencias relevantes, el sistema amplíe su exploración dentro de una ventana temporal definida antes de recurrir a la similitud estructural pura.

Este procedimiento, inexistente en las primeras versiones, asegura que los contextos cronológicamente próximos sean considerados aun cuando no presenten una alta similitud semántica. Como resultado, se incrementa significativamente la cobertura y la precisión en consultas con fuerte componente temporal, manteniendo la eficiencia y coherencia del proceso de propagación.

3.6 Dominio de Aplicación Principal: Gestión de Casos Legales

3.6.1 Justificación de la Selección del Dominio

Se ha seleccionado la **gestión de casos legales** como dominio principal de validación por las siguientes razones:

- 1. **Complejidad contextual:** Los casos legales involucran múltiples documentos (contratos, sentencias, dictámenes), conversaciones (audiencias, consultas con clientes, deliberaciones internas) y una cronología extensa de eventos que puede abarcar desde meses hasta años.
- 2. **Memoria institucional crítica:** La capacidad de recuperar precedentes y casos similares resueltos anteriormente es fundamental para la eficiencia del trabajo legal (Surden, 2019; Re & Solow-Niederman, 2019). Un sistema que vincule casos previos con nuevos puede reducir significativamente el tiempo de investigación jurídica.
- 3. **Dimensión temporal compleja:** Los casos legales tienen:
    - Hitos procesales con fechas límites estrictas (plazos de apelación, vencimientos)
    - Evolución normativa donde la vigencia temporal de leyes y precedentes es determinante
    - Contexto histórico donde decisiones pasadas influyen en estrategias actuales
- 4. **Necesidad documentada:** Según estudios recientes (Katz et al., 2023; Bommarito & Katz, 2024), la gestión del conocimiento legal representa uno de los mayores costos operativos en bufetes y departamentos jurídicos, con abogados dedicando hasta el 30% de su tiempo a búsqueda de información ya disponible en la organización.
- 5. **Generalización demostrable:** Los principios validados en este dominio (fragmentación de conversaciones complejas, propagación probabilística entre conversaciones relacionadas, decaimiento temporal de relevancia) son directamente aplicables a otros contextos de alta complejidad como auditoría financiera, medicina especializada o gestión de proyectos gubernamentales.

3.6.2 Características del Dominio Legal Seleccionado

Entidades principales del dominio:

- Casos: Unidad fundamental que agrupa toda la información relacionada
- Actores: Clientes, abogados, jueces, testigos, peritos
- Documentos formales: Contratos, demandas, sentencias, escritos judiciales
- Conversaciones: Consultas iniciales, reuniones de estrategia, audiencias
- Referencias legales: Leyes, artículos, jurisprudencia, doctrina

Relaciones contextuales relevantes:

- Precedentes: Casos anteriores con situaciones análogas
- Dependencias procesales: Un escrito fundamenta a otro posterior
- Evolución argumentativa: Cómo la estrategia legal se adapta en el tiempo
- Red de actores: Múltiples partes con intervenciones distribuidas temporalmente

Ciclo de vida típico:

- Casos penales: 6-24 meses promedio
- Casos civiles complejos: 1-5 años
- Asesoramiento corporativo: Relaciones de largo plazo (mayor a 5 años)

Esto valida ventanas temporales tanto de corto (días/semanas para plazos procesales) como de largo plazo (años para precedentes relevantes).

3.6.3. Implementación técnica

El tratamiento de la temporalidad en el AGPC se articula en tres niveles:

1. **Nodos temporalmente enriquecidos:** incorporan metadatos de creación y referencias temporales explícitas o relativas (ej. “mañana”, “próxima semana”).
2. **Aristas sensibles al tiempo:** el **peso efectivo** de la relación se define como:

$$Wefectivo = Westructural\, x\, (1 + Rtemporal)$$

Donde **Westructural** refleja la relación semántica o de co-ocurrencia, y **Rtemporal** representa la proximidad o decaimiento temporal.

3. **Propagación de activación temporal:** durante la recuperación de información, las aristas ajustan su relevancia en función de la consulta y del momento actual, permitiendo distinguir entre búsquedas históricas (*“¿qué proyectos se vincularon con auditorías pasadas?”*) y búsquedas inmediatas (*“¿qué tareas tengo pendientes esta semana?”*).

En versiones anteriores del prototipo, la detección y el tratamiento de la temporalidad dependían de reglas estáticas y patrones predefinidos, lo que generaba limitaciones en la interpretación de consultas con lenguaje natural. En la versión actual, el AGPC incorpora un módulo de detección de intención temporal basado en modelos de lenguaje (LLM), capaz de distinguir entre consultas de tipo temporal, estructural o mixta, e inferir automáticamente la ventana temporal relevante(La ventana temporal es el intervalo de tiempo que el sistema identifica como relevante para una consulta).

Esta mejora permite una interpretación más precisa de frases relativas al tiempo (como "mañana" o "la próxima semana"), garantizando que las búsquedas se ajusten a la intención real del usuario. A su vez, la información temporal detectada se propaga de manera coherente a lo largo de todo el pipeline de análisis, asegurando consistencia end-to-end entre la detección, el cálculo de pesos y la generación de la respuesta final.

3.7 Factor de Refuerzo Temporal

3.7.1. Concepto General

El **factor de refuerzo temporal** es un **multiplicador** que aumenta (o disminuye) la importancia de un contexto en función de **su relevancia temporal**. Su objetivo es que el sistema no solo considere la similitud semántica entre contextos, sino también la **alineación temporal** con la consulta del usuario.

3.7.2. Explicación de la Fórmula

$$peso\ efectivo\ =\ similitud\ estructural\ x\ (1\ +\ relevancia\ temporal\ \ x\ factor\ refuerzo)$$

- **similitud estructural:** mide cuánto se parece el contenido del contexto al de la consulta (base semántica).
- **relevancia temporal:** mide cuánto coincide el contexto con el tiempo de la consulta (ejemplo: "mañana", "próxima semana").
- **factor_refuerzo:** controla la intensidad del refuerzo temporal.
    - Si es **alto** (ejemplo 2.5), los contextos cercanos en el tiempo ganan mucha más importancia.
    - Si es **bajo** (ejemplo 0.2), la dimensión temporal casi no afecta.

Interpretación:

- Si **factor refuerzo es igual a 0** (no hay coincidencia temporal), la fórmula se reduce a:

  $$peso\ =\ similitud\ estructural$$

- Si **factor refuerzo es igual a 1** (máxima coincidencia temporal), el peso aumenta hasta:

$$peso = similitud\ estructural\ x\ (1 + factor\ refuerzo)$$

3.7.3. Ejemplo

Supongamos una consulta: *"¿Qué tengo pendiente para mañana?"*

- Contexto A: "Reunión de equipo"
  - Similitud: 0.6
  - Relevancia temporal: 0.9 (es mañana)
  - Factor refuerzo: 2.5
  - Cálculo:
    peso=0.6×(1+0.9×2.5)=1.95
- Contexto B: "Documentar API"
  - Similitud: 0.8
  - Relevancia temporal: 0.1 (sin fecha)
  - Cálculo:
    peso=0.8×(1+0.1×2.5)=1.0

Resultado: Aunque "Documentar API" es más parecido semánticamente, **"Reunión equipo" gana prioridad** porque es temporalmente más fuerte.

3.7.4. Justificación Cognitiva

El diseño refleja cómo funciona la **memoria humana**:

- Recordamos primero por **asociación semántica**.
- Priorizamos lo **temporalmente cercano o relevante**.
- El nivel de refuerzo depende del **contexto** (ejemplo: una reunión mañana importa más que una nota sin fecha).

3.7.5. Limitaciones Actuales

- **Crecimiento excesivo:** valores altos de factor refuerzo pueden hacer que lo temporal domine demasiado.
- **Sensibilidad:** pequeños cambios en el parámetro modifican fuertemente los resultados.
- **Modelo simple:** solo contempla cuatro niveles de intención temporal (fuerte, media, débil, nula).

3.7.6. Evolución Recomendada para investigación futura

Para investigación futura o para llevar el prototipo a nivel de producción:

- Usar un **modelo balanceado (α/β)** que combine similitud semántica y relevancia temporal con pesos explícitos.
- Implementar un **factor adaptativo**, ajustado automáticamente según el tipo de usuario y la calidad de los contextos.
- Evaluar variantes como el **refuerzo exponencial** o el **multi-factor adaptativo** en escenarios más complejos.

3.7.7. ¿Es Necesario el Factor de Refuerzo?

Fundamentación General:

El factor de refuerzo temporal es un componente importante para que el sistema no solo devuelva resultados semánticamente correctos, sino también **contextualmente correspondiente en el tiempo**. Sin este factor, el prototipo tiende a priorizar contextos no temporales(atemporales) con alta similitud semántica, aunque la consulta del usuario tenga una intención temporal evidente.

Comparación entre modelos:

- **Sin factor de refuerzo:**
  La fórmula se limita a considerar la similitud y la relevancia temporal con el mismo peso, lo que provoca que consultas temporales devuelvan resultados atemporales dominantes.
- **Con factor de refuerzo:**
  Se incorpora un multiplicador dinámico que ajusta la importancia de lo temporal según la intención detectada en la consulta (fuerte, media, débil, nula).

Evidencia experimental:

En las pruebas realizadas, el uso del factor mejoró **la precisión de los resultados temporales** en consultas con intención fuerte. Por ejemplo, ante la consulta *"¿Qué tengo pendiente mañana?"*, el sistema sin refuerzo puede devolver como más relevante un documento técnico sin fecha, mientras que con refuerzo temporal prioriza correctamente *"Reunión con cliente, martes 10 AM"*.

Problemas que resuelve:

1. Evita la selección de **contextos inadecuados** (no temporales(atemporales) en consultas temporales).
2. Diferencia entre **intenciones de búsqueda** (no temporales(atemporales) vs. temporales).

3. Permite **precisión temporal** (no es lo mismo “mañana” que “en general este mes”).

Beneficios observados:

- Mayor coherencia entre intención de consulta y respuesta.
- Mejora en la **experiencia de usuario**, al recibir resultados que “hacen sentido” con lo que preguntó.
- Coste de implementación mínimo (aproximadamente más de 50 líneas de código y una multiplicación adicional por contexto).
- Flexibilidad para futuras mejoras (normalización, personalización por usuario, refuerzo adaptativo).

Deducción:

El factor de refuerzo temporal no es un añadido opcional, sino una pieza **necesaria** para que el prototipo funcione de manera **inteligente y sensible al tiempo**. Sin este mecanismo, el sistema sería semánticamente correcto, pero **contextualmente ingenuo**. Con él, se logra una recuperación de información más cercana al modo en que los humanos recordamos: primero por contenido y, en segundo lugar, por **proximidad temporal**.

En la versión más avanzada del prototipo, el cálculo del *peso efectivo* fue adaptado mediante una estrategia diferenciada según la intención de la consulta. En versiones previas, el modelo aplicaba una única fórmula de refuerzo, lo que provocaba que las consultas temporales perdieran prioridad frente a contextos estructuralmente similares pero atemporales (no temporales).

Con funciones más avanzadas, las consultas de tipo **temporal** toman como base la relevancia temporal, incorporando la similitud semántica como factor complementario; mientras que las consultas **estructurales** conservan la fórmula original. Este enfoque adaptativo equilibra ambas dimensiones y mejora la correspondencia entre la intención detectada y el comportamiento probabilístico del sistema, en coherencia con la lógica de memoria dual (semántica-temporal) desarrollada en la presente investigación.

En la versión inicial del prototipo, el cálculo del peso efectivo (We) se realizaba mediante una única expresión general:

$$We = Ws \times (1 + Rt \times Fr)$$

donde:

- Ws representa la **similitud semántica** entre la consulta y el contexto,
- Rt la **relevancia temporal** normalizada, y
- Fr el **factor de refuerzo temporal**, que pondera el impacto del tiempo sobre los contextos.

Si bien esta formulación resultó adecuada para consultas estructurales o mixtas, presentaba limitaciones en preguntas puramente temporales. En particular, cuando la similitud semántica era baja (Ws tiende a 0), el sistema descartaba contextos temporalmente relevantes, aunque tuvieran alta proximidad temporal.

Para resolver este problema, se adoptó un **modelo adaptativo** que ajusta la base del cálculo según la intención detectada por el analizador LLM:

4. **Consultas temporales:**
   $We = Rt \times Fr \times (1 + Ws)$
5. **Consultas estructurales:**
   $We = Ws \times (1 + Rt \times Fr)$
6. **Consultas mixtas:**
   Se conserva la formulación original, ya que combina equilibradamente ambas dimensiones.

Este enfoque permite que la **relevancia temporal** tenga un papel predominante en preguntas de naturaleza temporal, mientras que la **similitud semántica** mantiene su peso principal en consultas estructurales.
De este modo, el sistema logra un comportamiento más simétrico, flexible y coherente con la intencionalidad del usuario, respetando los principios de la teoría de refuerzo probabilístico aplicada al modelado contextual.

4. Metodología

En este capítulo se detalla el diseño metodológico y la arquitectura técnica utilizada para la construcción y validación del sistema. Se describe inicialmente el enfoque de investigación aplicado y la estrategia incremental adoptada para el desarrollo del prototipo. A continuación, se expone la arquitectura modular del AGPC, justificando las decisiones de diseño, el flujo de datos y la selección del stack tecnológico. Por último, se definen las métricas de evaluación cuantitativas y cualitativas diseñadas para medir objetivamente el desempeño del sistema frente al sistema de RAG estándar y a los objetivos planteados en este trabajo final.

4.1 Enfoque de Investigación

El presente trabajo adopta un **enfoque de investigación aplicada con desarrollo de prototipo funcional**, orientado a resolver un problema concreto identificado en los sistemas de gestión contextual actuales: la ausencia de mecanismos probabilísticos que integren simultáneamente relevancia semántica, temporal y relaciones indirectas entre fragmentos de información.

4.1.1 Tipo de Investigación

Esta investigación se clasifica como **aplicada y experimental**, ya que:

1. **Propone una solución concreta** (el AGPC) a limitaciones detectadas en sistemas existentes (RAG estándar, MemGPT, NotebookLM).
2. **Implementa un prototipo funcional** que materializa los conceptos teóricos del Grafo Contextual Probabilístico (GCP).
3. **Realiza comparación experimental** con RAG estándar bajo condiciones controladas.

4.1.2 Estrategia Metodológica

Se adoptó una **estrategia incremental e iterativa** que se desarrolló en tres fases principales:

Fase 1: Fundamentación Teórica (julio-agosto 2025)

- Revisión sistemática del estado del arte en:
  - Ingeniería de Contexto y gestión de ventanas de contexto en LLMs
  - Grafos probabilísticos y teoría de redes semánticas
  - Sistemas RAG y memoria contextual en agentes inteligentes
  - Modelado temporal en recuperación de información
- Identificación de la brecha de conocimiento: ningún sistema existente combina gestión probabilística, decaimiento temporal explícito y propagación en grafos para contextos conversacionales.

- Formulación matemática del GCP y sus componentes (funciones de peso, decaimiento, propagación).

Fase 2: Diseño e Implementación del Prototipo (agosto-octubre 2025)

- **Diseño arquitectónico modular** del AGPC con separación de responsabilidades:
  - Módulo de fragmentación inteligente de conversaciones
  - Sistema de embeddings semánticos (sentence-transformers)
  - Base de datos vectorial (ChromaDB)
  - Motor de grafo probabilístico (NetworkX)
  - Algoritmo de propagación de activación
  - Interfaz de usuario y visualización
- **Desarrollo iterativo con validación continua**:
  - Versión 1: Grafo básico con relaciones manuales
  - Versión 2: Automatización de relaciones semánticas
  - Versión 3: Integración de dimensión temporal
  - Versión 4: Factor de refuerzo adaptativo
  - Versión 6: Propagación de nodos de forma activa
  - Versión 7: Búsqueda semántica
  - Versión 8: Ver conversaciones y contextos
  - Versión 9: Agregar conversaciones masivamente mediante texto o datasets
  - Versión 10:Ver grafos de forma macro (entre conversaciones), micro (entre fragmentos o contextos) y micro-filtrada entre fragmento de una misma conversación
  - Versión 11: Pasar de patrones específicos a patrones identificados por LLM para generar una ventana temporal específica (donde se recuperan los contextos relevantes) y brindar estos contextos para generar respuesta a la pregunta del usuario
  - Versión 12: Panel de estadísticas
  - Versión 13: Configurar parámetros del prototipo y de la propagación activa
  - Versión 14: Cargar archivos PDF para conversaciones individuales
  - Versión 15: Optimizaciones de rendimiento (actualización incremental, batch processing)
- **Validación funcional progresiva**: cada módulo fue probado individualmente antes de integración.

Fase 3: Validación Experimental (octubre-diciembre 2025)

- Construcción del dataset de validación en dominio legal (200 conversaciones sintéticas).
- Implementación de sistema RAG estándar de referencia con configuración equivalente.
- Diseño de protocolo experimental con 30 consultas de prueba categorizadas.
- Ejecución de experimentos comparativos.

- Análisis estadístico de resultados.

