La Guía Empresarial de Modelos de Lenguaje Pequeños (SLM) Locales: Del Hardware a los Flujos de Trabajo de Alto Valor

Sección 1: El Paradigma de los SLM: Redefiniendo la Accesibilidad y Eficiencia de la IA

Esta sección fundamental establece qué son los Modelos de Lenguaje Pequeños (SLM, por sus siglas en inglés), por qué representan un cambio estratégico significativo frente a la narrativa dominada por los Modelos de Lenguaje Grandes (LLM), y las tecnologías centrales que los hacen posibles. El objetivo es proporcionar al lector un marco conceptual sólido antes de profundizar en los detalles técnicos.

1.1. Más Allá de la Exageración: Definiendo el Modelo de Lenguaje Pequeño (SLM)

Un Modelo de Lenguaje Pequeño (SLM) es un modelo de inteligencia artificial, típicamente basado en la arquitectura Transformer, con un recuento de parámetros que varía desde unos pocos millones hasta aproximadamente 10-13 mil millones.[1] Esto contrasta marcadamente con los Modelos de Lenguaje Grandes (LLM), que pueden tener cientos de miles de millones o incluso billones de parámetros.[1] A pesar de su tamaño reducido, los SLM conservan capacidades fundamentales de Procesamiento del Lenguaje Natural (PLN) como la generación de texto, el resumen, la traducción y la respuesta a preguntas, lo que los hace altamente versátiles para una amplia gama de aplicaciones.[1]

El término "pequeño" es relativo y ha generado debate en la comunidad. Un modelo con mil millones de parámetros no es objetivamente pequeño, pero es compacto y eficiente en comparación con los LLM. Se ha propuesto el término "Modelo de Lenguaje Grande Pequeño" (Small Large Language Model), pero "SLM" se ha convertido en el término aceptado en la industria para denotar esta clase de modelos más ágiles.[1] Los SLM a menudo se derivan de los LLM, actuando como sus contrapartes más eficientes y especializadas.[3] La innovación fundamental detrás de los SLM no es la búsqueda de la inteligencia artificial general, sino hacer que la inteligencia artificial sea más accesible, asequible y eficiente para las tareas cotidianas.[8]

1.2. El Caso de Negocio para los SLM: Analizando las Ventajas Estratégicas de la Implementación Local

La creciente adopción de los SLM en el entorno empresarial está impulsada por un conjunto de ventajas estratégicas claras, especialmente cuando se implementan localmente en la infraestructura de una organización.

- **Rentabilidad**: Los SLM requieren significativamente menos potencia computacional para el entrenamiento, el ajuste fino (fine-tuning) y la inferencia. Esto reduce drásticamente los costos de infraestructura y operativos en comparación con los LLM, que a menudo exigen clústeres de GPU a gran escala.[1] Para las empresas, esto convierte a los SLM en una opción pragmática y económicamente viable para integrar la IA en sus operaciones.[12]
- **Velocidad y Baja Latencia**: Su tamaño más pequeño se traduce en una inferencia más rápida, es decir, generan respuestas con mayor celeridad. Esta característica es ideal para aplicaciones en tiempo real como chatbots interactivos, asistentes virtuales y herramientas de análisis de datos donde las respuestas inmediatas son cruciales para la experiencia del usuario y la eficiencia del flujo de trabajo.[1]
- **Privacidad y Seguridad Mejoradas**: Ejecutar SLM localmente (on-premise) elimina la necesidad de enviar datos corporativos sensibles a API de terceros. Esto otorga a las organizaciones un control total sobre su información, reduce drásticamente el riesgo de fugas de datos y simplifica el cumplimiento de regulaciones de privacidad como el GDPR o la CCPA.[1]
- **Accesibilidad y Capacidad sin Conexión**: Una de las ventajas más transformadoras de los SLM es su capacidad para ejecutarse en hardware de consumo, incluyendo ordenadores portátiles, teléfonos móviles y dispositivos de borde (edge devices).[1] Esto no solo democratiza el acceso a herramientas de IA potentes, sino que también permite la funcionalidad sin conexión, garantizando la continuidad del negocio en entornos con conectividad limitada o nula.[13]

La capacidad de ejecutar estos modelos en hardware local no es simplemente una característica técnica; representa un cambio de paradigma fundamental. Mientras que los LLM consolidaron la idea de la IA como un servicio masivo y centralizado en la nube, ofrecido por unos pocos gigantes tecnológicos, los SLM permiten un modelo descentralizado. En este nuevo modelo, la IA se convierte en una utilidad local, similar a una hoja de cálculo o un procesador de textos, que se ejecuta directamente en el dispositivo del usuario. Esta transición de la IA como un "servicio alquilado" a la IA como una "herramienta propia" tiene profundas implicaciones para la autonomía de los datos, la resiliencia operativa y fomenta un nuevo ecosistema de aplicaciones en el dispositivo que son inherentemente más rápidas y privadas.

1.3. Comprendiendo las Concesiones: Una Mirada Clara a las Limitaciones de los SLM

A pesar de sus numerosas ventajas, es crucial que las organizaciones comprendan las limitaciones inherentes de los SLM para establecer expectativas realistas y seleccionar la herramienta adecuada para cada tarea.

- **Alcance de Conocimiento Limitado**: Los SLM se entrenan con conjuntos de datos más pequeños y, a menudo, específicos de un dominio. Si bien esto los convierte en expertos en un nicho, tienen dificultades con tareas de conocimiento general que quedan fuera de su ámbito de entrenamiento.[1] Sacrifican la amplitud del conocimiento por la profundidad en un área específica.[12]
- **Riesgo de Amplificación de Sesgos**: Debido a que se entrenan con conjuntos de datos más pequeños, cualquier sesgo presente en esos datos puede ser amplificado en el comportamiento del modelo. Es esencial una curación de datos meticulosa y responsable para mitigar este riesgo.[1]
- **Complejidad y Razonamiento Reducidos**: Los SLM pueden flaquear en tareas de razonamiento de varios pasos o altamente matizadas que requieren la comprensión contextual profunda y amplia inherente a los modelos más grandes.[1] También pueden tener dificultades para mantener el contexto en conversaciones muy largas.[10]
- **Restricciones de Escalabilidad**: Aunque son eficientes, expandir las capacidades de un único SLM a nuevos dominios puede ser un desafío y puede impactar negativamente su rendimiento y velocidad.[10]

Curiosamente, la limitación del "alcance limitado" se transforma en una fortaleza dentro de arquitecturas de IA más avanzadas. El auge de la IA Agéntica, donde múltiples agentes de IA especializados colaboran para resolver problemas complejos, se ve directamente impulsado por los SLM.[12] En lugar de depender de un LLM monolítico que intenta hacerlo todo, una organización puede desplegar una flota de agentes SLM rentables y especializados: uno para procesar facturas, otro para clasificar tickets de soporte, y un tercero para generar código. Este enfoque modular es más escalable, manejable y, en última instancia, más potente que depender de un único modelo generalista. El futuro de la IA empresarial no reside en un cerebro gigante, sino en un equipo coordinado de especialistas SLM.

1.4. De LLM a SLM: Una Introducción a las Técnicas de Compresión Fundamentales

La creación de SLM eficientes a partir de modelos fundacionales más grandes se basa en un conjunto de técnicas de compresión sofisticadas. Comprender estos métodos es clave para apreciar cómo los SLM logran su equilibrio entre tamaño y rendimiento.

- **Destilación de Conocimiento (Knowledge Distillation)**: Este paradigma, conocido como "maestro-alumno", utiliza un LLM grande y potente (el maestro) para entrenar a un SLM más pequeño (el alumno). El alumno aprende a imitar las salidas y los patrones

de razonamiento del maestro, condensando eficazmente el conocimiento del modelo más grande en una forma más compacta. Esta es una manera rentable de crear modelos expertos especializados sin el costo prohibitivo de entrenarlos desde cero.[3]

- **Poda (Pruning)**: Esta técnica consiste en eliminar sistemáticamente parámetros (pesos, neuronas o incluso capas enteras) redundantes o menos importantes de una red neuronal entrenada. El objetivo es reducir el tamaño del modelo y la complejidad computacional sin afectar significativamente su rendimiento.[3]
- **Cuantización (Quantization)**: Esta es una optimización crítica que reduce la precisión numérica de los pesos del modelo (por ejemplo, de números de punto flotante de 32 bits a enteros de 8 o 4 bits). Este proceso reduce drásticamente la huella de memoria del modelo (VRAM) y puede acelerar significativamente la inferencia, haciendo posible ejecutar modelos potentes en hardware menos capaz.[3]

Sección 2: Arquitectura de su Entorno de IA Local: Requisitos de Hardware y Software

Esta sección proporciona una guía detallada y práctica sobre los prerrequisitos técnicos para ejecutar SLM localmente. Se centra en gran medida en la restricción de hardware más crítica —la VRAM de la GPU— y describe la pila de software necesaria.

2.1. El Papel Central de la VRAM: Una Guía Detallada de los Requisitos de la GPU

Para la inferencia local de SLM, la especificación de hardware más crítica es la cantidad de Memoria de Acceso Aleatorio de Video (VRAM) en la Unidad de Procesamiento Gráfico (GPU). Para un procesamiento eficiente, el modelo completo (o la mayor parte de él) debe cargarse en la VRAM.[14] La cantidad de VRAM disponible se convierte, por tanto, en el principal factor limitante que determina qué modelos se pueden ejecutar y con qué rendimiento.

La cuantización es la palanca clave para adaptar modelos potentes a hardware de consumo. Existe una relación directa entre el nivel de cuantización y el uso de VRAM: un modelo cuantizado a 4 bits requiere aproximadamente la mitad de VRAM que una versión de 8 bits, y una octava parte de una versión de 16 bits (FP16/BF16).[15] Los datos muestran claramente que los modelos de precisión completa como Gemma 7B (que requiere ~15.9 GB de VRAM) [15] o Phi-3 Medium (estimado en ~28 GB) están fuera del alcance de la mayoría de las GPU de consumo, que suelen tener entre 8 GB y 12 GB de VRAM. Sin embargo, la cuantización a 4 bits reduce estos mismos modelos a ~4.0 GB y ~7.9 GB respectivamente.[15] Esto no es solo una optimización de rendimiento; es la tecnología fundamental que hace viable la IA local en hardware de consumo. Esto implica que los avances en algoritmos de cuantización son ahora tan importantes como los avances en la arquitectura de los modelos para impulsar la adopción masiva de la IA.

La siguiente tabla resume los requisitos de VRAM para los SLM más populares en diferentes niveles de precisión, sirviendo como una guía práctica para la planificación de hardware.

Tabla 1: Requisitos de VRAM para SLM Populares (Precisión Completa y Cuantizados)

Familia del Modelo	Parámetros	VRAM Precisión Completa (FP16/BF16)	VRAM Cuantizado 8-bit (INT8)	VRAM Cuantizado 4-bit (INT4)	GPU de Consumo Recomendada
Phi-3 Mini	3.8B	~9.1 GB [19]	~5.1 GB [19]	~3.1 GB [19]	NVIDIA RTX 3050 (8GB) / 3060 (12GB)
Gemma 3	4B	~9.2 GB [16]	~4.6 GB (est.)	~2.6 GB [16]	NVIDIA RTX 3050 (8GB)
Gemma 2	7B/9B	~15.9 GB [15]	~8.0 GB [15]	~4.0 GB [15]	NVIDIA RTX 3060 (12GB) / 4060 (8GB)
Llama 3	8B	~16 GB (est.)	~8 GB (est.)	~4.5 GB (est.)	NVIDIA RTX 3060 (12GB) / 4060 (8GB)
Qwen 2	7B	~14 GB (est.)	~7 GB (est.)	~4.1 GB [20]	NVIDIA RTX 3060 (12GB) / 4060 (8GB)
Phi-3 Medium	14B	~28 GB (est.)	~14 GB (est.)	~7.9 GB [18]	NVIDIA RTX 3090/4090 (24GB)
Gemma 3	12B	~27.6 GB [16]	~13.8 GB (est.)	~6.9 GB [16]	NVIDIA RTX 3060 (12GB)

2.2. Perfiles de Hardware: De GPU de Consumo a Configuraciones Solo con CPU

La elección del hardware dependerá del presupuesto, los casos de uso previstos y la infraestructura existente.