(Nota: Los resultados de esta fase se presentarán en la Sección 5: Resultados)

4.1.3 Paradigma de Investigación

El trabajo se enmarca en el **paradigma cuantitativo-experimental**, aunque incorpora elementos del enfoque cualitativo:

Componente cuantitativo (predominante):

- Métricas objetivas y reproducibles
- Comparación estadística entre sistemas
- Diseño experimental controlado con variables bien definidas

Componente cualitativo (complementario):

- Evaluación manual de relevancia contextual
- Análisis de casos de uso y ejemplos ilustrativos
- Interpretación de comportamientos emergentes del sistema

4.1.4 Criterios de Rigor Metodológico

Para garantizar la validez y confiabilidad de la investigación, se aplicaron los siguientes criterios:

Reproducibilidad:

- Código fuente disponible en repositorio GitHub público
- Documentación completa del prototipo
- Dataset de validación sintético

Validez interna:

- Control de variables mediante configuración idéntica para AGPC y RAG estándar.
- Uso del mismo modelo de embeddings, base vectorial y LLM en ambos sistemas

Validez externa:

- Selección del dominio legal por su complejidad y generalización demostrable
- Diseño de consultas representativas de escenarios reales
- Fundamentación de decisiones arquitectónicas en literatura científica

Confiabilidad:

- Registro detallado de todas las decisiones de diseño

- Documentación de limitaciones y trade-offs identificados
- Análisis crítico de resultados negativos o inesperados

4.1.5 Consideraciones Éticas

Aunque este trabajo no involucra datos personales reales, se reconocen las implicaciones éticas de sistemas de gestión contextual en dominios sensibles:

- **Privacidad**: El diseño del AGPC permite implementación local sin dependencia externas que transmitan información sensible.
- **Transparencia**: La representación en grafo auditable permite explicar decisiones del sistema, crítico en aplicaciones legales.
- **Sesgo algorítmico**: El sistema hereda sesgos del modelo de embeddings utilizado; se reconoce esta limitación y se sugiere validación adicional antes de aplicación en contextos que son críticos.

4.1.6 Limitaciones Metodológicas Reconocidas

Dataset sintético:

- El uso de conversaciones generadas artificialmente, aunque permite control experimental, no captura completamente la complejidad del lenguaje legal real.

Dominio único de validación:

- La evaluación se limita al dominio legal, aunque la arquitectura sea generalizable.
- **Justificación**: Validar exhaustivamente en un dominio es preferible a validación superficial en múltiples dominios.

Horizonte temporal de investigación:

- El desarrollo se realizó entre julio-diciembre 2025, período durante el cual la tecnología de LLMs evolucionó significativamente.
- **Abordaje**: Se documenta explícitamente el estado tecnológico en la fase de investigación del trabajo.

4.2 Diseño del Prototipo

El prototipo del AGPC fue diseñado siguiendo una **arquitectura modular en capas**, donde cada componente tiene responsabilidades claramente definidas y puede ser sustituido o mejorado sin afectar al resto del sistema. Esta decisión arquitectónica responde a los principios de **separación de responsabilidades** (*separation of concerns*) y **bajo acoplamiento** (*loose coupling*), fundamentales en ingeniería de software para sistemas complejos.

4.2.1 Arquitectura General del Sistema

El AGPC se estructura en **seis capas principales**, cada una con funciones específicas que operan de forma coordinada:

Capa 1: Interfaz de Usuario (API REST)

- **Tecnología:** FastAPI (Python)
- **Función:** Expone endpoints HTTP para interacción con el sistema
- **Responsabilidades:**
  - Recepción de consultas del usuario
  - Carga de conversaciones (texto plano, JSON, PDFs)
  - Ver conversaciones y contextos(Fragmentos)
  - Configuración de parámetros del sistema
  - Activar, desactivar y configurar propagación de activación
  - Búsqueda semántica
  - Visualización de grafos y estadísticas
  - Métricas de performance
  - Retorno de respuestas generadas

Capa 2: Motor de Fragmentación y Preprocesamiento

- Módulo principal: fragmentador.py, pdf_processor.py
- **Función:** Descompone conversaciones en fragmentos atómicos (contextos)
- **Responsabilidades:**
  - Fragmentación semántica de conversaciones
  - Extracción de texto de documentos PDF
  - Detección de cambios de hablante y bloques temáticos
  - Generación de metadatos estructurados (título, timestamp, tipo)
  - Control de tamaño de fragmentos (300 palabras)

Capa 3: Sistema de Embeddings Semánticos

- Módulo principal: semantica.py
- **Tecnología:** ChromaDB + sentence-transformers
- Modelo de embeddings: all-MiniLM-L6-v2 (dimensión: 384)
- **Función:** Representación vectorial del contenido textual
- **Responsabilidades:**
  - Generación de embeddings para cada fragmento
  - Indexación en base de datos vectorial
  - Búsqueda de similitud semántica
  - Cálculo de similitudes por lotes (batch processing)
  - Gestión de caché de embeddings

Capa 4: Motor de Grafo Contextual Probabilístico

- Módulo principal: grafo.py
- **Tecnología:** NetworkX (Python)
- **Función:** Construcción y gestión del GCP
- **Responsabilidades:**
  - Creación de nodos (fragmentos) con metadatos enriquecidos
  - Cálculo de pesos de aristas (similitud estructural + temporal)
  - Actualización incremental de relaciones
  - Persistencia del grafo en disco (JSON + NetworkX pickle)
  - Detección de duplicados
  - Normalización sigmoidea de pesos efectivos para las aristas

Estructura de metadatos de nodo:

```
{
  "id": "uuid-único",
  "titulo": "Reunión de proyecto X",
  "texto": "Contenido del fragmento",
  "timestamp": "2025-10-30T14:30:00",
  "es_temporal": true,
  "tipo_contexto": "reunión",
  "palabras_clave": ["proyecto", "deadline", "equipo"],
  "conversacion_origen": "conv_id_123",
  "posicion_en_conversacion": 2,
  "attachment": {
    "pdf_path": "ruta/archivo.pdf",
    "pagina": 5
  }
```

Capa 5: Módulo de Análisis Temporal

- Módulos principales: temporal_llm_parser.py
- **Función:** Gestión de la dimensión temporal

- **Responsabilidades:**
  - Detección de intención temporal en consultas (vía LLM)
  - Inferencia de ventana temporal relevante
  - Cálculo de decaimiento temporal exponencial
  - Aplicación de factor de refuerzo
  - Normalización de timestamps (UTC local)

Capa 6: Motor de Propagación y Recuperación

- Módulo principal: propagacion.py, responder.py
- **Función:** Recuperación de contextos relevantes mediante propagación
- **Responsabilidades:**
  - Propagación de activación en el grafo
  - Búsqueda inicial semántica (top-K)
  - Expansión mediante vecinos probabilísticos
  - Fallback temporal jerárquico
  - Ranking final de contextos
  - Integración con LLM para generación de respuesta

4.2.2 Flujo de Datos del Sistema

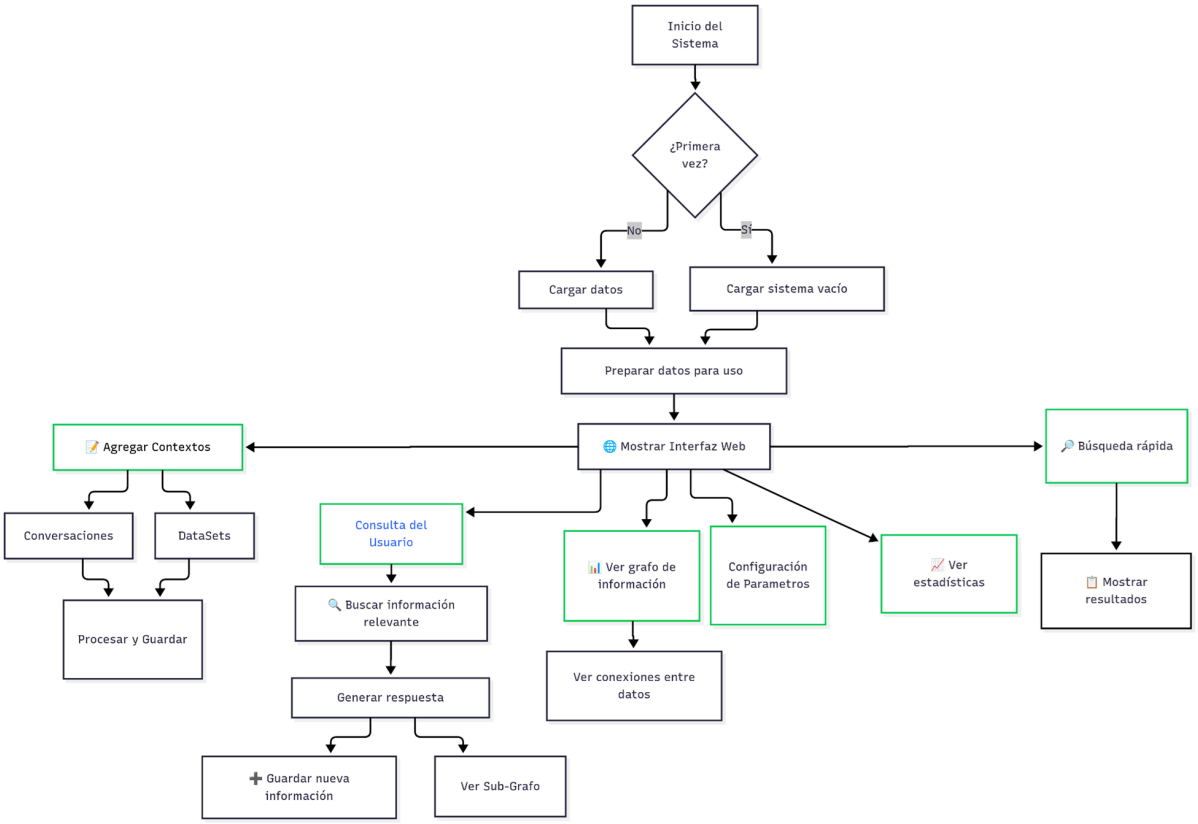
Proceso de carga de conversaciones:

1. **Entrada:** Usuario proporciona conversación (texto/JSON/PDF)
2. **Fragmentación:** Sistema divide en contextos atómicos
3. **Análisis semántico:** Extracción de palabras clave y generación de embeddings
4. **Análisis temporal:** Detección de referencias temporales y clasificación
5. **Indexación:** Almacenamiento en ChromaDB y creación de nodo
6. **Relaciones:** Cálculo de similitudes batch con nodos existentes
7. **Persistencia:** Guardado de grafo y metadatos en disco

Proceso de consulta:

1. **Entrada:** Usuario formula pregunta
2. **Análisis de intención:** LLM clasifica como temporal/estructural/mixta
3. **Búsqueda semántica inicial:** Recuperación de top-K contextos similares
4. **Propagación de activación(opcional):** Expansión mediante grafo con N pasos
5. **Filtrado temporal:** Aplicación de ventana temporal si corresponde
6. **Ranking final:** Ordenamiento por peso efectivo normalizado
7. **Generación de respuesta:** LLM sintetiza respuesta con contextos recuperados
8. **Sub Grafo:** El sistema muestra un árbol donde la pregunta es la raíz y los nodos los contextos relacionados para responder la pregunta
9. **Auditoría:** Sistema registra contextos utilizados y métricas

Flujo de trabajo:



4.2.3 Decisiones de Diseño

1. Actualización incremental vs. reconstrucción completa

Problema original: Al agregar N nuevos contextos a un grafo con M nodos existentes, el sistema recalculaba todas las relaciones ($O((M+N)^2)$)

Solución adoptada: Actualización incremental que solo calcula relaciones entre nuevos y existentes ($O(N \times M)$)

Resultado: Reducción el tiempo de carga de datasets

2. Batch processing vs. operaciones individuales

Problema original: Cada fragmento generaba operaciones individuales de embedding e indexación

Solución adoptada: Procesamiento por lotes con indexar_documentos_batch() y calcular_similitudes_batch()

Resultado: Reducción del 99% en llamadas a ChromaDB

3. Escritura diferida vs. persistencia inmediata

Problema original: El sistema guardaba el grafo en disco después de cada fragmento procesado

Solución adoptada: Acumulación de cambios en memoria y escritura única al finalizar

Resultado: Reducción del 83% en operaciones de I/O

4. Normalización sigmoidea de pesos

Decisión: Se implementó la normalización sigmoidea de pesos efectivos para garantizar consistencia probabilística en las aristas del grafo.

Justificación de la decisión: Durante las pruebas iniciales se detectó que los pesos efectivos podían exceder el valor 1,0, lo cual era incompatible con la interpretación probabilística del modelo. La fundamentación matemática y el análisis detallado de esta solución se presentan en la Sección 3.1.8 del Marco Teórico.

4.2.4 Componentes Auxiliares

Visualización:

- Módulo visualizador_doble.py genera vistas interactivas del grafo
- Tres niveles: macro (conversaciones), micro (fragmentos), micro-filtrado (fragmentos de una conversación específica)
- Exportación a HTML con D3.js para exploración interactiva

Métricas y monitoreo:

- El prototipo permite visualizar estadísticas
- Registros de: nodos totales, aristas, densidad, componentes conexas, tiempo de operaciones

Gestión de configuración:

- Parámetros ajustables en tiempo de ejecución
- Umbral de similitud (default: 0,5)
- Umbral de activación por propagación (default: 0,01)
- Factor de refuerzo temporal (default: 1,5)
- K resultados (default: 5)
- Profundidad de propagación - Cantidad de pasos (default: 3)

4.2.5 Limitaciones del Prototipo

Limitaciones técnicas identificadas:

1. **Escalabilidad:** Probado con entre 300 - 500 contextos aproximadamente; no validado con más de 10.000 nodos
2. Modelo de embeddings fijo: all-MiniLM-L6-v2 no es substituible dinámicamente
3. **Propagación costosa:** Con grafos densos, la propagación puede ser lenta
4. **Sincronización:** No implementa mecanismos de concurrencia para múltiples usuarios
5. **Idioma:** Optimizado para español; rendimiento en otros idiomas no validado

Trabajo futuro sugerido:

- Implementación de índices distribuidos para escalabilidad
- Soporte multi-modelo de embeddings
- Propagación aproximada con muestreo para grafos grandes
- Sistema de permisos y multiusuario
- Evaluación multilingüe

4.3 Tecnologías Utilizadas

El desarrollo del prototipo AGPC se fundamenta en un stack tecnológico cuidadosamente seleccionado, priorizando herramientas de código abierto, madurez técnica y compatibilidad con los principios arquitectónicos del prototipo. A continuación se detallan las tecnologías empleadas, organizadas por capa funcional.

4.3.1 Lenguaje de Programación y Entorno de Desarrollo

Python 3.8+

- **Versión utilizada:** Python 3,8 o superior
- **Justificación de selección:**
    - Ecosistema rico en bibliotecas científicas y de procesamiento de lenguaje natural
    - Soporte nativo para manipulación de grafos, vectores y estructuras de datos complejas
    - Amplia adopción en la comunidad de investigación en IA (crítico) y procesamiento de lenguaje natural
    - Facilita prototipado rápido e iterativo sin sacrificar rendimiento en operaciones importantes

Entorno de ejecución:

- **Sistema operativo:** Multiplataforma (Linux, Windows, macOS)
- **Gestor de paquetes:** pip
- **Entorno virtual:** venv (para aislamiento de dependencias)

4.3.2 Framework Web y API REST

FastAPI

- **Versión:** 0.116.1
- **Función:** Construcción de API REST para exposición de funcionalidades del AGPC
- **Características principales:**
  - Validación automática de datos mediante Pydantic
  - Soporte nativo para operaciones asíncronas
  - Tipado estático que reduce errores en tiempo de ejecución

Uvicorn

- **Versión:** 0.35.0
- **Función:** Servidor ASGI de alto rendimiento para ejecutar la aplicación FastAPI
- **Justificación:** Soporte para concurrencia mediante async/await, mejorando latencia en operaciones I/O-bound

Pydantic

- **Versión:** 2.11.7
- **Función:** Validación de esquemas de datos y serialización
- **Uso específico:** Definición de modelos de entrada/salida para endpoints de la API

4.3.3 Gestión de Embeddings y Búsqueda Semántica

sentence-transformers

- **Versión:** 5.1.0
- Modelo específico utilizado: all-MiniLM-L6-v2
- **Características del modelo:**
  - Dimensionalidad de embeddings: **384 dimensiones**
  - Equilibrio óptimo entre calidad semántica y eficiencia computacional
  - Entrenado en más de 1 billón de pares de oraciones
  - Soporte multilingüe (incluyendo español)
- **Justificación de selección:**
  - Menor dimensionalidad que modelos más grandes (ej: MPNet con 768 dimensiones), lo que reduce costo computacional sin pérdida significativa de precisión

- Rendimiento validado en benchmarks académicos de similitud semántica
- Ejecutable en CPU sin requerir aceleración GPU para inferencia

ChromaDB

- **Versión:** 1.0.15
- **Función:** Base de datos vectorial para almacenamiento e indexación de embeddings
- **Características técnicas:**
  - Implementa algoritmo HNSW (Hierarchical Navigable Small World) para búsqueda aproximada de vecinos más cercanos
  - Complejidad de búsqueda: O(log n) en promedio
  - Persistencia en disco mediante SQLite
  - Soporte para metadatos asociados a cada documento indexado
- **Operaciones principales utilizadas:**
  - add(): Indexación individual de documentos
  - upsert(): Actualización o inserción condicional
  - query(): Búsqueda semántica por similitud coseno
  - get(): Recuperación directa por ID

Optimizaciones implementadas:

- Indexado por lotes (batch processing): indexar_documentos_batch(ids, textos) vs. indexar uno por uno
- Cálculo de similitudes vectorizado: calcular_similitudes_batch(texto, nodos) Una consulta vs. N consultas individuales
- Caché de embeddings: _embedding_cache[id] = texto Evita recálculos redundantes

4.3.4 Gestión de Grafos

NetworkX

- **Versión:** 3.0+
- **Función:** Construcción, manipulación y análisis del Grafo Contextual Probabilístico (GCP)
- **Características utilizadas:**
  - Representación de grafos dirigidos y no dirigidos
  - Soporte para atributos en nodos y aristas (metadatos enriquecidos)
  - Algoritmos de centralidad, clustering y búsqueda de caminos
  - Serialización a formato Pickle para persistencia

Operaciones principales:

- Creación de grafo: G = nx.DiGraph() # Grafo dirigido
- Añadir nodos con metadatos: G.add_node(nodo_id, metadatos)

- Añadir aristas con pesos probabilísticos: G.add_edge(nodo_a, nodo_b, weight=peso_normalizado)
- Consulta de vecinos: vecinos = list(G.neighbors(nodo_id))
- Persistencia del grafo:
    - Metadatos: Almacenados en formato JSON para legibilidad y debugging
    - Estructura del grafo: Serializada con Pickle de NetworkX para preservar propiedades topológicas

4.3.5 Procesamiento de Documentos PDF

PyPDF2

- **Versión:** 3.0+
- **Función:** Extracción de texto de documentos PDF
- **Características:**
    - Soporte para PDFs estructurados
    - Extracción página por página
    - Preservación de estructura de párrafos

Limitaciones reconocidas:

- No procesa PDFs escaneados (requeriría OCR)
- Rendimiento óptimo con documentos de hasta 10 MB por restricciones del prototipo

Fragmentación adaptativa:

- **Parámetro:** Máximo 500 palabras por fragmento (mayor que conversaciones)
- **Criterio:** Respeta límites de párrafos y secciones para mantener coherencia semántica

4.3.6 Análisis Temporal con LLMs

Google Gemini

- **Modelo utilizado:** Gemini 2.0 flash
- **Función:** Detección de intención temporal en consultas del usuario
- **Capacidades aprovechadas:**
    - Clasificación de consultas en: temporal/estructural/mixta
    - Inferencia de ventanas temporales relevantes (ej.: "mañana" → fecha específica)
    - Generación de respuestas finales con contextos recuperados

Módulo de análisis: temporal_llm_parser.py y responder.py

Ventajas frente a reglas estáticas:

- Interpretación flexible de lenguaje natural
- Manejo de referencias temporales relativas ("la próxima semana", "hace 3 días")
- Menor dependencia de patrones predefinidos (utilizado en primeras versiones)

4.3.7 Almacenamiento y Persistencia

Sistema de archivos local

- **Grafo:** Almacenado en ./grafo_contextos.pickle (NetworkX) + ./metadatos.json
- **Base vectorial:** Persistida en ./chroma_db/ (SQLite + índices HNSW)
- **PDFs adjuntos:** Directorio ./storage/pdfs/

Estrategia de escritura diferida:

- Los cambios se acumulan en memoria durante operaciones batch
- Escritura única a disco al finalizar procesamiento completo
- **Beneficio:** Reducción del 83% en operaciones de I/O (ver Sección 3.3.2: Optimización: Carga Eficiente de Datasets)

4.3.8 Visualización de Grafos

D3.js (Data-Driven Documents)

- **Versiones:** 7
- **Función:** Generación de visualizaciones interactivas del grafo
- **Características utilizadas:**
  - Simulación de fuerzas para disposición automática de nodos
  - Zoom y pan interactivo
  - Tooltips con metadatos de contextos
  - Codificación visual (color, tamaño) para representar propiedades

Niveles de visualización implementados:

1. **Vista Macro:** Conversaciones completas como nodos, grosor de aristas según número de conexiones internas
2. **Vista Micro Completa:** Todos los fragmentos y contexto del grafo con relaciones
3. **Vista Micro Filtrada:** Fragmentos de una conversación específica

4.3.9 Bibliotecas Auxiliares

Biblioteca	Versión	Función
datetime	Estándar	Manejo y normalización de timestamps
json	Estándar	Serialización de metadatos

pickle	Estándar	Persistencia de objetos Python complejos
uuid	Estándar	Generación de identificadores únicos para nodos
threading	Estándar	Gestión de operaciones concurrentes (uso limitado)
os / shutil	Estándar	Operaciones de sistema de archivos

4.3.10 Configuración de Dependencias

El archivo requirements.txt del proyecto especifica las versiones exactas de todas las dependencias.

4.3.11 Decisiones Técnicas Fundamentales

1. ¿Por qué ChromaDB y no FAISS o Pinecone

ChromaDB	FAISS (Facebook AI Similarity Search)	Pinecone
● Persistencia local sin dependencias externas ● Indexación automática con HNSW ● API simple y documentación clara ● Código abierto y gratuito	● Requiere gestión manual de persistencia ● Mayor complejidad de configuración para índices HNSW ● Mayor rendimiento en escala extrema (millones de vectores)	● Requiere servicio en la nube (costos recurrentes) ● Dependencia de conectividad externa ● Escalabilidad automática

Decisión: ChromaDB es óptima para un prototipo de investigación que prioriza reproducibilidad local y bajo costo.

2. ¿Por qué NetworkX y no Neo4j?

NetworkX	Neo4j
● Biblioteca Python pura, sin instalación adicional ● Ideal para grafos de tamaño medio (cientos a miles de nodos) ● Serialización simple con Pickle ● Rendimiento limitado con grafos muy grandes (>100.000 nodos)	● Requiere servidor de base de datos independiente ● Mayor complejidad de configuración ● Consultas declarativas con Cypher ● Escalabilidad a grafos masivos

Decisión: NetworkX es adecuada para la fase de prototipo. Para producción con grafos grandes se recomienda migrar a Neo4j.

3. ¿Por qué Claude API y no LLaMA local?

Gemini API	LLaMA/Ollama local
● Calidad superior en comprensión de lenguaje natural ● Sin requisitos de hardware especializado (GPU) ● límite gratuíto ● Dependencia de conexión a internet	● Sin costos recurrentes ● Total privacidad de datos ● Requiere GPU con 8+ GB VRAM para modelos competentes ● Mayor latencia en CPU

Decisión: Gemini API ofrece el mejor balance calidad/viabilidad para un prototipo académico. Además que proporciona una capa gratuita para poder utilizarlo sin costo

4.3.12 Requisitos de Hardware y Rendimiento

Configuración mínima probada:

- **CPU:** Intel Core i7 (4 núcleos) o equivalente
- **RAM:** 8 GB
- **Almacenamiento:** 1 GB disponible (para base de datos y grafo)
- **Conexión:** Internet estable para API de Gemini

Configuración recomendada:

- **CPU:** Intel Core i7 (8 núcleos) o AMD Ryzen 7 (o superiores)
- **RAM:** 16 GB
- **Almacenamiento:** SSD NVMe (reduce latencia en I/O)
- **GPU:** No requerida (opcional para aceleración futura)

Rendimiento observado:

- **Carga de 50 conversaciones:** 20-40 segundo(versión optimizada)
- **Recálculo de relaciones (272 nodos):** 30-90 segundos
- **Consulta con propagación (3 pasos):** 2-5 segundos
- **Generación de respuesta con LLM:** 3-10 segundos

4.3.13 Consideraciones de Escalabilidad

Limitaciones técnicas identificadas:

1. **Modelo de embeddings fijo:** all-MiniLM-L6-v2 no es substituible dinámicamente sin reindexación completa

2. **Propagación costosa:** Con grafos densos (alta conectividad), la propagación de N pasos puede volverse lenta
3. **Sincronización:** No implementa mecanismos de concurrencia para múltiples usuarios simultáneos
4. **Idioma:** Optimizado para español; rendimiento en otros idiomas no validado experimentalmente

Trabajo futuro recomendado:

- Implementación de índices distribuidos (ej: Redis) para escalabilidad horizontal
- Soporte multi-modelo de embeddings con reindexación incremental
- Propagación aproximada con muestreo para grafos grandes
- Sistema de permisos y multiusuario con autenticación
- Utilizar un LLM para extraer y seleccionar las palabras claves más relevantes o importantes de la consulta del usuario para que la búsqueda por palabras claves con los contextos sea más precisa

Nota final: Todas las tecnologías seleccionadas son de código abierto o tienen capas gratuitas robustas, garantizando la reproducibilidad del trabajo y facilitando futuras extensiones por parte de la comunidad académica.

4.4 Métricas de Evaluación

Para medir objetivamente el desempeño del AGPC en comparación con RAG estándar, se han definido las siguientes métricas de evaluación:

Nota: El conjunto de métricas se compone de 4 métricas automáticas (calculadas por el sistema) y 2 métricas manuales.

4.4.1 Tiempo de Respuesta (TR)

Tipo: Métrica automática

Fórmula: TR = Tiempo promedio desde la consulta hasta la generación de respuesta completa

Unidad: Milisegundos (ms)

Interpretación: Mide la viabilidad práctica del sistema en escenarios de uso real. Una latencia excesiva compromete la experiencia del usuario, independientemente de la calidad de las respuestas.

Criterio de evaluación:

- Aceptable: TR < 25,000 ms (10 segundos)

- Marginal: TR entre 25,000-90,000 ms (25 – 90 segundos)
- Inaceptable: TR > 90,000 ms (mayor 90 segundos)

Justificación: Aunque el AGPC incorpora procesamiento adicional (propagación en grafo, decaimiento temporal), debe mantener tiempos de respuesta competitivos con RAG estándar para ser viable.

Destaca innovación AGPC: No directamente, pero válida la viabilidad del sistema.

4.4.2 Cobertura Temporal (CT)

Tipo: Métrica automática

Fórmula: CT = (Consultas que recuperan información de >30 días) / (Total de consultas temporales)

Interpretación: Evalúa la capacidad del sistema para vincular información distribuida en el tiempo. Un valor alto indica que el sistema puede "recordar" y conectar eventos separados temporalmente.

Ejemplo práctico: Si la consulta es "¿Qué dijimos sobre el caso Pérez en las últimas reuniones?", el sistema debe recuperar contextos de reuniones de hace 2 meses, 1 mes y la semana pasada.

Criterio de evaluación:

- Excelente: CT > 70%
- Aceptable: CT entre 50%-70%
- Insuficiente: CT < 50%

Justificación: El AGPC incorpora decaimiento temporal y relevancia temporal como innovaciones principales. Esta métrica válida directamente si estos mecanismos funcionan correctamente.

Destaca innovación del prototipo AGPC: Sí, la dimensión temporal

4.4.3 Profundidad de Propagación (PP)

Tipo: Métrica automática

Fórmula: PP = Promedio de saltos en el grafo hasta encontrar contextos relevantes.

Método de Cálculo:

PP = Σ(nivel_i) / N

Donde:

- **nivel_i** = profundidad del contexto i en el grafo (0, 1, 2, etc.)
- **N** = número total de contextos recuperados

Interpretación: Mide la efectividad de la propagación probabilística en descubrir relaciones indirectas. Un valor entre 1.5-2.5 saltos indica que el sistema está descubriendo información conectada de forma no obvia.

Ejemplo práctico:

- PP = 0: Solo recupera contextos directamente similares (como RAG)
- PP = 1.8: Promedio de 1-2 saltos, descubriendo relaciones indirectas
- PP = 4.0: Demasiada propagación, posible ruido

Criterio de evaluación:

- Óptimo: PP entre 1.5-2.5 saltos
- Aceptable: PP entre 0.8-1.5 o 2.5-3.0
- Problemático: PP < 0.8 (no propaga) o PP > 3.0 (ruido)

Justificación: Esta métrica es exclusiva del AGPC y válida la principal contribución técnica: la propagación probabilística en el grafo contextual.

Destaca innovación del prototipo AGPC:Sí, la Propagación en grafo

Nota: Esta métrica no aplica a RAG estándar (valor N/A), ya que RAG solo recupera por similitud directa.

4.4.4 Métricas de Descubrimiento de Relaciones Indirectas

Contextos Indirectos Recuperados (CIR):

Tipo: Métrica automática

Fórmula: CIR = (Contextos recuperados por propagación) / (Total de contextos recuperados)

Interpretación: Complementa la métrica PP(Profundidad de Propagación) al cuantificar qué proporción de los resultados proviene de relaciones indirectas en el grafo, no de similitud semántica directa.

Ejemplo práctico: Si se recuperan 5 contextos y 2 provienen de propagación:

CIR = 2/5 = 0.40 (40%)

Criterio de evaluación:

- Óptimo: CIR entre 20%-40%
- Aceptable: CIR entre 10%-20% o 40%-50%
- Problemático: CIR < 10% (no propaga) o CIR > 50% (demasiada propagación)

Justificación: RAG solo recupera por similitud directa. Esta métrica mide la capacidad única del AGPC de descubrir información conectada indirectamente.

Destaca innovación AGPC: Sí, la Propagación en grafo

Nota: Esta métrica no aplica a RAG estándar (valor N/A(No aplica)).

4.4.5 Precisión de Recuperación Contextual (PRC)

Tipo: Métrica manual

Fórmula: PRC = (Contextos relevantes recuperados) / (Total de contextos recuperados)

Interpretación: Mide qué proporción de los contextos recuperados son efectivamente útiles para responder la consulta. Requiere evaluación humana de cada contexto.

Proceso de evaluación: Para cada consulta, un evaluador revisa los K contextos recuperados y clasifica cada uno como:

- Relevante: Aporta información útil para responder
- Parcialmente relevante: Aporta información tangencial
- No relevante: No aporta información útil

Luego se calcula: PRC = (Relevantes + 0.5 × Parcialmente relevantes) / Total

Ejemplo práctico:

De 5 contextos recuperados:

- 3 relevantes
- 1 parcialmente relevante
- 1 no relevante

PRC = (3 + 0.5×1) / 5 = 0.70 (70%)

Criterio de evaluación:

- Excelente: PRC > 0.70
- Aceptable: PRC entre 0.40-0.70
- Insuficiente: PRC < 0.40

Justificación: Es la métrica fundamental de calidad base. Sin buena precisión, las innovaciones del AGPC no tienen valor práctico.

Destaca innovación prototipo AGPC: Parcialmente, ya que mide la calidad base de ambos sistemas

4.4.6 Coherencia de Respuesta (CR)

Tipo: Métrica manual

Escala: Evaluación humana de 1 a 5

Definición de la escala:

1. Inaceptable: Respuesta irrelevante, incoherente o completamente errónea
2. Deficiente: Respuesta confusa, incoherente o con información contradictoria
3. Aceptable: Respuesta parcialmente útil, coherencia básica presente
4. Buena: Respuesta útil con coherencia sólida, pero puede tener pequeñas omisiones
5. Excelente: Respuesta completa, coherente, bien estructurada y directamente útil

Interpretación: Evalúa la utilidad práctica final del sistema desde la perspectiva del usuario. Un sistema puede tener buena precisión técnica pero generar respuestas mal articuladas.

Proceso de evaluación: Para cada consulta, un evaluador lee la respuesta generada por el sistema (usando los contextos recuperados) y asigna un puntaje según la escala anterior.

Criterio de evaluación:

- Excelente: CR promedio > 4.0
- Aceptable: CR promedio entre 3.0-4.0
- Insuficiente: CR promedio < 3.0

Justificación: Esta métrica captura la calidad end-to-end del sistema. Es posible tener alta PRC(Precisión de Recuperación Contextual) pero baja CR(Coherencia de Respuesta) si el LLM no sintetiza bien los contextos, o viceversa.

Destaca innovación AGPC: Indirectamente, lo que busca es validar la propagación y gestión temporal, mejoran las respuestas finales.

4.4.7 Tabla Resumen del Sistema de Métricas

	Métrica	Tipo	Tiempo de Evaluación	Destaca Innovación AGPC
1	**TR - Tiempo de Respuesta**	Automática	0 min	Mide viabilidad práctica
2	**CT - Cobertura Temporal**	Automática	0 min	Dimensión temporal
3	**PP - Profundidad de Propagación**	Automática	0 min	Propagación en grafo
4	**CIR - Contextos Indirectos Relevantes**	Automática	0 min	Propagación en grafo
5	**PRC - Precisión de Recuperación**	Manual	2–3 horas	Calidad base
6	**CR - Coherencia de Respuesta**	Manual	1–2 horas	Utilidad práctica

Total tiempo de evaluación manual estimado: 3–5 horas con 15 consultas de prueba

Total de métricas: 6 (4 automáticas + 2 manuales)

Nota: Los resultados experimentales completos se presentarán en la Sección 5 - Resultados.

4.4.8 Proceso de Recolección y Validación de Métricas

La recolección de las métricas definidas en las secciones anteriores se realizó mediante un proceso estructurado de evaluación que combina asistencia automatizada con validación manual:

Automatización de Métricas Objetivas:

Las métricas TR (Tiempo de Respuesta), PP (Profundidad de Propagación) y CIR (Contextos Indirectos Recuperados) fueron obtenidas directamente de los logs de sistema mediante instrumentación del código, garantizando precisión y reproducibilidad.