- **GPU de Consumo**: Este es el escenario más común para ejecutar SLM localmente. Las tarjetas de la serie RTX de NVIDIA (series 30 y 40) son opciones excelentes. Modelos como la RTX 3060 (12GB), la RTX 3090/4090 (24GB) y las GPU para portátiles como la RTX 4060 (8GB) ofrecen puntos de entrada muy capaces para una amplia gama de SLM cuantizados.[16]
- **Estaciones de Trabajo Profesionales**: Para tareas más exigentes como el ajuste fino (fine-tuning) o la ejecución de modelos más grandes sin cuantización, las GPU profesionales como la NVIDIA A100, A6000 o H100 son la opción preferida. Ofrecen

capacidades de VRAM mucho mayores (40-80 GB) y un rendimiento computacional superior.[14]

- **Inferencia Solo con CPU**: Es factible ejecutar SLM únicamente en la CPU, aunque es significativamente más lento que la inferencia en GPU. Esta opción es viable para tareas que no son en tiempo real. El rendimiento de motores de inferencia como llama.cpp en CPU modernas (Intel Core Ultra, AMD Ryzen) y procesadores basados en ARM (AWS Graviton, Apple M-series) es notablemente bueno, ya que están altamente optimizados para este propósito.[22]

La viabilidad de la inferencia en CPU demuestra una conexión causal crucial: la optimización del software puede redefinir la relevancia del hardware existente. Aunque la narrativa de la industria se centra en la GPU, un software altamente optimizado como llama.cpp está convirtiendo la inferencia solo con CPU en una realidad práctica para muchas tareas con SLM. Esto implica que las organizaciones pueden comenzar a experimentar con SLM en su hardware de servidor o en los portátiles de sus empleados sin una inversión masiva e inmediata en GPU, lo que reduce significativamente la barrera de entrada para la experimentación con IA.

2.3. La Pila de Software: Herramientas Esenciales para la Implementación

Una configuración local exitosa requiere una pila de software específica diseñada para la eficiencia y la facilidad de uso.

- **Bibliotecas Centrales**: La base de cualquier configuración local incluye Python como lenguaje de programación, la biblioteca transformers de Hugging Face para un acceso sencillo a los modelos, y PyTorch como el framework de aprendizaje profundo subyacente.[18]
- **Motores de Optimización**: El componente crítico para el rendimiento es un motor de inferencia basado en C++ como llama.cpp. Herramientas populares como Ollama y LM Studio utilizan llama.cpp como su backend.[25]
  llama.cpp está específicamente diseñado para una inferencia eficiente en CPU y GPU, soportando varios métodos de cuantización (como el formato GGUF) y aceleraciones de hardware.[22]
- **Bibliotecas de Soporte**: Para tareas avanzadas como el ajuste fino en hardware de consumo, bibliotecas como bitsandbytes (para cuantización y optimizadores de bajo consumo de memoria) y PEFT (Parameter-Efficient Fine-Tuning) son esenciales.[26]

Sección 3: Guía Práctica para la Instalación y Gestión de SLM Locales

Esta sección proporciona instrucciones prácticas y paso a paso para desplegar SLM utilizando las herramientas más populares y fáciles de usar, pasando de la teoría a la aplicación práctica.

3.1. Método 1: El Enfoque Sencillo con Ollama

Ollama es una herramienta de código abierto que simplifica drásticamente el proceso de descarga, gestión y ejecución de SLM a nivel local. Funciona a través de una sencilla interfaz de línea de comandos y se ejecuta como un servicio en segundo plano, gestionando la complejidad de la configuración del modelo por el usuario.[1]

- **Instalación**:
  - **macOS**: Descargue la aplicación desde el sitio web oficial o instálela a través de Homebrew con brew install ollama.
  - **Windows**: Descargue y ejecute el instalador OllamaSetup.exe desde el sitio web oficial.
  - **Linux**: Utilice el script de instalación proporcionado en el sitio web: curl -fsSL https://ollama.com/install.sh | sh.
  - Se puede verificar la instalación ejecutando ollama -v en el terminal.[20]
- **Comandos Centrales**:
  - **Descargar un modelo**: ollama pull <nombre_del_modelo> (p. ej., ollama pull phi3). Este comando descarga el modelo desde el registro de Ollama a su máquina local.[28]
  - **Ejecutar un modelo**: ollama run <nombre_del_modelo> (p. ej., ollama run phi3). Este comando inicia una sesión de chat interactiva con el modelo especificado. Si el modelo no está presente localmente, lo descargará automáticamente antes de ejecutarlo.[1]
  - **Listar modelos locales**: ollama list muestra todos los modelos que ha descargado.
  - **Eliminar un modelo**: ollama rm <nombre_del_modelo> elimina un modelo de su máquina para liberar espacio.[20]
- **Personalización con Modelfile**: Ollama permite una personalización avanzada a través de un archivo llamado Modelfile. Este archivo de texto simple permite definir el comportamiento de un modelo, como establecer un mensaje de sistema (system prompt), ajustar parámetros como la temperatura (creatividad), o incluso importar un modelo personalizado en formato GGUF desde un archivo local.[20]

3.2. Método 2: El Flujo de Trabajo Guiado por la Interfaz Gráfica con LM Studio

LM Studio es una aplicación de escritorio que proporciona una interfaz gráfica de usuario (GUI) para descubrir, descargar y ejecutar SLM locales. Es especialmente adecuada para

usuarios que prefieren una interfaz visual en lugar de la línea de comandos.[29]