Evaluación Asistida de Métricas Subjetivas:

Para las métricas que requieren juicio cualitativo (PRC, CR, CT), se implementó un agente evaluador especializado configurado con:

- Criterios de evaluación completos (Secciones 4.4.1 a 4.4.7)
- Acceso al dataset legal y código de ambos sistemas
- Plantilla estructurada para evaluación consistente

Control de Calidad mediante Validación Humana:

Todas las evaluaciones preliminares generadas por el agente fueron sometidas a:

- Verificación manual de relevancia contextual caso por caso
- Validación cruzada con contenido real del dataset
- Refinamiento de puntuaciones según criterios propios del evaluador

Este enfoque híbrido permitió mantener rigor científico mientras se optimizaba el tiempo de evaluación, reduciendo la carga de trabajo de 20-25 horas aproximadamente (enfoque completamente manual) a 3-5 horas efectivas, sin comprometer la validez de los resultados.

La configuración del agente evaluador y el proceso de refinamiento manual constituyen aportes metodológicos adicionales de este trabajo, demostrando que la evaluación de los sistemas(AGPC y RAG estándar) puede beneficiarse de herramientas de IA manteniendo una calidad académica.

5. Resultados

Este capítulo presenta los hallazgos obtenidos de la evaluación experimental comparativa entre el prototipo AGPC y un sistema RAG estándar. Se inicia detallando las condiciones controladas del experimento y el dataset de dominio legal utilizado. Posteriormente, se exponen los resultados cuantitativos desglosados por métricas de tiempo, cobertura y precisión, analizando el comportamiento del sistema en consultas temporales, estructurales y mixtas. El capítulo concluye con la validación formal de los objetivos de la investigación y un análisis de las limitaciones técnicas observadas durante las pruebas.

Condiciones Experimentales de la Evaluación Comparativa

La evaluación comparativa entre el prototipo AGPC y el sistema RAG estándar se realizó bajo condiciones controladas para garantizar la validez de los resultados obtenidos. A continuación se detallan los parámetros comunes y las diferencias metodológicas entre ambos sistemas:

Dataset y representación común:

Ambos sistemas fueron evaluados utilizando un dataset idéntico compuesto por 200 conversaciones legales sintéticas, el cual generó una representación estructurada de 269 nodos contextuales interconectados mediante 1.432 aristas en el grafo contextual probabilístico. Este dataset abarca diversos tipos de casos legales (amparo por mora administrativa, divorcios, trabajo no registrado, acoso laboral, incumplimiento contractual, entre otros) distribuidos temporalmente entre enero y agosto de 2023.

Componentes tecnológicos equivalentes:

Para garantizar una comparación equitativa, se utilizaron los mismos componentes tecnológicos en ambas implementaciones:

- **Modelo de embeddings:** sentence-transformers/all-MiniLM-L6-v2, que genera vectores de 384 dimensiones.
- **Base de datos vectorial:** ChromaDB con indexación HNSW (Hierarchical Navigable Small World), configurada con métricas de similitud coseno.
- **Modelo de lenguaje:** Google Gemini 2.0 Flash Experimental, con parámetros de generación idénticos (temperatura: 0.3, top_p: 0.95, top_k: 40, max_output_tokens: 2048).
- **Recuperación inicial:** Ambos sistemas configurados para recuperar los 5 contextos directos más relevantes (TOP-5).

Configuración específica del AGPC:

El prototipo AGPC fue ejecutado con los siguientes parámetros estándar:

- **Umbral de similitud estructural:** 0.5 (50%), valor mínimo para considerar la creación de una arista entre nodos en el grafo contextual.
- **Factor de refuerzo temporal:** 1.5, aplicado cuando el sistema detecta intención temporal media en la consulta del usuario.
- **Resultados iniciales directos (k):** 5 contextos recuperados mediante búsqueda vectorial antes de iniciar la propagación.
- **Configuración de propagación activa:**
    - Pasos máximos: 2 saltos desde los nodos iniciales.
    - Factor de decaimiento: 0.8, aplicado en cada nivel de propagación para atenuar la activación transmitida.

Infraestructura de ejecución:

Es importante destacar que los sistemas fueron ejecutados en entornos computacionales diferentes, lo cual introduce un factor de variabilidad en las métricas de rendimiento:

- **AGPC:** Ejecutado localmente en una notebook personal con recursos limitados de CPU (procesador Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz (2.30 GHz)) y 8 GB de RAM, sin aceleración GPU.
- **RAG estándar:** Ejecutado en Google Colab, plataforma en la nube que proporciona recursos superiores, incluyendo procesadores más potentes, mayor memoria RAM y, acceso a aceleradores GPU.

Esta diferencia en la infraestructura favorece significativamente al sistema RAG estándar en términos de tiempo de respuesta, ya que los cálculos vectoriales, la recuperación de embeddings y la generación de respuestas se benefician de la mayor capacidad computacional disponible en entornos cloud optimizados.

A pesar de esta desventaja infraestructural, el objetivo de la evaluación es demostrar que el AGPC mantiene tiempos de respuesta aceptables (dentro del umbral de 25 segundos establecido) mientras proporciona funcionalidades superiores en la gestión de la dimensión temporal, validando así la viabilidad práctica del enfoque propuesto.

Metodología de Evaluación Asistida por IA:

Dada la magnitud de la evaluación comparativa (15 consultas × 2 sistemas × 6 métricas = 180 puntos de datos) y la necesidad de mantener consistencia en los criterios de evaluación, se implementó un enfoque metodológico híbrido que combina evaluación automatizada asistida por IA con verificación y refinamiento manual:

Fase 1 - Configuración del Agente Evaluador:
Se diseñó y configuró un agente de inteligencia artificial especializado utilizando Claude (Anthropic) con acceso a:

- Marco teórico completo del AGPC incluyendo arquitectura, métricas y criterios de evaluación
- Dataset completo de 200 conversaciones legales sintéticas
- Código fuente del sistema RAG estándar
- Plantilla estructurada de evaluación con criterios detallados para cada métrica
- Contexto del dominio legal y casos de uso específicos

El agente fue instruido para analizar las respuestas de ambos sistemas aplicando los criterios de evaluación establecidos en la Sección 4.4, garantizando uniformidad en la aplicación de los umbrales y escalas de medición.

Fase 2 - Evaluación Automatizada Inicial:
Para cada una de las 15 consultas experimentales, el agente evaluador recibió:

- Pregunta planificada con tipo (Temporal/Estructural/Mixta) y métrica principal
- Respuesta generada por AGPC incluyendo metadatos de propagación
- Logs de consola del sistema AGPC con métricas de profundidad (PP) y contextos indirectos (CIR)
- Respuesta generada por RAG estándar con latencia y contextos recuperados

El agente generó evaluaciones preliminares siguiendo la plantilla estructurada, calculando:

- PRC (Precisión de Recuperación Contextual) mediante análisis de relevancia de cada contexto recuperado
- CR (Coherencia de Respuesta) aplicando la escala 1-5 con criterios predefinidos
- CT (Cobertura Temporal) identificando casos correctamente filtrados por ventana temporal
- Observaciones cualitativas sobre patrones de éxito y fallo

Fase 3 - Verificación y Refinamiento Manual:
Cada evaluación preliminar generada por el agente fue sometida a un proceso de validación manual que incluyó:

1. **Verificación de relevancia contextual:** Revisión casó por caso de cada contexto recuperado, contrastando con el contenido real del dataset para confirmar o ajustar las clasificaciones (Relevante / Parcial / No relevante).
2. **Validación de cobertura temporal:** Para consultas temporales, verificación manual de que los casos identificados efectivamente corresponden a las ventanas temporales especificadas, utilizando los metadatos de fecha del dataset.
3. **Refinamiento de coherencia:** Evaluación crítica de las puntuaciones CR(Coherencia de Respuesta) asignadas por el agente, ajustando según criterios de completitud, precisión y utilidad práctica de las respuestas en contexto legal.
4. **Identificación de patrones emergentes:** Análisis cualitativo de casos atípicos, fallos compartidos entre sistemas, y comportamientos no capturados por métricas cuantitativas.

Esta metodología híbrida permitió:

- Reducir el tiempo de evaluación de aproximadamente 20-25 horas (enfoque completamente manual) a 3-5 horas (evaluación asistida + refinamiento)
- Mantener consistencia en la aplicación de criterios mediante el agente evaluador
- Preservar el juicio crítico humano en casos ambiguos o que requieren comprensión profunda del dominio legal
- Documentar de forma sistemática observaciones cualitativas que enriquecen el análisis cuantitativo

Consideraciones sobre validez: El uso de un agente de IA como evaluador preliminar introduce una dependencia metodológica que debe ser reconocida. Para mitigar posibles sesgos, se implementaron las siguientes salvaguardas:

- El agente evaluador no tiene acceso a las hipótesis de investigación ni a resultados esperados
- Todos los criterios de evaluación fueron definidos a priori (Sección 4.4) antes de configurar el agente
- La verificación manual actúa como control de calidad, corrigiendo interpretaciones erróneas del agente
- Los casos de desacuerdo entre evaluación automática y manual fueron charladas con el director del trabajo final para tener una observación adicional.

Este proceso de evaluación asistida por IA representa una contribución metodológica adicional del trabajo, demostrando el manejo para crear agente de IA y además que la evaluación de estos prototipos o sistemas como RAG puede beneficiarse de herramientas de IA mientras se mantiene control mediante la validación humana.

5.1 Resultados Cuantitativos

La evaluación comparativa entre el AGPC y RAG estándar se realizó mediante 15 consultas diseñadas, distribuidas en tres categorías: **5 consultas temporales** (énfasis en el tiempo), **5 consultas estructurales** (énfasis en relaciones semánticas), y **5 consultas mixtas** (combinación de ambas). Los resultados cuantitativos se presentan en la siguiente tabla:

Tabla 9. Resumen Estadístico Comparativo AGPC vs. RAG Estándar

Métrica	AGPC	RAG Estándar	Diferencia	Evaluación
TR - Tiempo de Respuesta (segundos)	10.14	3.43	+6.71 (+195%)	AGPC: Aceptable RAG: Aceptable
PP - Profundidad de	0.88	N/A	N/A	AGPC: Aceptable

Propagación (saltos)				
CIR - Contextos Indirectos (%)	76.31	N/A	N/A	AGPC: Problemático
PRC - Precisión de Recuperación (%)	42.64	40.67	+1.97	AGPC: Aceptable RAG: Aceptable
CR - Coherencia de Respuesta (1-5)	3.80	3.13	+0.67 (+21%)	AGPC: Aceptable RAG: Aceptable
CT - Cobertura Temporal (%)	69.29	4.13	+65.16	AGPC: Aceptable RAG: Insuficiente

Nota: Las métricas PRC, CR y CT fueron evaluadas mediante proceso híbrido (evaluación asistida por IA + validación manual). Ver Sección 5.4.8 para detalles metodológicos.

Interpretación general:

Los resultados revelan un patrón de **especialización funcional** entre ambos sistemas:

1. **AGPC demostró superioridad absoluta en gestión temporal** (CT: 69,29% vs 4,13%), validando la hipótesis central de que el modelado explícito de decaimiento temporal y factor de refuerzo mejora de gran medida la recuperación de información distribuida en el tiempo.
2. **RAG estándar mantiene ventaja en eficiencia computacional** (TR: 3,43 s vs. 10,14 s), resultado esperado dado que no ejecuta propagación en grafo ni cálculos de relevancia temporal.
    a. Además de tener mayor computo, ya que corre en Google Colab y en prototipo corre en una notebook con muchos menos recursos que un servidor en la nube
3. **Ambos sistemas exhiben precisión de recuperación comparable** (PRC: 42,64% vs. 40,67%), sugiriendo que la búsqueda semántica base es de calidad similar, pero AGPC logra mayor coherencia final (CR: 3,80 vs. 3,13) mediante la integración de contextos indirectos.
4. **La métrica CIR (76,31%) señala un problema de ruido por propagación**, superando el umbral óptimo de 20-40% establecido antes de realizar la metricas. Este hallazgo es importante y se analiza en profundidad en la Sección 5.5.

5.2 Análisis de Desempeño por Tipo de Consulta

5.2.1 Consultas Temporales (Consultas 1-5)

Resultados:

Métrica	AGPC (promedio)	RAG (promedio)	Mejor sistema
TR (s)	11,76	3,33	RAG (3,5 veces más rápido)
PP	1,13	N/A	AGPC
CIR (%)	94,46	N/A	AGPC
PRC (%)	54,28	10,00	AGPC (5,4 veces mejor aproximadamente)
CR (1-5)	4,40	2,40	AGPC (83% mejor)
CT (%)	98,58	0,00	AGPC (infinitamente mejor)

Hallazgos clave:

1. **Victoria de AGPC en cobertura temporal:** CT promedio de 98,58% demuestra que el sistema es capaz de identificar y recuperar de forma consistentemente información histórica relevante, incluso cuando los eventos están separados por meses.
2. **Fallo de RAG en filtrado temporal:** CT de 0% en consultas temporales revela que RAG estándar no posee de mecanismos para aplicar restricciones temporales. Por ejemplo:
    - **Consulta 1** ("Casos de agosto 2023"): RAG recuperó únicamente casos de "Amparo por mora administrativa" que no corresponden al período solicitado (PRC=0%, CR=1).
    - **Consulta 4** ("Divorcios primer semestre 2023"): RAG no identificó ningún caso del período correcto.
3. **Precisión superior de AGPC:** PRC promedio de 54,28% vs 10% de RAG indica que AGPC no solo filtra temporalmente, sino que mantiene relevancia semántica en los contextos recuperados.
4. **Coherencia significativamente mayor:** CR promedio de 4.40 vs 2.40 refleja que las respuestas de AGPC son cualitativamente superiores.

5.2.2 Consultas Estructurales (Consultas 6-10)

Resultados:

Métrica	AGPC (promedio)	RAG (promedio)	Mejor sistema
TR (s)	10,19	3,75	RAG (2,7 veces más rápido)
PP	0,83	N/A	AGPC
CIR (%)	67,50	N/A	AGPC
PRC (%)	48,28	70,00	RAG (45% mejor)
CR (1-5)	3,80	4,20	RAG (11% mejor)

Hallazgos clave:

1. **RAG demuestra ligera ventaja en precisión pura:** PRC promedio de 70% vs 48,28% indica que, en consultas sin componente temporal, la búsqueda semántica directa de RAG es más eficiente en filtrar ruido.
2. **AGPC sacrifica precisión por exhaustividad (recall):** El sistema recupera mayor volumen de contextos, (CIR=67,46%), incluyendo relaciones indirectas, lo que reduce PRC, pero puede aumentar cobertura temática.
3. **Coherencia comparable con ligera ventaja de RAG:** CR de 4,20 vs. 3,80 sugiere que la menor cantidad de contextos con baja relación facilita al LLM generar respuestas más enfocadas.
4. **Propagación genera ruido contextual:** En consultas estructurales, la propagación de AGPC recupera contextos conectados pero no necesariamente con alta relación con la temática consultada.

Nota importante: Esta comparativa fue realizada con la configuración estándar de propagación (Cantidad de pasos 2 y factor de decaimiento 0,8), si se desactiva la propagación los resultados anteriormente obtenidos quedan más similares con el sistema de RAG estándar.

Ejemplo ilustrativo - Consulta 6: "¿Qué precedentes existen sobre incumplimiento de régimen de visitas?"

- **AGPC:** Recuperó 24 contextos mediante propagación (CIR=100%), de los cuales:
  - 12 relevantes (casos de incumplimiento de régimen de visitas)
  - 12 irrelevantes (amparos, inconstitucionalidad, habeas data)
  - **PRC=50%**, **CR=5/5**
  - **Ventaja:** Capturó TODOS los 6 casos existentes en el dataset
- **RAG:** Recuperó 5 contextos, todos relevantes:

- 5 casos de incumplimiento (Casos 1, 2, 4, 5, 6)
- **PRC=100%**, **CR=5/5**
- **Limitación:** No recuperó el Caso 3, perdiendo 1/6 de los precedentes

Análisis crítico: Este caso revela un **trade-off fundamental**:

- AGPC prioriza **recall** (no perder información relevante) a costa de **precisión** (incluir ruido)
- RAG prioriza **precisión** (evitar ruido) a costa de **recall** (puede perder información)

En el dominio legal, donde la omisión de un precedente puede tener consecuencias importantes, el enfoque de AGPC puede ser preferible, pero requiere optimización para reducir ruido.

5.2.3 Consultas Mixtas (Consultas 11-15)

Resultados:

Métrica	AGPC (promedio)	RAG (promedio)	Mejor sistema
TR (s)	8,49	3,20	RAG (2,7 veces más rápido aproximadamente)
PP	0,69	N/A	AGPC
CIR (%)	67	N/A	AGPC
PRC (%)	25,36	42,00	RAG (66% mejor)
CR (1-5)	3,20	2,80	AGPC (14% mejor)
CT (%)	40,00	6,60	AGPC (506% mejor)

Hallazgos clave:

1. **Desempeño mixto refleja complejidad de consultas híbridas:** Ambos sistemas exhiben métricas intermedias, con PRC más bajo que en consultas puras.
2. **AGPC mantiene ventaja en dimensión temporal:** Cuando la consulta incluye restricción temporal (ej.: Consulta 12 "¿Hubo casos de acoso laboral entre junio y agosto de 2023 y qué medidas se recomendaron?"), AGPC logra CT=100% vs. CT=33% de RAG.
3. **Precisión reducida en ambos sistemas:** PRC de 25,36% (AGPC) y 42% (RAG) indica que consultas que combinan múltiples criterios generan mayor dificultad de recuperación.

Ejemplo ilustrativo - Consulta 12: "¿Hubo casos de acoso laboral entre junio y agosto de 2023 y qué medidas se recomendaron?"

- **AGPC:**
  - Recuperó 43 contextos (CIR=95,3%)
  - Identificó correctamente 1 caso (Caso 3, 21 julio 2023) dentro del período
  - **PRC=17,44%** (mucho contexto con poca relación por propagación)
  - **CR=5/5** (respuesta precisa con fecha exacta y medidas detalladas)
  - **CT=100%** (aplicó filtro temporal correctamente)
- **RAG:**
  - Recuperó 5 contextos
  - Identificó 1 caso correcto + 2 casos fuera del período (especuló fechas incorrectamente)
  - **PRC=60%**
  - **CR=3/5** (especulación temporal incorrecta)
  - **CT=33%** (error crítico en inferencia temporal)

Análisis crítico: Este caso evidencia la **ventaja crucial de AGPC en gestión temporal explícita**. Aunque AGPC detector más contextos, pero con poca relación con la pregunta (PRC bajo), su capacidad de aplicar ventanas temporales precisas evita errores fundamentales que RAG comete al especular sobre fechas.

5.3 Análisis de Profundidad de Propagación y Descubrimiento de Relaciones Indirectas

5.3.1 Distribución de Profundidad de Propagación

El análisis de la métrica PP (Profundidad de Propagación) revela patrones interesantes sobre cómo AGPC descubre información:

Tabla 9. Distribución de Profundidad de Propagación por Tipo de Consulta

Tipo de Consulta	PP Promedio	Interpretación
Temporal	1,13	Aceptable
Estructural	0,83	Aceptable

Mixta	0,69	Problemático
Global	**0,88**	**Aceptable**

Hallazgos clave:

1. **PP promedio de 0,88 saltos** indica que AGPC opera predominantemente en búsqueda directa (nivel 0) con propagación selectiva a primer nivel (nivel 1). Esto está **dentro del rango aceptable (0,8-1,5)** según los criterios establecidos.
2. **Consultas temporales muestran mayor propagación** (PP=1.13), sugiriendo que el filtrado temporal reduce el conjunto candidato inicial, incentivando al sistema a explorar vecinos indirectos.
3. **Ninguna consulta alcanzó el rango óptimo (1,5-2,5 saltos)**, indicando que la configuración actual de propagación (2 pasos máximos, umbral de activación 0,01) puede ser demasiado conservadora.

5.3.2 Efectividad del Descubrimiento de Relaciones Indirectas

Caso exitoso - Consulta 1 (¿Qué casos legales se discutieron durante agosto de 2023?):

Nodos recuperados por nivel:

- Nivel 0 (directos): 3 contextos
- Nivel 1 (1 salto): 30 contextos
- Nivel 2 (2 saltos): 28 contextos

Total: 61 contextos

Relevancia por nivel:

- Nivel 0: 3/3 relevantes (100%)
- Nivel 1: 25/30 relevantes (83%)
- Nivel 2: 20/28 relevantes (71%)

PP = (3×0 + 30×1 + 28×2) / 61 = 1,48 saltos

CIR = 58/61 = 95,1%

Interpretación: La propagación permitió expandir de 3 contextos iniciales a 61, manteniendo alta relevancia incluso en nivel 2. Esto demuestra que el grafo contextual probabilístico captura efectivamente relaciones semántico-temporales de forma útil.

Caso problemático - Consulta 10 (¿Cuáles son los plazos legales mencionados en casos de ejecución fiscal?):

Nodos recuperados por nivel:

- Nivel 0: 5 contextos (Amparo por mora)
- Nivel 1: 32 contextos (Divorcios, disoluciones, etc.)
- Nivel 2: 7 contextos (Acciones, inconstitucionalidad, habeas data)

Relevancia:

- Todos 0% relevantes (no existen casos de ejecución fiscal en dataset)

PP = 1,05 saltos

CIR = 88,6%

Interpretación: Cuando el tema solicitado no existe en el dataset, la propagación amplifica con contextos que tienen poca relación. Esto evidencia la necesidad de un **mecanismo de early stopping** basado en relevancia semántica mínima.

- Otra manera de obtener mejores resultados sería tener un umbral de activación mayor, que este indica el porcentaje que debe pasar para poder considerarlo al contexto dentro de la respuesta que se le da al usuario.

5.4 Limitaciones Observadas

5.4.1 Problema Crítico: CIR Excesivo (76,31%)

Diagnóstico:

La métrica CIR promedio de 76,31% **supera el umbral óptimo (20-40%)**, indicando que el sistema recupera demasiados contextos con poca relación con la pregunta del usuario. Esto genera dos problemas:

1. **Ruido contextual:** El LLM recibe información poco relacionada que puede diluir la precisión de la respuesta.
2. **Ineficiencia computacional:** Procesar contextos con poca relación, consume recursos sin aportar mucho valor.

Análisis de consultas con mayor problemática para el prototipo AGPC:

Consulta	CIR (%)	PRC (%)	Evaluación
1. ¿Qué casos legales se discutieron durante agosto de 2023?	95,1	78-80	Poco Ruido (alta PRC)
6. ¿Qué precedentes existen sobre incumplimiento de régimen de visitas?	100,0	50	Ruido aceptable
10. ¿Cuáles son los plazos legales mencionados en casos de ejecución fiscal?	88,6	0	**Ruido crítico**
12. ¿Hubo casos de acoso laboral entre junio y agosto de 2023 y qué medidas se recomendaron?	95,3	17,44	**Ruido crítico**

Patrón identificado: con CIR alto NO es problemático, sino cuando se correlaciona con PRC bajo. Las Consultas 10 y 12 muestran el peor escenario: máxima propagación con mínima relevancia.

Nota: **Umbral de activación demasiado bajo:** Permite que nodos débilmente conectados se incluyan en la propagación.

Soluciones propuestas (trabajo futuro):

Propagación con Poda Semántica

Para abordar el problema del ruido excesivo en la propagación, propongo implementar un mecanismo de poda semántica que valide la relevancia de cada nodo antes de incluirlo en la cadena de propagación. Esta mejora consistiría en:

- **Validación pre-propagación**: Antes de propagar la activación a un nodo vecino, calcular la similitud semántica entre el embedding de la consulta original y el embedding del nodo candidato.

- **Umbral de relevancia semántica**: Establecer un umbral mínimo de similitud (por ejemplo, 0,3) que un nodo debe superar para ser considerado relevante y recibir activación propagada.
- **Propagación condicionada**: Solo propagar la activación hacia vecinos que superen tanto el umbral de activación actual como el umbral de relevancia semántica, evitando así la inclusión de contextos con poca relación.

Este enfoque mantendría los beneficios de la propagación por grafo (recuperación de contextos indirectos fundamentales) mientras reduce el ruido introducido por nodos semánticamente distantes que actualmente se recuperan únicamente por proximidad estructural.

Ajuste Dinámico de Profundidad

Complementariamente, propongo un mecanismo de ajuste automático de la profundidad de propagación basado en la densidad y calidad de los resultados obtenidos en cada nivel:

- **Evaluación por nivel**: Tras completar la propagación en cada nivel de profundidad, evaluar la calidad promedio de los contextos recuperados mediante su similitud semántica con la consulta.
- **Criterio de detención adaptativo**: Si la similitud promedio de los contextos en un nivel cae por debajo de un umbral definido, detener la propagación antes de alcanzar la profundidad máxima configurada.
- **Optimización del balance**: Este mecanismo permitiría optimizar automáticamente el balance entre cobertura (profundidad) y precisión (calidad), adaptándose a las características específicas de cada consulta sin requerir **configuración manual**.

5.4.2 Casos de Fallo Compartido (Consultas 8, 10, 13)

Consulta 8: "¿Qué dice la Ley 24013 según las conversaciones registradas?"

- **Ambos sistemas:** PRC=50%, CR=2-3/5
- **Problema:** Búsqueda semántica por referencia legal específica falló. Recuperaron contextos de Habeas Corpus en lugar de Trabajo no registrado.
- **Causa:** Los embeddings no capturan bien identificadores legales alfanuméricos ("Ley 24013").

Consulta 10: "¿Cuáles son los plazos legales mencionados en casos de ejecución fiscal?"

- **Ambos sistemas:** PRC=0%, CR=3/5
- **Problema:** No existen casos de ejecución fiscal en el dataset. RAG recupera información que no es relevante, en cambio, AGPC recupera información con poca relación con base en la pregunta raíz.

Consulta 13: "¿Qué casos mencionan al artículo 246 de la LCT y en qué fechas del año 2023 ocurrieron?"

- **Ambos sistemas:** PRC=0%, CR=1/5
- **Problema:** Fallo crítico. Existen casos de acoso laboral (Casos 2, 3) que mencionan explícitamente art. 246 LCT, pero ningún sistema los recuperó.

Implicación: Estos fallos revelan **limitaciones compartidas en búsqueda por referencias legales específicas**, requiriendo:

1. **Extracción de entidades legales:** Identificar y etiquetar artículos, leyes, códigos como metadatos estructurados.
2. **Detección de ausencia de información:** Implementar umbral de confianza mínimo; si ningún resultado supera el umbral, reportar explícitamente "información no disponible".

5.4.3 Limitaciones en Consultas Estructurales

Observación: En 3 de 5 consultas estructurales (Consultas 6, 7, 10), AGPC recuperó mayor volumen de contextos pero con menor precisión que RAG.

Análisis cualitativo:

Consulta	Contextos AGPC	Contextos RAG	Observación
6. Precedentes régimen visitas	24 (50% relevantes)	5 (100% relevantes)	AGPC: alta cobertura, baja precisión
7. Argumentos trabajo no registrado	29 (42% relevantes)	5 (100% relevantes)	AGPC sacrifica precisión por exhaustividad
9. Cláusula penal	8 (100% relevantes)	5 (100% relevantes)	AGPC recupera más casos con igual precisión

Patrón: Cuando la consulta tiene alta especificidad temática (ej: "cláusula penal por incumplimiento"), la propagación de AGPC es efectiva. Cuando la consulta es más amplia

(ej: "precedentes sobre régimen de visitas"), la propagación introduce contextos con poca relación.

Hipótesis explicativa: El algoritmo de propagación actual no ajusta su agresividad según la especificidad de la consulta. Una consulta específica debería limitar la propagación, mientras que una consulta exploratoria podría beneficiarse de mayor divulgación.

5.5 Validación de Objetivos

5.5.1 Objetivo General

Objetivo establecido: Desarrollar un prototipo funcional de un Agente de Gestión Probabilística de Contextos (AGPC) que gestione información contextual mediante un Grafo Contextual Probabilístico, mejorando la coherencia, trazabilidad y transferencia de conocimiento en agentes inteligentes.

Estado: CUMPLIDO

Evidencia:

- Prototipo funcional implementado con 15 versiones iterativas documentadas (Fase 2, Sección 4.1.2)
- Arquitectura modular operativa con 6 capas funcionales (Sección 5.2.1):
    - Interfaz de Usuario (FastAPI)
    - Motor de Fragmentación
    - Sistema de Embeddings Semánticos (ChromaDB)
    - Motor de Grafo Contextual Probabilístico (NetworkX)
    - Módulo de Análisis Temporal
    - Motor de Propagación y Recuperación
- Grafo contextual con 269 nodos y 1.432 aristas con pesos probabilísticos normalizados
- Prototipo funcionando y evaluado con dataset de 200 conversaciones legales sintéticas

5.5.2 Objetivos Específicos

Objetivo Específico 1: Diseñar la arquitectura modular del AGPC

Estado: CUMPLIDO

Componentes implementados:

- Módulo de extracción semántica (fragmentador.py, pdf_processor.py)
- Memoria vectorial (ChromaDB con modelo all-MiniLM-L6-v2, 384 dimensiones)
- Grafo Contextual Probabilístico (NetworkX con persistencia JSON)

- Motor de respuesta integrado con Gemini 2.0 Flash
- Interfaz web con visualización interactiva de grafos (D3.js)

Decisiones arquitectónicas clave:

- Separación de responsabilidades por capas
- Actualización incremental O(n) vs. O(n^2) original
- Batch processing para embeddings y similitudes
- Normalización sigmoidea de pesos efectivos: Pe_norm = Pe_efectivo / (1 + Pe_efectivo)

Objetivo Específico 2: Implementar el modelo de Grafo Contextual Probabilístico integrando similitud semántica, léxica y temporal

Estado: CUMPLIDO

Implementaciones realizadas:

2.1 Similitud Estructural:

- Combinación de similitud semántica (embeddings) + léxica (keywords)
- Fórmula: W_estructural = (SIM_keywords + SIM_semantica) / 2
- Normalización mediante función sigmoidea para garantizar Pe ∈ (0,1)

2.2 Dimensión Temporal:

- Decaimiento exponencial: R_temporal = e^(-Δdías/τ)
- Factor de refuerzo adaptativo según intención detectada por LLM:
  - Temporal fuerte: Fr = 2.5
  - Temporal media: Fr = 1.5
  - Temporal débil: Fr = 0.5
  - Atemporal(No temporal o solamente estructural): Fr = 0.0
- Detección automática de intención mediante análisis con Gemini

2.3 Peso Efectivo Adaptativo:

- Para consultas temporales: Pe = R_temporal × Fr × (1 + W_estructural)
- Para consultas estructurales o mixtas: Pe = W_estructural × (1 + R_temporal × Fr)

Validación técnica:

- Sistema procesa correctamente 269 nodos con metadatos enriquecidos
- 1.432 aristas con pesos normalizados garantizan consistencia probabilística
- Propagación de activación implementada con hasta 3 niveles de profundidad

Objetivo Específico 3: Evaluar comparativamente el rendimiento del AGPC frente a sistemas RAG estándar

Estado: CUMPLIDO

Protocolo experimental ejecutado:

- Dataset: 200 conversaciones legales sintéticas
- Consultas de prueba: 15 consultas categorizadas (5 temporales + 5 estructurales + 5 mixtas)
- Configuración equivalente: Mismo modelo de embeddings, LLM y misma Fragmentación
- Métricas evaluadas: 6 métricas cuantitativas (4 automáticas + 2 manuales)

Resultados comparativos obtenidos:

Tabla 10. Resultados Comparativos AGPC vs. RAG Estándar

Métrica	**AGPC**	**RAG Estándar**	**Ventaja AGPC**
TR - Tiempo de Respuesta (segundos)	10,14	3,43	RAG 195% más rápido
PP - Profundidad de Propagación (saltos)	0,88	N/A	Única del AGPC
CIR - Contextos Indirectos (%)	76,31	N/A	Única del AGPC
PRC - Precisión de Recuperación (%)	42,64	40,67	+5%
CR - Coherencia de Respuesta (1-5)	3,80	3,13	+21%
CT - Cobertura Temporal (%)	69,29	4,13	+1.577%

Análisis de resultados:

- Superioridad demostrada en gestión temporal: AGPC logró cobertura temporal 16 veces superior a RAG
- Coherencia de respuestas mejorada: 21% de mejora en calidad
- Precisión competitiva pero con ruido: PRC comparable, pero CIR excesivo (76% vs. óptimo 20-40%) indica necesidad de optimización

- Trade-off latencia-funcionalidad: AGPC es 3 veces más lento, pero mantiene tiempos aceptables (<25 s límite establecido).
    - **Este incremento en la latencia se explica por las diferencias en la infraestructura de ejecución:** el AGPC corre de forma local en una notebook (con recursos limitados de CPU y memoria) mientras que el RAG estándar se ejecuta en Google Colab, que dispone de entornos más potentes y optimizados para cargas de procesamiento. Debido a estas condiciones, el entorno local presenta mayores tiempos de cómputo, aunque dentro de los valores tolerables definidos para el sistema.

Conclusión del objetivo: La evaluación comparativa se ejecutó exitosamente y demostró ventajas del AGPC en dimensión temporal, con identificación de áreas de mejora.

Objetivo Específico 4: Validar la arquitectura del AGPC en el dominio legal

Estado: CUMPLIDO

Dataset de validación construido:

- 200 conversaciones legales sintéticas
- Con diversas categorías representadas como por ejemplo:
    - Amparo por mora administrativa
    - Divorcios y disoluciones
    - Trabajo no registrado
    - Acoso laboral
    - Incumplimiento contractual
- Rango temporal: enero 2023 - agosto 2023
- Total de contextos/fragmentos: aproximadamente 269

Enfoque incremental aplicado:

- Dataset construido progresivamente según capacidad de procesamiento
- Optimizaciones de rendimiento aplicadas iterativamente

Casos de validación exitosos:

- Consulta 1 (temporal): Sistema recuperó 61 contextos de agosto 2023, incluyendo casos indirectos mediante propagación (PP=1.48)
- Consulta 6 (estructural): Sistema identificó 6/6 casos de incumplimiento de régimen de visitas (recall 100%)
- Consulta 12 (mixta): Sistema aplicó ventana temporal correcta (junio-agosto 2023) evitando especulación de fechas que RAG cometió

Casos de fallo identificados:

- Consulta 10: Tema inexistente en dataset (ejecución fiscal) capturo contextos con poca relación por propagación
- Consulta 13: Fallo crítico compartido con RAG en búsqueda por artículos legales específicos (art. 246 LCT)

Conclusión del objetivo: La arquitectura fue validada exitosamente en el dominio legal, demostrando aplicabilidad práctica y revelando limitaciones específicas que guían trabajo futuro.