- **Instalación y Configuración**: El proceso comienza con la descarga e instalación de la aplicación LM Studio desde su sitio web oficial.[29]
- **Flujo de Trabajo**:
    1. **Descubrir y Descargar**: La pestaña "Discover" (o "Home") permite buscar modelos en Hugging Face. Es crucial buscar modelos en formato GGUF, que están optimizados para la inferencia local. Los modelos se pueden descargar directamente dentro de la aplicación con un solo clic.[29]
    2. **Chatear y Configurar**: En la pestaña "Chat", se puede cargar un modelo en la memoria. El panel derecho ofrece opciones de configuración detalladas, como el número de capas que se descargarán en la GPU (GPU offloading) para acelerar la inferencia, la longitud del contexto y otros parámetros del modelo.[29]
    3. **Servidor de Inferencia Local**: Una de las características más potentes de LM Studio es su servidor de inferencia local incorporado. Al iniciarlo desde la pestaña "Local Server", la aplicación expone una API compatible con OpenAI. Esto es clave para integrar el SLM local con otras aplicaciones, scripts y herramientas de desarrollo que ya están diseñadas para funcionar con la API de OpenAI.[30]

La existencia de herramientas como Ollama y LM Studio no es solo una conveniencia; es una capa de abstracción crítica. Anteriormente, ejecutar un modelo local requería gestionar entornos de Python, clonar repositorios de GitHub y ejecutar scripts complejos. Estas herramientas abstraen toda esa complejidad en simples comandos como ollama run o unos pocos clics en una GUI. Este patrón, visto en otras olas tecnológicas como Docker, está expandiendo la audiencia principal de la IA local desde ingenieros de IA/ML a una base mucho más amplia de desarrolladores de software e incluso usuarios avanzados no técnicos. Esta abstracción es el catalizador que impulsará la adopción generalizada en la empresa.

Además, la estandarización en torno a una API compatible con OpenAI [30] es una tendencia sutil pero profundamente importante. Significa que los desarrolladores pueden construir aplicaciones que apunten a la API de OpenAI y, con un simple cambio de la base_url, alternar entre un potente modelo en la nube como GPT-4 y un SLM local y privado como Phi-3. Esto reduce drásticamente la fricción del desarrollo y el despliegue, permitiendo sistemas de IA híbridos donde se puede prototipar con una API en la nube y desplegar con un modelo local por razones de privacidad y costo, todo sin cambiar el código de la aplicación.

3.3. Configuración Avanzada: Optimizando el Rendimiento

Para extraer el máximo rendimiento de una configuración local, se pueden ajustar varios parámetros clave.

- **Descarga a la GPU (GPU Offloading)**: Tanto Ollama como LM Studio permiten descargar un número específico de capas del modelo a la VRAM de la GPU. Esto permite a los usuarios con VRAM limitada acelerar una porción del modelo en la GPU, mientras que el resto se ejecuta en la CPU, logrando un equilibrio entre velocidad y limitaciones de hardware.

- **Gestión de Hilos de la CPU**: Para cargas de trabajo pesadas en la CPU, el proyecto llama.cpp aconseja que, para un rendimiento óptimo, el número de hilos debe establecerse igual al número de núcleos de CPU *físicos*, no lógicos. Este es un parámetro de ajuste crítico para maximizar el rendimiento de la CPU.[25]
- **Elección de la Cuantización Correcta**: Al descargar modelos de Hugging Face, es fundamental seleccionar el archivo de cuantización adecuado. Los archivos con sufijos como Q4_K_M o Q5_K_M suelen ofrecer el mejor equilibrio entre rendimiento (velocidad y bajo uso de memoria) y precisión. Los valores de Q más bajos son más rápidos pero pueden degradar la calidad de la respuesta.[25]

Sección 4: Selección del SLM Adecuado para Tareas Administrativas, Financieras y Comerciales

Esta sección pasa del "cómo" al "qué", proporcionando recomendaciones de modelos específicos adaptadas a las necesidades empresariales declaradas por el usuario. Se basa en un análisis comparativo detallado de los principales candidatos de SLM.

4.1. Evaluación de los Modelos Líderes: Un Análisis Comparativo

Antes de asignar modelos a tareas específicas, es útil comprender las fortalezas y características distintivas de las principales familias de SLM disponibles.

- **Familia Phi-3 de Microsoft**: Conocida por su sólido rendimiento en razonamiento, matemáticas y codificación a pesar de su pequeño tamaño. Este rendimiento se logra a través de datos de entrenamiento de muy alta calidad, descritos como "similares a libros de texto". El modelo Phi-3-mini (3.8B) es un destacado por su excepcional relación rendimiento-tamaño.[1]
- **Familia Gemma de Google**: Construidos con la misma tecnología que los modelos más grandes Gemini, los modelos Gemma (p. ej., 2B, 4B, 9B) son equilibrados, multimodales (Gemma 3) y se integran perfectamente con los ecosistemas de Google y Hugging Face.[1]
- **Meta Llama 3 (8B)**: La versión más pequeña de la popular familia Llama 3, este modelo es un potente generalista, ajustado por instrucciones para el diálogo y conocido por su sólido rendimiento en una amplia gama de benchmarks.[39]
- **Familia Qwen2 de Alibaba**: Una serie versátil de modelos con fuertes capacidades multilingües y de razonamiento en contextos largos. Las variantes más pequeñas (0.5B, 1.5B, 7B) son altamente eficientes y tienen una licencia comercialmente permisiva (Apache 2.0).[1]

La siguiente tabla proporciona una visión estratégica que permite a un responsable de la toma de decisiones comparar rápidamente los modelos basándose en criterios relevantes para el negocio, en lugar de solo en benchmarks brutos.