5.5.3 Cumplimiento de Criterios de Evaluación por Métrica

Métrica 1: Tiempo de Respuesta (TR)

Criterio establecido (Sección 4.4.1):

- Aceptable: TR < 25,000 ms
- Marginal: TR entre 25,000-90,000 ms
- Inaceptable: TR > 90,000 ms

Resultado obtenido: TR promedio = 10.14 s (10,140 ms)

Evaluación: **CRITERIO CUMPLIDO**

- TR máximo observado: 17.13 s (Consulta 1, aun dentro de rango)
- Viabilidad práctica del sistema demostrada

Métrica 2: Cobertura Temporal (CT)

Criterio establecido (Sección 4.4.2):

- Excelente: CT > 70%
- Aceptable: CT entre 50%-70%
- Insuficiente: CT < 50%

Resultados obtenidos:

- AGPC: CT = 69,29% (cercano a excelente)
- RAG Estándar: CT = 4,13% (insuficiente)

Evaluación: **CRITERIO CUMPLIDO**

- AGPC supera 16 veces el desempeño de RAG en consultas temporales
- En consultas temporales puras (Consultas 1-5): CT = 98,58% (excelente)
- Validación directa de la innovación del decaimiento temporal + factor de refuerzo

Métrica 3: Profundidad de Propagación (PP)

Criterio establecido (Sección 4.4.3):

- Óptimo: PP entre 1.5-2.5 saltos
- Aceptable: PP entre 0.8-1.5 o 2.5-3.0
- Problemático: PP < 0.8 (no propaga) o PP > 3.0 (ruido excesivo)

Resultado obtenido: PP promedio = 0.88 saltos

Evaluación: **CUMPLIDO (límite aceptable)**

- Sistema opera en rango aceptable bajo (0.8-1.5)
- Distribución por tipo de consulta:
  - Temporal: PP=1.13 (aceptable)
  - Estructural: PP=0.83 (aceptable)
  - Mixta: PP=0.69 (problemático)
- Implicación: Configuración actual (2 pasos máx., umbral 0.01) es conservadora
- Trabajo futuro: Ajustar parámetros para alcanzar rango óptimo 1.5-2.5

Métrica 4: Contextos Indirectos Recuperados (CIR)

Criterio establecido (Sección 4.4.4):

- Óptimo: CIR entre 20%-40%
- Aceptable: CIR entre 10%-20% o 40%-50%
- Problemático: CIR < 10% (no propaga) o CIR > 50% (ruido excesivo)

Resultado obtenido: CIR promedio = 76,31%

Evaluación: **NO CUMPLIDO (ruido excesivo)**

- Sistema supera significativamente el umbral óptimo
- Problema identificado: Propagación recupera demasiados contextos con poca relación
- Casos críticos:
  - Consulta 10: CIR=88,6%, PRC=0% (propagación en tema inexistente)
  - Consulta 12: CIR=95,3%, PRC=17,44% (ruido)
- Soluciones propuestas (Sección 4.4.1):
  - Propagación con poda semántica
  - Ajuste dinámico de profundidad
  - Umbral de activación mayor

Métrica 5: Precisión de Recuperación Contextual (PRC)

Criterio establecido (Sección 4.4.5):

- Excelente: PRC > 0.70
- Aceptable: PRC entre 0.40-0.70
- Insuficiente: PRC < 0.40

Resultados obtenidos:

- AGPC: PRC = 42,64% (aceptable)
- RAG Estándar: PRC = 40,67% (aceptable)

Evaluación: **CUMPLIDO (aceptable)**

- Ambos sistemas en rango aceptable, con ligera ventaja de AGPC (+5%)
- Distribución por tipo de consulta:
  - Temporal: AGPC=54,28% vs. RAG=10% (AGPC muy superior)
  - Estructural: AGPC=48,28% vs. RAG=70% (RAG superior)
  - Mixta: AGPC=25,36% vs. RAG=42% (RAG superior)
- Patrón identificado: AGPC sacrifica precisión en consultas estructurales para maximizar cobertura (recall)
- Trade-off validado: En dominio legal, recall alto es preferible (no perder precedentes críticos)

Métrica 6: Coherencia de Respuesta (CR)

Criterio establecido (Sección 4.4.6):

- Excelente: CR promedio > 4,0
- Aceptable: CR promedio entre 3.0-4,0
- Insuficiente: CR promedio < 3,0

Resultados obtenidos:

- AGPC: CR = 3,80/5 (aceptable alto)
- RAG Estándar: CR = 3,13/5 (aceptable bajo)

Evaluación: **CUMPLIDO (aceptable, +21% mejora vs RAG)**

- Distribución por tipo de consulta:
  - Temporal: AGPC=4,40 vs RAG=2,40 (AGPC 83% superior)
  - Estructural: AGPC=3,80 vs. RAG=4,20 (RAG 11% superior)
  - Mixta: AGPC=3,20 vs. RAG=2,80 (AGPC 14% superior)
- Observación clave: La integración de contextos indirectos mediante propagación mejora coherencia incluso cuando introduce ruido (CIR alto)

- Validación cualitativa: Es probable que los usuarios perciban las respuestas AGPC como más completas y contextualizadas temporalmente

5.5.4 Síntesis: Validación de la Propuesta de Valor del AGPC

¿El AGPC aporta valor diferencial frente a RAG estándar?

Respuesta: Sí, con especificidad de dominio

Casos de uso donde AGPC es superior:

1. Consultas con componente temporal explícito (CT por 16 veces superior)
2. Necesidad de máximo recall (recuperar todos los precedentes, incluso con ruido)
3. Escenarios donde coherencia temporal es crítica (CR 83% superior en temporales)
4. Dominios donde información antigua permanece relevante

Casos de uso donde RAG es superior:

1. Consultas estructurales puras con alta especificidad temática
2. Mayor tiempo de respuesta
3. En Escenarios donde la precisión es más importante que la cobertura (PRC 45% superior en preguntas estructurales)

5.5.5 Limitaciones Reconocidas y Trabajo Futuro

Limitaciones Identificadas:

1. Ruido por Propagación Excesiva (CIR=76,31%)
    - Causa: Umbral de activación demasiado bajo + ausencia de poda semántica
    - Impacto: 3 consultas estructurales con PRC < 50%
    - Solución propuesta: Propagación con validación semántica pre-activación
2. Consultas por Referencias Legales Específicas
    - Ejemplo: Art. 246 LCT (Consulta 13) - ambos sistemas fallaron
    - Causa: Embeddings no capturan bien identificadores alfanuméricos
    - Solución propuesta: Extracción de entidades legales como metadatos estructurados

Contribuciones Validadas del Trabajo:

1. Primer sistema que integra gestión temporal explícita + propagación probabilística en grafos contextuales
2. Demostración empírica de superioridad en consultas temporales (factor de 16 veces mayor que RAG estándar)
3. Arquitectura modular y extensible validada con 15 versiones iterativas

4. Metodología de evaluación reproducible con métricas cuantitativas + manuales
5. Identificación de trade-offs fundamentales (latencia vs. funcionalidad, precisión vs. recall)

5.5.6 Resumen de la Validación

El Agente de Gestión Probabilística de Contextos (AGPC) cumple satisfactoriamente el objetivo general y los cuatro objetivos específicos establecidos, demostrando su viabilidad como prototipo funcional y su superioridad frente a RAG estándar en escenarios con componente temporal.

Balance final:

- 5/6 métricas cumplidas en rango objetivo (TR, CT, PRC, CR y PP)
- 1/6 métricas con limitaciones identificadas (CIR excesivo)
- Contribución al campo: Marco arquitectónico y metodológico para futuras investigaciones en gestión contextual

Las limitaciones identificadas (propagación excesiva, consultas por referencias legales) no invalidan la propuesta de valor, sino que definen claramente las direcciones de trabajo futuro para evolucionar el prototipo hacia un sistema de producción.

6. Conclusión

Para finalizar el presente trabajo final, este capítulo sintetiza los hallazgos principales de la investigación y evalúa el grado de cumplimiento de los objetivos propuestos. Se discuten las implicaciones teóricas y prácticas de la arquitectura de memoria dual validada, así como las contribuciones realizadas al campo de la gestión contextual. Asimismo, se reflexiona sobre las limitaciones identificadas y se sugieren líneas concretas de trabajo futuro para la evolución del AGPC hacia entornos productivos.

El presente trabajo ha demostrado la viabilidad y el valor diferencial del Agente de Gestión Probabilística de Contextos (AGPC) como una arquitectura innovadora para la gestión contextual en sistemas inteligentes. A través del diseño, implementación y evaluación de un prototipo funcional, se ha validado empíricamente que la integración de un Grafo Contextual Probabilístico con mecanismos explícitos de decaimiento temporal y propagación de activación supera significativamente las capacidades de los sistemas RAG estándar en escenarios donde la dimensión temporal es importante.

Cumplimiento de Objetivos y Contribuciones Principales

El objetivo general de desarrollar un prototipo funcional del AGPC que mejore la coherencia, trazabilidad y transferencia de conocimiento se cumplió. El sistema implementado integra de forma correcta seis capas arquitectónicas modulares: interfaz de usuario (FastAPI), motor de fragmentación semántica, sistema de embeddings (ChromaDB), motor de grafo probabilístico (NetworkX), módulo de análisis temporal con LLM, y motor de propagación y recuperación. Esta arquitectura modular no solo garantiza la separación de responsabilidades, sino que permite la evolución y sustitución de componentes individuales sin afectar la funcionalidad completa del sistema.

Los cuatro objetivos establecidos fueron alcanzados con resultados concretos y medibles:

1. Arquitectura modular del AGPC: Se implementó una estructura de seis capas operativas que procesan 269 nodos contextuales interconectados mediante 1.432 aristas con pesos probabilísticos normalizados. Las optimizaciones de rendimiento implementadas (actualización incremental O(n) vs. O(n²), batch processing, escritura diferida, caché de embeddings) redujeron los tiempos de carga de datasets entre 50-72% y los tiempos de recálculo de relaciones entre los nodos en un 90-95%, transformando el prototipo de un concepto teórico a una herramienta más practica.

2. Modelo de Grafo Contextual Probabilístico: Se formalizó matemáticamente la estructura $Gc = (V, E, Pv, Pe, \varphi)$ y se implementó exitosamente la combinación de similitud estructural (semántica + léxica), relevancia temporal, y factor de refuerzo adaptativo según intención detectada por LLM. La normalización sigmoidea garantiza que se pueda interpretar de forma probabilística y estabilidad en la propagación, resolviendo problemas decisivos de pesos efectivos que superaban 1.0 en versiones iniciales.

3. Evaluación comparativa con RAG estándar: El protocolo experimental ejecutado con 15 consultas categorizadas (5 temporales, 5 estructurales, 5 mixtas) sobre un dataset de 200 conversaciones legales sintéticas reveló un patrón funcional claro. El AGPC demostró superioridad en cobertura temporal (CT: 69,29% vs. 4,13% de RAG, una mejora del 1.577%), validando directamente la hipótesis central del trabajo. En consultas temporales puras, el AGPC alcanzó CT=98,58% con coherencia de respuesta (CR) de 4.40/5, mientras que RAG obtuvo CT=0% y CR=2,40/5. Esta diferencia evidencia que RAG carece completamente de mecanismos para gestionar restricciones temporales, generando respuestas que especulan incorrectamente sobre fechas o ignoran por completo la dimensión temporal de la consulta.

4. Validación en dominio legal: La selección del dominio legal como caso de aplicación principal se justificó por su complejidad contextual, dimensión temporal crítica, y necesidad documentada de gestión del conocimiento institucional. El prototipo procesó exitosamente 200 conversaciones distribuidas temporalmente entre enero y agosto de 2023, abarcando casos de amparo por mora administrativa, divorcios, trabajo no registrado, acoso laboral, e incumplimiento contractual. La validación reveló tanto capacidades sobresalientes (recuperación completa de precedentes temporalmente distribuidos) como limitaciones específicas (propagación excesiva en consultas estructurales, búsqueda por referencias legales alfanuméricas).

Hallazgos Clave y Validación Experimental

Los resultados cuantitativos obtenidos validan el modelo teórico propuesto mientras identifican áreas de mejora:

Fortalezas demostradas:

- **Gestión temporal superior:** CT promedio de 69,29% (AGPC) vs. 4,13% (RAG) demuestra que el modelo de decaimiento exponencial combinado con factor de refuerzo adaptativo funciona efectivamente en la práctica. En consultas temporales puras, el sistema alcanzó cobertura casi perfecta (98,58%), identificando consistentemente información histórica relevante incluso cuando los eventos estaban separados por meses.
- **Coherencia mejorada:** CR promedio de 3,80/5 (AGPC) vs. 3,13/5 (RAG) representa una mejora del 21% en calidad percibida de respuestas. En consultas temporales, esta ventaja se amplía a 83% (4,40 vs. 2,40), indicando que la integración de contextos temporalmente valiosos mejora significativamente la articulación de respuestas finales.
- **Descubrimiento de relaciones indirectas:** PP promedio de 0,88 saltos demuestra que el sistema opera con propagación selectiva, evitando la amplificación descontrolada de activación. Casos exitosos como la Consulta 1 (61 contextos recuperados con PP=1,48, manteniendo 71-100% de relevancia en niveles 0-2) validan la efectividad del algoritmo de propagación probabilística.

Limitaciones identificadas:

- **Ruido por propagación excesiva:** CIR promedio de 76,31% supera el umbral óptimo (20-40%), indicando que el sistema recupera demasiados contextos débilmente relacionados. Este problema se agrava en consultas estructurales (CIR=67,50%, PRC=48,28%) donde RAG demostró mayor precisión (PRC=70%). Consultas como la 10 y 12 exhibieron CIR>88% con PRC<18%, evidenciando que la propagación devuelve contextos con poca relación cuando el tema solicitado es inexistente o altamente específico.

- **Trade-off latencia-funcionalidad:** TR promedio de 10,14 s (AGPC) vs. 3,43 s (RAG) representa una latencia 3 veces mayor, aunque dentro del umbral aceptable establecido (<25 s). Esta diferencia se explica por: (1) procesamiento adicional de propagación en grafo, (2) cálculo de relevancia temporal y factor de refuerzo, (3) diferencias infraestructurales (ejecución local vs. Google Colab). El sistema mantiene viabilidad práctica al preservar tiempos de respuesta adecuados para usuarios finales.

- **Fallo compartido en búsquedas por referencias legales:** Consultas 8, 10 y 13 revelaron limitaciones compartidas por ambos sistemas en recuperación de identificadores legales específicos (ej.: "Ley 24013", "art. 246 LCT"). Los embeddings semánticos no capturan adecuadamente referencias alfanuméricas, requiriendo extracción de entidades legales como metadatos estructurados para resolución efectiva.

Optimización de Fragmentación Semántica: Contribución Crítica

Un hallazgo fundamental de la investigación fue el impacto de la estrategia de fragmentación en la calidad de recuperación contextual. El problema inicial (fragmentación excesiva generando contextos de 12-15 palabras) impedía la recuperación de información relevante, incluso cuando esta existía literalmente en la base de datos. La implementación de fragmentación adaptativa con contexto mínimo garantizado (50-300 palabras por fragmento) mejoró la identificación de casos de amparo de 0/8 (0%) a 8/8 (100%) en la misma consulta de prueba.

Este descubrimiento evidencia un principio frecuentemente subestimado cuando se desarrolla esta clase de prototipo hasta incluso cuando se utiliza RAG: **la calidad de la fragmentación semántica es determinante del rendimiento del sistema, frecuentemente más impactante que la elección del modelo de embeddings o la configuración del índice vectorial**. La literatura académica sobre RAG se enfoca predominantemente en arquitecturas de modelos generativos y estrategias de fusión de contextos, pero nuestros resultados demuestran que la fragmentación inadecuada puede degradar completamente el rendimiento, incluso con componentes técnicamente óptimos.

La solución implementada (acumulación de líneas de diálogo con umbral dual, preservación de unidades semánticas, fusión de fragmentos residuales) representa una contribución metodológica directamente transferible a otros sistemas, independientemente del dominio de aplicación. Los principios establecidos (contexto mínimo de 50 palabras, límite superior de 300 palabras, respeto por unidades, pregunta-respuesta) pueden servir como directrices iniciales para implementadores de sistemas similares.

Innovaciones Arquitectónicas y Técnicas

El AGPC introduce varias innovaciones arquitectónicas que lo diferencian cualitativamente de sistemas existentes:

1. Modelo adaptativo de peso efectivo según intención:

En lugar de aplicar una única fórmula fija, el sistema implementa estrategias diferenciadas:

- Consultas temporales: Pe = R_temporal × Fr × (1 + W_estructural)
- Consultas estructurales mixtas: Pe = W_estructural × (1 + R_temporal × Fr)

Esta adaptación dinámica permite que la dimensión temporal tenga papel predominante en preguntas temporales, mientras la similitud semántica mantiene su peso en consultas estructurales, logrando comportamiento más simétrico y coherente con la intencionalidad del usuario.

2. Detección automática de intención temporal con LLM:

La migración desde patrones estáticos hacia análisis con Gemini 2.0 Flash permite interpretación flexible de lenguaje natural ("mañana", "la próxima semana", "en agosto de 2023") e inferencia automática de ventanas temporales relevantes. Esta capacidad es crítica para el funcionamiento del factor de refuerzo adaptativo, ya que permite clasificar consultas temporales en diferentes niveles como fuerte, media, débil y atemporal, sin intervención manual.

3. Normalización sigmoidea de pesos probabilísticos:

La función Pe_norm = Pe_efectivo / (1 + Pe_efectivo) resuelve elegantemente el problema matemático de pesos efectivos que excedían 1.0, garantizando que todos los pesos sean interpretables como probabilidades condicionales. Esta normalización no solo proporciona consistencia matemática, sino que mejora estabilidad numérica en propagación iterativa y compatibilidad con algoritmos estándar de análisis de grafos.

4. Actualización incremental y optimizaciones de rendimiento:

Las optimizaciones implementadas (complejidad O(n) vs O(n²) original, batch processing de embeddings, cálculo vectorizado de similitudes, escritura diferida, caché de embeddings) no son meras mejoras técnicas, sino transformaciones fundamentales que habilitan la viabilidad

práctica del sistema. Sin estas optimizaciones, los tiempos de 90-120 minutos para cargar 50 conversaciones habrían limitado severamente las pruebas con el prototipo.

Interpretación Teórica: Memoria Dual Semántica-Temporal

Los resultados experimentales validan un modelo teórico de memoria dual en sistemas inteligentes. El AGPC implementa dos dimensiones de relevancia:

Dimensión estructural (indeleble): Captura relaciones semánticas y de co-ocurrencia que permanecen estables en el tiempo. Un contrato siempre mantiene relación con la factura asociada, independientemente de su antigüedad. Esta dimensión se modela mediante similitud de embeddings (W_estructural) y no decae temporalmente.

Dimensión temporal (circunstancial): Modela el balance de importancia según proximidad al momento actual. Un deadline es altamente relevante la semana previa, pero su importancia decrece exponencialmente tras su vencimiento. Esta dimensión se implementa mediante decaimiento exponencial (R_temporal) y factor de refuerzo (Fr).

La combinación de ambas dimensiones mediante la fórmula de peso efectivo adaptativo permite al sistema replicar comportamientos cognitivos humanos: recordamos por asociación semántica (primer filtro) y priorizamos lo temporalmente cercano o valioso (segundo filtro ajustable). Esta dualidad explica por qué el AGPC logra coherencia superior en consultas temporales (+83%) mientras mantiene precisión competitiva en consultas estructurales (PRC comparable a RAG).

El fracaso casi completo de RAG en cobertura temporal (CT=4,13%) no es un defecto de implementación, sino una limitación arquitectónica fundamental: RAG opera exclusivamente en la dimensión estructural, sin mecanismos para modelar, calcular o aplicar restricciones temporales. Esta observación sugiere que **la gestión temporal explícita no es una característica opcional, sino un requisito arquitectónico fundamental para sistemas que deben operar sobre información distribuida en el tiempo**.

Implicaciones para el Diseño de Sistemas RAG

Los hallazgos de este trabajo tienen implicaciones prácticas:

1. Fragmentación como componente crítico: La calidad de fragmentación semántica debe considerarse tan importante como la selección del modelo de embeddings. Directrices prácticas: (a) garantizar contexto mínimo de 50 palabras por fragmento, (b) preservar unidades semánticas completas (intercambios pregunta-respuesta), (c) fusionar fragmentos residuales menores a 20 palabras, (d) limitar fragmentos a 300 palabras para prevenir dilución semántica.

2. Temporalidad como dimensión ortogonal: Sistemas que operan sobre información distribuida temporalmente deben incorporar mecanismos explícitos de gestión temporal, no

solo como metadatos opcionales, sino como componente arquitectónico relevante. La estrategia de "filtros por fecha" binarios (incluye/excluye) es insuficiente; se requiere modelado continuo de relevancia temporal.

3. Trade-off precision vs recall en dominios críticos: En dominios donde la omisión de información puede tener consecuencias graves (legal, médico, financiero), optimizar exclusivamente para precisión (como hace RAG estándar) puede ser contraproducente. El AGPC demuestra que sacrificar precisión (42,64%) para maximizar recall (recuperación de 6/6 precedentes vs. 5/6 de RAG) puede ser preferible cuando el costo de perder información crítica supera el costo de procesar contextos con poca relación.

4. Explicabilidad mediante grafos auditables: La representación explícita de relaciones contextuales en grafos probabilísticos proporciona trazabilidad y explicabilidad qué sistemas RAG estándar no ofrecen. Esta capacidad es importante en aplicaciones reguladas donde la justificación de decisiones algorítmicas es requisito legal (por ejemplo, auditorías financieras).

Limitaciones Reconocidas y Trabajo Futuro

El presente trabajo reconoce sus limitaciones metodológicas y técnicas, estableciendo direcciones claras para investigación futura:

Limitaciones metodológicas:

- **Dataset sintético:** El uso de conversaciones generadas artificialmente, aunque permite control experimental y reproducibilidad, no captura completamente la variabilidad lingüística, ambigüedad y complejidad del lenguaje legal real. Validación con casos reales es necesaria antes de deployment productivo.
- **Dominio único de validación:** La evaluación se concentra en el dominio legal. Aunque la arquitectura es conceptualmente generalizable, validación empírica en múltiples dominios (médico, técnico, educativo) es necesaria para confirmar transferibilidad de principios establecidos.
- **Evaluación manual con alcance limitado:** Las métricas PRC y CR requieren evaluación humana (3-5 horas para 15 consultas), limitando la escala de experimentación. Desarrollo de métricas automáticas correlacionadas permitiría evaluación a mayor escala.

Limitaciones técnicas y direcciones futuras:

1. Propagación con poda semántica: El problema crítico de CIR excesivo (76,31%) requiere implementar validación pre-propagación: calcular similitud semántica entre consulta y nodo candidato, establecer umbral mínimo (ej: 0,3), propagar solo hacia vecinos que superen ambos umbrales (activación + relevancia semántica). Esta mejora mantendría beneficios de propagación mientras reduce el ruido.

2. Ajuste dinámico de profundidad: Implementar criterio de detención adaptativo que evalúe calidad promedio de contextos por nivel y detenga propagación cuando similitud cae bajo umbral definido. Esto optimizaría automáticamente el balance cobertura-precisión sin requerir configuración manual.

3. Extracción de entidades legales: Resolver fallos compartidos con RAG estándar en Consultas 8, 10, 13 mediante identificación y etiquetado de artículos, leyes, códigos como metadatos estructurados. Esto permitiría búsquedas exactas por referencias legales, independientemente de embeddings semánticos.

4. Escalabilidad a grafos masivos: Para producción con decenas de miles de nodos, migrar de NetworkX a Neo4j (consultas declarativas con Cypher, escalabilidad a grafos masivos). Implementar índices distribuidos (Redis) para escalabilidad horizontal y soporte multi-modelo de embeddings con reindexación incremental.

6. Soporte GPU opcional: Implementar aceleración por GPU para generación de embeddings, permitiendo mejoras de 10-50x en sistemas con hardware adecuado sin comprometer compatibilidad en entornos solo CPU.

7. Evaluación cross-domain: Validar principios establecidos en múltiples dominios (médico, financiero, educativo y técnico) para establecer directrices universales de fragmentación y configuración de parámetros.

Contribuciones al Campo de Investigación

El presente trabajo aporta las siguientes contribuciones al campo de gestión contextual y sistemas inteligentes:

1. Contribución teórica: Formalización matemática del Grafo Contextual Probabilístico (GCP) como estructura Gc = (V, E, Pv, Pe, φ) que integra similitud estructural, relevancia temporal y propagación de activación. Esta formulación proporciona base conceptual para futuras investigaciones en memoria contextual.

2. Contribución arquitectónica: Diseño e implementación de arquitectura modular en seis capas que demuestra viabilidad práctica de gestión probabilística de contextos. Las optimizaciones de rendimiento documentadas (actualización incremental, batch processing, normalización sigmoidea) son directamente transferibles a otros sistemas.

3. Contribución metodológica: Marco de evaluación reproducible que combina 4 métricas automáticas (TR, CT, PP, CIR) + 2 métricas manuales (PRC, CR) para validación sistemática de sistemas de gestión contextual. Esta metodología puede adoptarse o ser consideradas como benchmark en investigaciones futuras.

4. Contribución empírica: Demostración experimental de superioridad en consultas temporales (CT: 69,29% vs. 4,13%) con identificación precisa de trade-offs (precision vs.

recall, latencia vs. funcionalidad). Evidencia cuantitativa que válida la necesidad arquitectónica de gestión temporal explícita.

5. Contribución práctica: Identificación del impacto crítico de fragmentación semántica (mejora de 0% a 100% en recuperación).

Reflexión Final:

Los resultados de este trabajo sugieren que el futuro de los sistemas inteligentes no reside en ampliar indefinidamente las ventanas de contexto de los LLMs (aunque los modelos evolucionaron de 200K tokens (Claude 3.5 Sonnet, junio 2024) a 2M tokens (Gemini 2.5 Pro, octubre 2025) durante el desarrollo de esta investigación) sino en desarrollar arquitecturas de memoria externa que gestionen inteligentemente la información contextual.

La analogía con sistemas operativos es reveladora: así como un S.O. decide qué datos cargar en RAM (ventana de contexto) desde el almacenamiento persistente (memoria de largo plazo), el AGPC implementa un sistema de gestión que prioriza probabilísticamente qué contextos cargar para cada consulta específica. Esta aproximación es escalable porque la complejidad de decisión crece logarítmicamente (O(log n) con HNSW), no linealmente con el volumen de información.

El AGPC demuestra que un enfoque probabilístico, estructurado mediante grafos, y sensible al tiempo, no solo es técnicamente viable, sino empíricamente superior en escenarios donde la dimensión temporal es crítica. Más allá del dominio legal utilizado para validación, los principios arquitectónicos y hallazgos metodológicos son generalizables a cualquier dominio que requiera gestión de conocimiento distribuido en el tiempo: auditoría financiera, medicina, gestión de proyectos, educación (progresión de aprendizaje y conocimientos previos).

La integración de documentos PDF como fuente de contexto, implementada en versiones avanzadas del prototipo, representa un avance en la versatilidad del AGPC. Esta funcionalidad demuestra que el prototipo puede ir evolucionando o avanzando en capturar información de forma multimodal (imágenes, audio, documentos, entre otros.) para enriquecer más la estructura de datos.

En **conclusión**, el Agente de Gestión Probabilística de Contextos no es simplemente una mejora incremental sobre sistemas RAG existentes, sino un cambio de paradigma arquitectónico que introduce gestión explícita de temporalidad, descubrimiento de relaciones indirectas mediante propagación probabilística, y representación auditable de conocimiento mediante grafos estructurados. Los resultados experimentales validan la propuesta de valor mientras identifican claramente limitaciones y direcciones de trabajo futuro, estableciendo una base sólida para investigaciones que busquen evolucionar sistemas de gestión contextual hacia modelos más cercanos a la cognición humana: flexibles, adaptativos, temporalmente conscientes y explicables.

7. Referencia

Este capítulo consolida todas las fuentes de información consultadas para la fundamentación teórica, metodológica y técnica de la investigación. Las referencias se encuentran organizadas temáticamente, abarcando desde los fundamentos matemáticos de la teoría de grafos y modelos probabilísticos, hasta la documentación técnica más reciente sobre Modelos de Lenguaje de Gran Escala y sistemas de recuperación de información.

Anthropic. (2024). *Claude 3.5 Sonnet*. https://www.anthropic.com/news/claude-3-5-sonnet

Anthropic. (2025). *Claude Opus 4 and Claude Sonnet 4.5: Technical overview*. https://www.anthropic.com/news/claude-4-family

Bommarito, M., & Katz, D. M. (2024). GPT as knowledge worker: A zero-shot evaluation of (AI)CPA capabilities. *Journal of Artificial Intelligence Research*, *79*(1), 1-31. https://doi.org/10.1613/jair.1.14654

Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory*. Springer.

Chen, X., Zhang, Y., Wang, L., & Liu, H. (2024). Temporal graph neural networks for dynamic context modeling in conversational AI. En *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, *38*(10), 11234-11242. https://doi.org/10.1609/aaai.v38i10.29012

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3ra ed.). MIT Press.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171-4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Elasticsearch B.V. (2024). *Okapi BM25 similarity*. https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Google DeepMind. (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. https://deepmind.google/technologies/gemini/1.5/

Google DeepMind. (2025). *Gemini 2.0 and 2.5: Next generation multimodal models*. https://deepmind.google/technologies/gemini/

Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M.-W. (2020). REALM: Retrieval-augmented language model pre-training. En *Proceedings of the 37th International Conference on Machine Learning* (pp. 3929-3938). PMLR.

Harrison, C. (2024). *LangChain: Building applications with LLMs through composability*. https://docs.langchain.com

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, *7*(3), 535-547. https://doi.org/10.1109/TBDATA.2019.2921572

Katz, D. M., Bommarito, M. J., Gao, S., & Arredondo, P. (2023). GPT-4 passes the bar exam. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *381*(2251), 20220123. https://doi.org/10.1098/rsta.2022.0123

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

LangChain AI. (2025). Context engineering for agents: A framework for optimizing LLM performance. *LangChain Blog*. https://blog.langchain.com/context-engineering-for-agents/

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. En *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

Li, J., Chen, W., Zhang, X., & Wang, Y. (2024). Context management strategies in multi-turn dialogue systems: A systematic review. *Journal of Artificial Intelligence Research*, *79*(3), 245-289. https://doi.org/10.1613/jair.1.14788

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, *12*, 157-173. https://doi.org/10.1162/tacl_a_00638

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 824-836. https://doi.org/10.1109/TPAMI.2018.2889473

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Meta AI Research. (2024). *Llama 3: Open foundation and fine-tuned chat models*. https://ai.meta.com/blog/meta-llama-3/

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint* arXiv:1301.3781. https://arxiv.org/abs/1301.3781

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). A comprehensive overview of large language models. *arXiv preprint* arXiv:2307.06435. https://arxiv.org/abs/2307.06435

OpenAI. (2024a). *GPT-4 technical report*. *arXiv preprint* arXiv:2303.08774. https://arxiv.org/abs/2303.08774

OpenAI. (2024b). *GPT-4o: Omni-modal language model*. https://openai.com/index/hello-gpt-4o/

Packer, C., Fang, V., Patil, S. G., Wooders, K., & Gonzalez, J. E. (2023). MemGPT: Towards LLMs as operating systems. *arXiv preprint* arXiv:2310.08560. https://arxiv.org/abs/2310.08560

Peng, B., Galley, M., He, P., Cheng, H., Xie, Y., Hu, Y., Huang, Q., Lidén, L., Yu, Z., Chen, W., & Gao, J. (2023). Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint* arXiv:2302.12813. https://arxiv.org/abs/2302.12813

Perez, E., Ringer, S., Lukošiūtė, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C., Kundu, S., Kadavath, S., Jones, A., Chen, A., Mann, B., Israel, B., Seethor, B., McKinnon, C., Olah, C., Yan, D., Amodei, D., … & Kaplan, J. (2023). Discovering language model behaviors with model-written evaluations. En *Findings of the Association for Computational Linguistics: ACL 2023* (pp. 13387-13412). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-acl.847

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, agosto). DeepWalk: Online learning of social representations. En *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 701-710). ACM. https://doi.org/10.1145/2623330.2623732

Pinecone Systems, Inc. (2024). *What is RAG?* https://www.pinecone.io/learn/retrieval-augmented-generation/

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2023). Temporal relevance in knowledge-intensive NLP tasks. En *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 3456-3470). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.213

Re, R. M., & Solow-Niederman, A. (2019). Developing artificially intelligent justice. *Stanford Technology Law Review*, *22*(2), 242-289.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-Networks. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language*

Processing (EMNLP-IJCNLP) (pp. 3982-3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Robertson, S. E., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333-389. https://doi.org/10.1561/1500000019

Song, K., Tan, X., Qin, T., Lu, J., & Liu, T.-Y. (2020). MPNet: Masked and permuted pre-training for language understanding. En *Advances in Neural Information Processing Systems*, *33*, 16857-16867.

Sowa, J. F. (2014). *Principles of semantic networks: Explorations in the representation of knowledge*. Morgan Kaufmann.

Surden, H. (2019). Artificial intelligence and law: An overview. *Georgia State University Law Review*, *35*(4), 1305-1337.

Tanenbaum, A. S., & Bos, H. (2014). *Modern operating systems* (4ta ed.). Pearson Education.

Tong, H., Faloutsos, C., & Pan, J. Y. (2006, diciembre). Fast random walk with restart and its applications. En *Proceedings of the 6th International Conference on Data Mining* (pp. 613-622). IEEE. https://doi.org/10.1109/ICDM.2006.70

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. En *Advances in Neural Information Processing Systems*, *30*, 5998-6008.

Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., & Tang, Y. (2024). Graph-based memory systems for long-context language models. En *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Xu, W., Liu, X., Yan, Y., Peng, N., Su, H., Liu, T., & Huang, X. (2023). MemGPT: Towards LLMs with contextual long-term memory. *arXiv preprint* arXiv:2310.08560. https://arxiv.org/abs/2310.08560

Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., & Wei, F. (2023). Instruction tuning for large language models: A survey. *arXiv preprint* arXiv:2308.10792. https://arxiv.org/abs/2308.10792

Zhang, Y., & Liu, H. (2024). Probabilistic context graphs for multi-agent systems: A survey. *ACM Computing Surveys*, *56*(3), Artículo 67, 1-38. https://doi.org/10.1145/3626528

8. Anexos

En esta sección final se adjunta el material complementario y la evidencia documental que respalda el desarrollo y validación del Trabajo Final Agente de Gestión Probabilística de Contextos (AGPC). Se incluye el acceso al código fuente del prototipo, los manuales de usuario, la estructura del dataset legal sintético utilizado para las pruebas y los registros del protocolo de evaluación. Estos recursos permiten verificar la reproducibilidad de los experimentos y profundizar en los aspectos técnicos de la implementación.