Tabla 2: Análisis Comparativo de los Principales SLM para Tareas Empresariales

Característica	**Phi-3-mini (3.8B)**	**Gemma 3 (4B)**	**Llama 3 (8B)**	**Qwen 2 (7B)**
Fortaleza Principal	Razonamiento fuerte, Lógica, Código [32]	Multimodalidad, Multilingüe [35]	Generalista de alta calidad, Diálogo [39]	Multilingüe, Contexto largo [39]
Ventana de Contexto	128K tokens [18]	128K tokens [35]	8K (ampliable con ajuste fino)	32K+ (hasta 128K en algunas versiones) [40]
Multilingüe	Bueno, pero entrenado principalmente en inglés [32]	Excelente, >140 idiomas [35]	Bueno [39]	Excelente [39]
Multimodalidad	Modelo de visión disponible (Phi-3.5-Vision) [33]	Sí (entrada de Imagen + Texto) [35]	Modelo de visión disponible en la familia Llama	Modelo de visión disponible (Qwen-VL) [27]
Licencia Comercial	Licencia MIT (Sí) [41]	Términos de Uso de Gemma (Sí)	Licencia Llama 3 (Sí)	Apache 2.0 (Sí) [39]
Mejor para...	Tareas con mucha lógica, análisis de datos, generación de código	Chatbots, creación de contenido con imágenes, aplicaciones globales	Chatbots de alta calidad, asistentes de escritura	Tareas empresariales en múltiples idiomas

4.2. Flujos de Trabajo Administrativos: Gestión de Documentos y Comunicaciones Internas

- **Tareas**: Resumir informes largos o transcripciones de reuniones, redactar borradores de correos electrónicos, responder a consultas sobre una base de conocimientos interna.
- **Requisitos Clave**: Sólida comprensión del lenguaje, una ventana de contexto larga y capacidad para seguir instrucciones.
- **Recomendaciones de Modelos**:
    - **Phi-3-mini-128k**: Una elección excelente debido a su masiva ventana de contexto de 128K tokens, que le permite procesar documentos muy grandes en una sola pasada.[32] Su fuerte capacidad de razonamiento ayuda a crear resúmenes coherentes y precisos.
    - **Qwen2 (7B o superior)**: También es un fuerte contendiente con sus capacidades de contexto largo y su robusta comprensión del lenguaje.[39]

4.3. Flujos de Trabajo Financieros: Extracción y Análisis de Datos

- **Tareas**: Extraer datos estructurados de facturas, recibos y estados financieros; clasificar transacciones; analizar informes para obtener métricas clave.
- **Requisitos Clave**: Alta precisión, capacidad para manejar documentos estructurados y semi-estructurados, y potencialmente capacidades de visión (para documentos escaneados).
- **Recomendaciones de Modelos**:
    - **Modelos de Visión-Lenguaje (VLM) con Ajuste Fino**: En este dominio, un modelo de texto genérico es insuficiente. El mejor enfoque es tomar un VLM como **Qwen-VL** [27] o
      Phi-3.5-Vision [34] y realizar un ajuste fino (fine-tuning) con un conjunto de datos etiquetado de las facturas de la propia empresa. Estos modelos pueden comprender el diseño y extraer información directamente de las imágenes, evitando los frágiles pasos de OCR.[27]
    - **NuExtract 1.5**: Un modelo especializado, ajustado a partir de Phi-3.5-mini específicamente para la extracción de información estructurada, lo que demuestra el poder del ajuste fino específico del dominio.[43]

4.4. Flujos de Trabajo Comerciales: Soporte al Cliente e Inteligencia de Ventas

- **Tareas**: Potenciar chatbots de soporte al cliente, realizar análisis de sentimiento sobre los comentarios de los clientes, resumir llamadas de ventas.
- **Requisitos Clave**: Fuerte habilidad conversacional, fiabilidad (baja alucinación) y la capacidad de integrarse con fuentes de datos en tiempo real.
- **Recomendaciones de Modelos**:
    - **Llama 3 (8B)** o **Gemma 3 (4B)**: Ambos son excelentes opciones para construir chatbots debido a su fuerte capacidad para seguir instrucciones y sus habilidades de diálogo.[39]
    - **El Enfoque RAG**: Para este caso de uso, la elección del SLM base es secundaria a la arquitectura. La solución más efectiva será un sistema de **Generación Aumentada por Recuperación (RAG)**. Cualquiera de los SLM recomendados puede servir como el "generador", pero deben estar conectados a una base de conocimientos de manuales de productos, tickets de soporte pasados y políticas de la empresa para proporcionar respuestas precisas y actualizadas.[45]

El análisis de estos casos de uso revela una conclusión crítica: simplemente elegir el "mejor" SLM no es suficiente. Para la extracción de facturas, un modelo de texto genérico fracasará; se requiere un VLM con ajuste fino.[27] Para el soporte al cliente, un SLM independiente alucinará; una arquitectura RAG es esencial para la precisión.[45] La implicación más amplia es que las empresas deben cambiar su enfoque de la "selección de modelos" a la "arquitectura

de soluciones". El valor se crea no por el modelo en bruto, sino por cómo se integra en un sistema más grande con pipelines de datos especializados (ajuste fino) o recuperación de datos en tiempo real (RAG).

Sección 5: Personalización Avanzada: Adaptando los SLM a sus Datos Propietarios

Esta sección detalla las dos estrategias principales para adaptar un SLM de propósito general al conocimiento y los procesos específicos de una empresa, proporcionando un marco para decidir cuándo utilizar cada enfoque.

5.1. Estrategia 1: Ajuste Fino (Fine-Tuning) para la Especialización en un Dominio

El ajuste fino es el proceso de continuar el entrenamiento de un modelo pre-entrenado en un conjunto de datos más pequeño y específico de un dominio. Este proceso ajusta los pesos internos del modelo para convertirlo en un experto en un nicho específico, enseñándole nuevas habilidades, estilos o formatos.[48]

- **Cuándo Utilizarlo**: El ajuste fino es ideal para enseñar al modelo una nueva *habilidad*, *estilo* o *formato*. Por ejemplo:
    - Enseñarle a comprender la jerga y los acrónimos únicos de su industria.[48]
    - Entrenarlo para generar texto en un formato específico, como una cláusula legal o un resumen médico.
    - Adaptarlo para que comprenda el diseño de un tipo de documento específico, como una factura de un proveedor particular.[27]
- **Métodos**:
    - **Ajuste Fino Completo**: Reentrenar todos los parámetros del modelo. Este método es computacionalmente muy costoso y, en general, no es factible para la mayoría de las organizaciones con hardware de consumo.[1]
    - **Ajuste Fino Eficiente en Parámetros (PEFT)**: Técnicas como **LoRA (Low-Rank Adaptation)** son mucho más eficientes. Congelan los pesos del modelo original y entrenan solo un pequeño número de nuevas capas "adaptadoras". Esto reduce drásticamente los requisitos computacionales y de memoria, haciendo que el ajuste fino sea accesible en GPU de consumo.[1]