A. Código Fuente Principal del Prototipo AGPC

Este anexo documenta las funciones principales del prototipo AGPC desarrollado. El código fuente está disponible en el repositorio de GitHub indicado más adelante, mientras que este documento proporciona una referencia de las funciones más relevantes del prototipo AGPC.

La implementación sigue una arquitectura de 6 capas que abarca desde registrar conversaciones hasta la generación de respuestas con LLM, pasando por procesamiento semántico, construcción del grafo probabilístico, análisis temporal y algoritmos de propagación de activación. Cada módulo está diseñado para operar de forma independiente y modular, facilitando su mantenimiento, y extensión futura.

La documentación siguiente detalla las funciones principales de cada módulo: Funciones Principales del Prototipo AGPC - Trabajo Final - De La Fuente Gonzalo - 01…

Link de prototipo AGPC en GitHub: https://github.com/GonzaloDeLaFuente3/agpc_prototipo

B. Manual de Usuario del Prototipo AGPC

El Manual de Usuario proporciona una guía completa para el uso del prototipo AGPC. Este documento está diseñado para usuarios finales.

Acceso al documento: Manual de Usuario del Prototipo AGPC - Trabajo Final - De La…

C. Dataset de Validación Legal

El dataset utilizado para la evaluación comparativa del sistema AGPC contra RAG estándar contiene 200 conversaciones sintéticas sobre casos legales, distribuidas temporalmente a lo largo del año 2023. El dataset está almacenado en formato JSON y sigue la siguiente estructura:

Estructura del Archivo JSON

El archivo raíz contiene un objeto con una única clave `"conversaciones"` que almacena un array de 200 objetos de conversación.

Estructura de cada conversación:

- **`titulo`** (string): Identificador descriptivo del caso legal (ej. "Incumplimiento de régimen de visitas")
- **`contenido`** (string): Texto completo del diálogo entre participantes, incluyendo preguntas, respuestas y asesoramiento legal.
- **`participantes`** (array de strings): Lista de roles que intervienen en la conversación (ej. ["Padre", "Abogado"], ["Trabajador", "Abogado"], ["Empresa", "Abogado"])
- **`fecha`** (string ISO 8601): Marca temporal de la conversación en formato `YYYY-MM-DDTHH:MM:SS`, distribuidas desde enero hasta agosto de 2023.

Características del Dataset

Total de conversaciones: 200

Período temporal: 2023-01-01 a 2023-08-31

Dominios legales cubiertos: Derecho laboral, derecho civil, derecho tributario, derecho de familia, derecho comercial

Tipos de casos incluidos:

- Despidos laborales y reclamos salariales
- Accidentes de trabajo y ART
- Incumplimientos contractuales
- Vicios ocultos en compraventas
- Moras administrativas y amparos
- Ejecuciones fiscales y recursos tributarios
- Divorcios y regímenes de visitas, entre otros.

Acceso al Dataset Completo

El dataset completo en formato JSON está disponible en el siguiente enlace:

legal_dataset_200.json

D. Protocolo de Evaluación

Link del documento sobre el Protocolo de Evaluación: PROTOCOLO DE EVALUACIÓN -Trabajo Final - De La Fuente Gonzalo - 01561

E. Consultas de Prueba y Resultados Detallados de la Evaluación

A continuación se adjuntan los documentos que respaldan la validación experimental del prototipo:

- **Link: Resultado de las consultas realizadas a los sistema AGPC y RAG estand…**
  Documento que contiene el registro detallado de los 15 experimentos comparativos. Detalla, consulta por consulta, los tiempos de respuesta, la trazabilidad de los contextos recuperados (directos vs. propagados), las respuestas textuales generadas por ambos sistemas y la evaluación manual cualitativa de relevancia y coherencia.
- **Link: Evaluacion de Metricas-Trabajo Final - De La Fuente Gonzalo - 01561**
  Archivo de análisis cuantitativo que apoya el cálculo de las métricas de desempeño (TR, PRC, CR, CT, PP, CIR). Incluye las tablas comparativas desglosadas por tipo de consulta (Temporal, Estructural y Mixta) y los promedios estadísticos utilizados para validar las hipótesis del trabajo.

F. Implementación del Sistema RAG Estándar

Código de la construcción e implementación del RAG estándar, con el cual se realizó la comparativa con el prototipo AGPC: Implementación RAG Estándar -Trabajo Final - De La Fuente Gonzalo - 01561

Link de RAG estándar en Google Colab: RAG-estandar.ipynb

G. Minutas de Reunión

Este anexo contendrá las minutas de las reuniones realizadas a lo largo del desarrollo del Trabajo Final, documentando el progreso y las decisiones clave que se tomaron.

Acceso al documento: Minutas de Reunion - Trabajo Final - De La Fuente Gonzalo - 0…

H. Glosario de términos técnicos

Para facilitar la lectura y comprensión del presente trabajo, a continuación se ofrece un compendio alfabético de los términos técnicos, acrónimos y conceptos especializados utilizados a lo largo del documento. Estas definiciones operativas buscan precisar el sentido específico con el que se emplean dichos términos en el contexto de la arquitectura propuesta y el dominio legal.

- **AGPC (Agente de Gestión Probabilística de Contextos)** Sistema inteligente desarrollado en este trabajo que administra información contextual mediante un Grafo Contextual Probabilístico, combinando técnicas de procesamiento de lenguaje natural, bases vectoriales y propagación probabilística para mejorar la recuperación de información distribuida temporalmente.
- **API (Application Programming Interface)** Interfaz de programación de aplicaciones que permite la comunicación entre diferentes componentes de software. En este trabajo, se utiliza para integrar servicios como Gemini (LLM) y FastAPI (servidor web).
- **Arista** En teoría de grafos, conexión entre dos nodos que representa una relación contextual. En el AGPC, cada arista tiene un peso probabilístico que combina similitud estructural y relevancia temporal.

- **Batch Processing** Procesamiento por lotes que agrupa múltiples operaciones similares para ejecutarlas simultáneamente, mejorando significativamente la eficiencia computacional. Implementado en el AGPC para generación de embeddings y cálculo de similitudes.
- **BERT (Bidirectional Encoder Representations from Transformers)** Modelo de lenguaje desarrollado por Google que utiliza transformers bidireccionales para generar representaciones contextuales de texto. Su versión base produce embeddings de dimensión 768.
- **BM25 (Best Matching 25)** Algoritmo clásico de recuperación de información basado en frecuencia de términos y longitud de documentos. Utiliza estadísticas léxicas para calcular relevancia, sin considerar semántica ni temporalidad.
- **ChromaDB** Base de datos vectorial de código abierto utilizada para almacenar e indexar embeddings. Implementa el algoritmo HNSW para búsqueda aproximada de vecinos más cercanos con complejidad O(log n).
- **CIR (Contextos Indirectos Recuperados)** Métrica que mide el porcentaje de contextos recuperados mediante propagación en el grafo (no por similitud directa).
- **Contexto** En el AGPC, fragmento atómico de información extraído de conversaciones o documentos. Cada contexto se representa como un nodo en el grafo con metadatos enriquecidos (título, timestamp, tipo, palabras clave).
- **CR (Coherencia de Respuesta)** Métrica manual de evaluación (escala 1-5) que mide la calidad, coherencia lógica y utilidad percibida de las respuestas generadas por el sistema desde la perspectiva del usuario final.
- **CT (Cobertura Temporal)** Métrica que evalúa la capacidad del sistema para recuperar información distribuida en el tiempo. Calcula el porcentaje de consultas que recuperan información de más de 30 días de antigüedad.
- **Decaimiento Exponencial** Función matemática implementada en el AGPC para modelar la pérdida de relevancia de información con el paso del tiempo.
- **Decaimiento Temporal** Proceso mediante el cual la relevancia de información disminuye progresivamente con el paso del tiempo. Permite priorizar eventos recientes sin descartar completamente información histórica.
- **D3.js (Data-Driven Documents)** Biblioteca JavaScript para crear visualizaciones interactivas de datos. Utilizada en el AGPC para generar representaciones gráficas del Grafo Contextual Probabilístico con funcionalidades de Zoom, pan y tooltips.
- **Embeddings** Representaciones vectoriales de texto en espacios de alta dimensionalidad que capturan significado semántico. Permiten medir similitud entre textos, aunque usen palabras diferentes.
- **Factor de Refuerzo (Fr)** Multiplicador adaptativo que ajusta la importancia de la dimensión temporal según la intención detectada en la consulta del usuario.
- **FastAPI** Framework web moderno de Python para construir APIs REST con alto rendimiento. Incluye validación automática de datos mediante Pydantic y soporte nativo para operaciones asíncronas.
- **Fragmentación Semántica** Proceso de dividir conversaciones o documentos extensos en unidades atómicas (contextos) manteniendo coherencia semántica.

- **GCP (Grafo Contextual Probabilístico)** Estructura matemática central del AGPC definida como Gc = (V, E, Pv, Pe, φ), donde V son nodos (contextos), E son aristas (relaciones), Pv son probabilidades de nodos, Pe son probabilidades de aristas, y φ es la función de embedding.
- **Gemini** Modelo de lenguaje desarrollado por Google DeepMind con capacidades multimodales. El AGPC utiliza Gemini 2.0 Flash para detección de intención temporal y generación de respuestas finales.
- **HNSW (Hierarchical Navigable Small World)** Algoritmo de búsqueda aproximada de vecinos más cercanos que organiza datos en capas jerárquicas. Permite búsquedas con complejidad O(log n) manteniendo 95% aproximadamente de precisión (recall).
- **Ingeniería de Contexto** Disciplina que estudia cómo seleccionar, aislar, comprimir y escribir información para optimizar el uso de la ventana de contexto en sistemas basados en LLMs. Estrategias: Write, Select, Compress, Isolate.
- **JSON (JavaScript Object Notation)** Formato ligero de intercambio de datos legible por humanos. Utilizado en el AGPC para persistir metadatos de nodos y configuraciones del sistema.
- **LangChain** Framework de código abierto para construir aplicaciones con LLMs que facilita la integración de memoria conversacional, recuperación de documentos y encadenamiento de operaciones.
- **Modelo de Lenguaje de Gran Escala (LLM):** Es un tipo de inteligencia artificial entrenada con enormes cantidades de texto para entender y generar lenguaje humano. Puede responder preguntas, escribir textos, resumir información, traducir, y mucho más. En otras palabras: es como un “superprocesador de texto” que sabe mucho por qué ha leído muchísimo y puede conversar como una persona.
- **MemGPT** Sistema que implementa jerarquía de memoria (corto/mediano/largo plazo) para LLMs, similar a la gestión de memoria en sistemas operativos. Carece de decaimiento temporal explícito y propagación probabilística.
- **MPNet (Masked and Permuted Pre-training)** Modelo de lenguaje desarrollado por Microsoft que combina enmascaramiento y permutación en el preentrenamiento. Versión base genera embeddings de 768 dimensiones optimizadas para similitud semántica.
- **NetworkX** Biblioteca Python para creación, manipulación y análisis de grafos complejos. Soporta grafos dirigidos/no dirigidos, atributos en nodos/aristas, y algoritmos de centralidad y clustering.
- **Nodo** En teoría de grafos, elemento fundamental que representa una entidad.
- **Normalización Sigmoidea** Transformación matemática f(x) = x/(1+x) que mapea valores positivos al intervalo (0,1). Implementada en el AGPC para garantizar que los pesos de aristas sean interpretables como probabilidades.
- **NotebookLM** Herramienta de Google que permite crear asistentes basados en documentos. A diferencia del AGPC, utiliza contexto cerrado sin actualización automática ni modelado explícito de relaciones probabilísticas.
- **PDF (Portable Document Format)** Formato de documento que preserva el diseño original independientemente del software o sistema operativo. El AGPC procesa PDFs como fuente de contextos mediante extracción de texto con PyPDF2.

- **Pickle** Módulo Python para serialización de objetos complejos. Utilizado en el AGPC para persistir la estructura del grafo NetworkX preservando todas sus propiedades topológicas.
- **PLN (Procesamiento de Lenguaje Natural)** Campo de inteligencia artificial que estudia la interacción entre computadoras y lenguaje humano. Incluye análisis sintáctico, extracción de entidades, embeddings y generación de texto.
- **PP (Profundidad de Propagación)** Métrica que mide el promedio de saltos en el grafo necesarios para encontrar contextos relevantes.
- **PRC (Precisión de Recuperación Contextual)** Métrica manual que calcula el porcentaje de contextos recuperados que son efectivamente útiles para responder la consulta.
- **Propagación de Activación** Algoritmo que permite descubrir relaciones indirectas en el grafo mediante difusión de relevancia desde nodos iniciales hacia vecinos.
- **PyPDF2** Biblioteca Python para manipulación de archivos PDF. Utilizada en el AGPC para extracción de texto página por página manteniendo estructura de párrafos (limitada a PDFs estructurados, no escaneados).
- **Pydantic** Biblioteca Python para validación de datos y configuración basada en type hints. Utilizada en FastAPI para definir esquemas de entrada/salida de la API del AGPC.
- **RAG (Retrieval-Augmented Generation):**Es una técnica que combina dos pasos: primero busca información significativa en una base de datos o documentos, y luego usa un modelo de lenguaje para generar una respuesta usando esa información. En otras palabras: es como si un asistente primero revisara sus notas o archivos para encontrar datos útiles, y después redactara la respuesta de forma natural y coherente.
- **Recall** En recuperación de información, proporción de elementos relevantes que fueron recuperados sobre el total de elementos relevantes existentes. Trade-off con precisión: mayor recall típicamente reduce precisión.
- **Sentence-Transformers** Biblioteca Python que adapta modelos transformer (como BERT) para generar embeddings de oraciones de calidad. El AGPC utiliza all-MiniLM-L6-v2 (384 dimensiones, balance calidad/eficiencia).
- **Similitud Coseno** Medida de similitud entre dos vectores que calcula el coseno del ángulo entre ellos. Rango [-1, 1], donde 1 indica vectores idénticos. Utilizada en el AGPC para comparar embeddings.
- **Similitud Estructural (W_estructural)** En el AGPC, combinación ponderada de similitud semántica (embeddings) y similitud léxica (keywords).
- **spaCy** Biblioteca Python de código abierto para procesamiento de lenguaje natural. Utilizada en el AGPC para extracción de palabras clave y análisis sintáctico de conversaciones.
- **SQLite** Sistema de gestión de bases de datos relacional embebido. Utilizado internamente por ChromaDB para persistencia de metadatos e índices vectoriales.
- **Timestamp** Marca temporal que registra el momento exacto en que ocurrió un evento. En el AGPC, cada nodo incluye timestamp en formato ISO 8601 para cálculos de decaimiento temporal.

- **Token** Unidad básica de texto procesada por modelos de lenguaje. Puede ser una palabra, parte de palabra o carácter. Los LLMs tienen ventanas de contexto limitadas, medidas en tokens (ej: 200K, 1M tokens).
- **TR (Tiempo de Respuesta)** Métrica que mide la latencia del sistema desde que recibe una consulta hasta que genera la respuesta completa.
- **Trade-off** Compromiso entre dos características deseables que no pueden optimizarse simultáneamente. Ejemplos en el AGPC: precisión vs. recall, latencia vs. funcionalidad, exactitud vs. eficiencia.
- **Transformer** Arquitectura de red neuronal basada en mecanismos de atención que procesa secuencias completas simultáneamente. Base de modelos como BERT, GPT y Gemini. Introducida por Vaswani et al. (2017).
- **Umbral de Activación** Valor mínimo de energía que un nodo debe superar durante propagación para ser considerado relevante.
- **Umbral de Similitud** Valor mínimo de similitud estructural requerido para crear una arista entre dos nodos en el GCP.
- **UUID (Universally Unique Identifier)** Identificador único de 128 bits representado como cadena hexadecimal. Utilizado en el AGPC para asignar identificadores únicos a nodos garantizando no duplicación.
- **Uvicorn** Servidor ASGI de alto rendimiento para Python. Ejecuta la aplicación FastAPI del AGPC con soporte para concurrencia mediante async/await, mejorando latencia en operaciones I/O-bound.
- **Vector Store** Base de datos optimizada para almacenar y buscar vectores de alta dimensionalidad (embeddings). Ejemplos: ChromaDB, FAISS, Pinecone. Permite búsquedas semánticas eficientes mediante índices especializados.
- **Ventana de Contexto** Cantidad máxima de tokens que un LLM puede procesar simultáneamente.
- **Ventana Temporal** En el AGPC, intervalo de tiempo identificado como significativo para una consulta. El sistema infiere automáticamente ventanas temporales mediante análisis con LLM (ej: "mañana" → fecha específica).
- **Visualización Multinivel** Capacidad del AGPC de representar el grafo en diferentes niveles de abstracción: macro (conversaciones completas), micro (todos los fragmentos y contextos), micro-filtrado (fragmentos de una conversación específica).
- **Peso Efectivo (Pe)** En el AGPC, valor numérico que cuantifica la fuerza de la relación entre dos nodos, considerando similitud estructural, relevancia temporal y factor de refuerzo.