5.2. Estrategia 2: Generación Aumentada por Recuperación (RAG) para un Contexto en Tiempo Real

RAG es un marco que conecta un SLM a una base de conocimientos externa (p. ej., documentos internos de una empresa, bases de datos o API). Cuando se recibe una consulta,

el sistema primero *recupera* información relevante de la base de conocimientos y luego proporciona esta información al SLM como contexto para *generar* una respuesta.[45]

- **Cuándo Utilizarlo**: RAG es la solución ideal para tareas que requieren acceso a información factual, actualizada o propietaria. Se utiliza para enseñar al modelo nuevos *hechos*, no nuevas habilidades.
    - **Chatbots de Soporte al Cliente**: Responden preguntas basadas en la documentación más reciente del producto.[45]
    - **Preguntas y Respuestas sobre la Base de Conocimientos Interna**: Permiten a los empleados hacer preguntas sobre políticas de la empresa, historiales de proyectos o documentación técnica.[47]
    - **Análisis Financiero**: Generan informes basados en datos de mercado en tiempo real obtenidos de una API.
- **Componentes Centrales de un Pipeline RAG**:
    1. **Base de Conocimientos**: Su colección de documentos (PDF, documentos de Word, etc.) o datos.
    2. **Fragmentación e Incrustación (Chunking & Embedding)**: Los documentos se dividen en fragmentos más pequeños, y un modelo de incrustación convierte cada fragmento en un vector numérico.
    3. **Almacén de Vectores (Vector Store)**: Una base de datos especializada que almacena estos vectores para una búsqueda eficiente.
    4. **Recuperador (Retriever)**: Cuando un usuario hace una pregunta, esta también se convierte en un vector, y el recuperador encuentra los fragmentos de documentos más similares (es decir, más relevantes) del almacén de vectores.
    5. **Generador (SLM)**: La pregunta original y los fragmentos recuperados se entregan al SLM, que genera una respuesta final con conciencia contextual.[45]

5.3. Enfoques Híbridos: RAG vs. Ajuste Fino

Los dos métodos no son mutuamente excluyentes y pueden combinarse para crear soluciones aún más potentes. Un marco de decisión claro puede ayudar a determinar el mejor enfoque.

- **Use RAG cuando**: La tarea es intensiva en conocimiento, requiere información actualizada y necesita trazabilidad de la fuente. Generalmente es más barato, más rápido de implementar y más fácil de mantener que el ajuste fino.[47]
- **Use el Ajuste Fino cuando**: Necesita cambiar el comportamiento fundamental, el estilo o el vocabulario especializado del modelo.
- **Ejemplo Híbrido**: Para un bot de análisis financiero, se podría *ajustar finamente* un SLM en informes financieros para enseñarle el lenguaje y la estructura de las finanzas (una nueva habilidad), y luego usar un sistema *RAG* para alimentarlo con precios de acciones y noticias en tiempo real para generar su análisis (nuevos hechos).

El marco RAG aborda las debilidades centrales de los modelos de lenguaje: la alucinación y el conocimiento desactualizado.[45] Para una empresa, donde la precisión y la responsabilidad

son primordiales, esto no es solo una característica, es un requisito. La implicación es que la ventaja competitiva a largo plazo en la IA empresarial no provendrá de tener el mejor modelo base, sino de tener los mejores datos propietarios y la arquitectura RAG más efectiva para aprovecharlos. RAG transforma el SLM de un generador de texto de "caja negra" a un motor de razonamiento transparente y controlable que opera sobre los datos de confianza de una empresa.

Además, la implementación de un sistema RAG [45] obliga a una organización a confrontar la calidad y gobernanza de sus datos. Un pipeline RAG es tan bueno como la base de conocimientos de la que se nutre. Si la documentación interna está desactualizada, es contradictoria o está mal organizada, el sistema RAG producirá malos resultados. Por lo tanto, la decisión de adoptar RAG tiene un poderoso efecto de segundo orden: crea un fuerte incentivo empresarial para invertir en la curación de datos, la gestión del conocimiento y la gobernanza de datos. El acto de construir un sistema RAG a menudo expone y ayuda a solucionar problemas de gestión de la información de larga data dentro de una organización, proporcionando un valor que va mucho más allá de la propia aplicación de IA.

Sección 6: Recomendaciones Estratégicas y Perspectivas Futuras

Esta sección final sintetiza los hallazgos del informe en un marco estratégico de alto nivel para el lector y proporciona una perspectiva a futuro sobre la evolución del panorama de los SLM.

6.1. Un Marco de Decisión para la Adopción de SLM

Las organizaciones que consideren la adopción de SLM locales deben evaluar su estrategia a través de una serie de preguntas clave:

1. **Ajuste Problema-Solución**: ¿Nuestro caso de uso requiere latencia en tiempo real, privacidad de datos y capacidad sin conexión? Si la respuesta es afirmativa, se debe favorecer un SLM local.
2. **Complejidad de la Tarea**: ¿La tarea es específica de un dominio (adecuada para un SLM) o requiere un conocimiento general del mundo y un razonamiento complejo de varios pasos? Si es lo segundo, un LLM más potente podría seguir siendo necesario.
3. **Verificación de la Realidad del Hardware**: ¿Cuál es nuestro hardware disponible (especialmente VRAM)? Esto dictará el tamaño y el nivel de cuantización del modelo que podemos ejecutar, utilizando la Tabla 1 como guía.
4. **Estrategia de Personalización**: ¿Nuestro problema requiere nuevos *hechos* (comenzar con RAG) o nuevas *habilidades* (considerar el ajuste fino)?
5. **Construir vs. Comprar**: ¿Deberíamos construir una solución personalizada a partir de un modelo de código abierto, o existen SLM comerciales especializados y pre-ajustados que se adapten a nuestras necesidades?

6.2. Tendencias Emergentes: El Futuro de la IA en el Dispositivo y la IA Agéntica

El panorama de los SLM está evolucionando rápidamente, impulsado por varias tendencias clave que darán forma al futuro de la IA empresarial.

- **El Auge de los Sistemas Agénticos**: El futuro de la automatización empresarial reside en la creación de sistemas colaborativos de múltiples agentes SLM especializados. Esta arquitectura es más resiliente, escalable y potente que los modelos monolíticos, permitiendo la automatización de flujos de trabajo complejos que requieren diversas habilidades.[12]
- **IA Hiper-Eficiente en el Dispositivo**: La tendencia hacia modelos más pequeños y potentes continuará, impulsando más capacidades de IA directamente a dispositivos móviles y de borde. Esto desbloqueará nuevas aplicaciones en el IoT, los wearables y la computación personal que requieren latencia cero y privacidad total.[8]
- **La Comoditización del Razonamiento**: A medida que los potentes SLM de código abierto se vuelven omnipresentes y fáciles de ejecutar, la capacidad central de realizar razonamiento basado en el lenguaje se convertirá en una mercancía. El diferenciador clave para las empresas serán sus datos propietarios y las formas únicas en que integren estos "motores de razonamiento" SLM en sus productos y flujos de trabajo a través de marcos RAG y agénticos.

6.3. Recomendaciones Finales

Para las organizaciones que se embarcan en su viaje con los SLM, las siguientes recomendaciones pueden servir como una hoja de ruta pragmática:

- **Comenzar Pequeño, Comenzar Ahora**: Se alienta a las organizaciones a comenzar a experimentar de inmediato utilizando herramientas como Ollama o LM Studio en el hardware existente. La barrera de entrada nunca ha sido tan baja, y la experimentación práctica es la forma más rápida de comprender el potencial y las limitaciones de estos modelos.
- **Priorizar RAG**: Para la mayoría de los casos de uso empresarial que involucran datos propietarios, construir un pipeline RAG robusto es el primer paso más pragmático y de mayor valor. Asegura la precisión, la fiabilidad y proporciona una base sólida sobre la cual construir aplicaciones de IA más complejas.
- **Desarrollar una Mentalidad de Portafolio**: Se desaconseja la búsqueda de un único "mejor" SLM. En su lugar, se debe fomentar la construcción de una infraestructura flexible que pueda acomodar un portafolio de diferentes modelos, cada uno elegido por sus fortalezas específicas en relación con una tarea empresarial determinada. Este enfoque permite a la organización aprovechar el mejor modelo para cada trabajo, optimizando el rendimiento y la eficiencia en toda la empresa.

Obras citadas

1. Small Language Models (SLM): A Comprehensive Overview - Hugging Face, fecha de acceso: septiembre 21, 2025, https://huggingface.co/blog/jjokah/small-language-model
2. Small Language Models (SLMs) Can Still Pack a Punch: A survey - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/html/2501.05465v1
3. What are Small Language Models (SLM)? - IBM, fecha de acceso: septiembre 21, 2025, https://www.ibm.com/think/topics/small-language-models
4. LLMs vs. SLMs: The Differences in Large & Small Language Models | Splunk, fecha de acceso: septiembre 21, 2025, https://www.splunk.com/en_us/blog/learn/language-models-slm-vs-llm.html
5. [2505.19529] Small Language Models: Architectures, Techniques, Evaluation, Problems and Future Adaptation - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/abs/2505.19529
6. [2410.20011] A Survey of Small Language Models - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/abs/2410.20011
7. SLMs vs LLMs: What are small language models? - Red Hat, fecha de acceso: septiembre 21, 2025, https://www.redhat.com/en/topics/ai/llm-vs-slm
8. [2409.15790] Small Language Models: Survey, Measurements, and Insights - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/abs/2409.15790
9. Small Language Models vs. LLMs: Finding the Right Fit for Your Needs - Iris.ai, fecha de acceso: septiembre 21, 2025, https://iris.ai/blog/small-language-models-vs-ll-ms-finding-the-right-fit-for-your-needs
10. Small Language Models (SLMs) vs LLMs: Benefits & Use Cases - Talent500, fecha de acceso: septiembre 21, 2025, https://talent500.com/blog/small-language-models-slms-vs-llms-benefits-use-cases/
11. SLM vs LLM: Choosing the Right AI Model for Your Business - Openxcell, fecha de acceso: septiembre 21, 2025, https://www.openxcell.com/blog/slm-vs-llm/
12. Advantages of SLM vs LLM - ModelOp, fecha de acceso: septiembre 21, 2025, https://www.modelop.com/ai-governance/slm-vs-llm
13. Phi-3 Technical Report | Continuum Labs, fecha de acceso: septiembre 21, 2025, https://training.continuumlabs.ai/models/foundation-models/phi-3-technical-report
14. Running the larger Google Gemma 7B 35GB LLM on a single GPU for over 7x Inference Performance gains - Michael O'Brien, fecha de acceso: septiembre 21, 2025, https://obrienlabs.medium.com/running-the-larger-google-gemma-7b-35gb-llm-for-7x-inference-performance-gain-8b63019523bb
15. google/gemma-7b · [AUTOMATED] Model Memory Requirements - Hugging Face, fecha de acceso: septiembre 21, 2025, https://huggingface.co/google/gemma-7b/discussions/67

16. GPU System Requirements Guide for Gemma 3 Multimodal - ApX Machine Learning, fecha de acceso: septiembre 21, 2025, https://apxml.com/posts/gemma-3-gpu-requirements
17. Gemma 3 QAT Models: Bringing state-of-the-Art AI to consumer GPUs, fecha de acceso: septiembre 21, 2025, https://developers.googleblog.com/en/gemma-3-quantized-aware-trained-state-of-the-art-ai-to-consumer-gpus/
18. Phi-3 is a family of lightweight 3B (Mini) and 14B (Medium) state-of-the-art open models by Microsoft. - Ollama, fecha de acceso: septiembre 21, 2025, https://ollama.com/library/phi3
19. Phi-3-mini: Specifications and GPU VRAM Requirements - ApX Machine Learning, fecha de acceso: septiembre 21, 2025, https://apxml.com/models/phi-3-mini
20. ollama/ollama: Get up and running with OpenAI gpt-oss, DeepSeek-R1, Gemma 3 and other models. - GitHub, fecha de acceso: septiembre 21, 2025, https://github.com/ollama/ollama
21. Could you provide information on the minimum GPU requirements and the supported virtual machines for the phi-3-medium-128k-instruct model in Azure? - Microsoft Learn, fecha de acceso: septiembre 21, 2025, https://learn.microsoft.com/en-us/answers/questions/2225708/could-you-provide-information-on-the-minimum-gpu-r
22. Small Language Models (SLMs) inference with llama.cpp on Graviton4, fecha de acceso: septiembre 21, 2025, https://builder.aws.com/content/2pViH36qEfqDYwv0kEysJ0kqDp4/small-language-models-slms-inference-with-llamacpp-on-graviton4
23. Run SLMs locally: Llama.cpp vs. MLX with 10B and 32B Arcee models - YouTube, fecha de acceso: septiembre 21, 2025, https://www.youtube.com/watch?v=Vk1W6evtsjE
24. Challenging GPU Dominance: When CPUs Outperform for On-Device LLM Inference - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/html/2505.06461v1
25. Achieving Maximum CPU Performance in Local SLMs | by Robert Hallock (Intel) | Medium, fecha de acceso: septiembre 21, 2025, https://medium.com/@intel.robert/achieving-maximum-cpu-performance-in-local-slms-55c8571aadb0
26. Phi-3 Tutorial: Hands-On With Microsoft's Smallest AI Model - DataCamp, fecha de acceso: septiembre 21, 2025, https://www.datacamp.com/tutorial/phi-3-tutorial
27. Fine-tuning Qwen2.5-VL for Document Information Extraction - Ubiai, fecha de acceso: septiembre 21, 2025, https://ubiai.tools/how-to-fine-tune-qwen2-5-vl-for-document-information-extraction/
28. How to Use Phi3 with Ollama: Step-by-Step Guide - BytePlus, fecha de acceso: septiembre 21, 2025, https://www.byteplus.com/en/topic/553346
29. Get started with LM Studio | LM Studio Docs, fecha de acceso: septiembre 21, 2025, https://lmstudio.ai/docs/app/basics

30. LMStudio LLM - AnythingLLM Docs, fecha de acceso: septiembre 21, 2025, https://docs.useanything.com/setup/llm-configuration/local/lmstudio
31. LM Studio - LiteLLM, fecha de acceso: septiembre 21, 2025, https://docs.litellm.ai/docs/providers/lm_studio
32. microsoft/Phi-3-mini-128k-instruct - Hugging Face, fecha de acceso: septiembre 21, 2025, https://huggingface.co/microsoft/Phi-3-mini-128k-instruct
33. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/pdf/2404.14219
34. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone, fecha de acceso: septiembre 21, 2025, https://www.microsoft.com/en-us/research/publication/phi-3-technical-report-a-highly-capable-language-model-locally-on-your-phone/
35. [2503.19786] Gemma 3 Technical Report - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/abs/2503.19786
36. Gemma - Google DeepMind, fecha de acceso: septiembre 21, 2025, https://deepmind.google/models/gemma/
37. google/gemma-3-1b-it - Hugging Face, fecha de acceso: septiembre 21, 2025, https://huggingface.co/google/gemma-3-1b-it
38. Introducing Gemma 3: The most capable model you can run on a single GPU or TPU, fecha de acceso: septiembre 21, 2025, https://blog.google/technology/developers/gemma-3/
39. Top 10 Small Language Models [SLMs] in 2025 - Intuz, fecha de acceso: septiembre 21, 2025, https://www.intuz.com/blog/best-small-language-models
40. Recommendations for Summarization Models (Small Context)? : r/LocalLLaMA - Reddit, fecha de acceso: septiembre 21, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1dk78ge/recommendations_for_summarization_models_small/
41. phi3:instruct - Ollama, fecha de acceso: septiembre 21, 2025, https://ollama.com/library/phi3:instruct
42. Best route for text extraction from Invoice documents - Beginners - Hugging Face Forums, fecha de acceso: septiembre 21, 2025, https://discuss.huggingface.co/t/best-route-for-text-extraction-from-invoice-documents/134854
43. NuExtract 1.5: The Best Open-Source SLM for Text Extraction - YouTube, fecha de acceso: septiembre 21, 2025, https://www.youtube.com/watch?v=WtbApiyLaPQ
44. Compare Gemma 2 vs. Llama 3 vs. Phi-3 in 2025 - Slashdot, fecha de acceso: septiembre 21, 2025, https://slashdot.org/software/comparison/Gemma-2-vs-Llama-3-vs-Phi-3/
45. What Is RAG (Retrieval-Augmented Generation)? A Full Guide - Snowflake, fecha de acceso: septiembre 21, 2025, https://www.snowflake.com/en/fundamentals/rag/
46. RAG in Customer Support: Enhancing Chatbots and Virtual Assistants - Signity Solutions, fecha de acceso: septiembre 21, 2025, https://www.signitysolutions.com/blog/rag-in-customer-support
47. Evaluating Your RAG Solution - Towards Data Science, fecha de acceso:

septiembre 21, 2025, https://towardsdatascience.com/evaluating-your-rag-solution/
48. How to Fine Tune LLMs for Your Documents and Data, fecha de acceso: septiembre 21, 2025, https://blog.bisok.com/general-technology/how-to-fine-tune-llm
49. Why do most RAG Applications utilise LLMS rather than Small Language Models - Reddit, fecha de acceso: septiembre 21, 2025, https://www.reddit.com/r/LocalLLM/comments/1h6iifd/why_do_most_rag_applications_utilise_llms_rather/
50. SlimLM: An Efficient Small Language Model for On-Device Document Assistance - arXiv, fecha de acceso: septiembre 21, 2025, https://arxiv.org/html/2411.09944v1