

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Ingeniería de Computadores

Hau izenburua da/Título del trabajo

Egilearen izen-abizenak / Nombre y apellidos de la o el autor

Dirección

Zuzendaria 1 / Director(a) 1

Zuzendaria 2 / Director(a) 2

7 de abril de 2025

Esker onak / Agradecimientos

Eskerrak eman nahi izanez gero, hemen idatzi testua. En caso de querer añadir agradecimientos, escribir aquí el texto.

Atal hau nahi ez baduzu, *main.tex* fitxategian komentatu lerro hori. En caso de no querer este apartado, comentalo en el fichero *main.tex*.

Laburpena / Resumen

Idatzi hemen laburpena. Escribe aquí el resumen.

Índice de contenidos

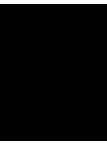
Índice de contenidos	v
Índice de figuras	vi
Índice de tablas	vii
Índice de algoritmos	ix
1 Introducción	1
2 Antecedentes	3
2.1. Agentes LLM	3
2.1.1. Modelos LLM	3
2.1.2. Interacción con herramientas externas	4
2.1.3. Abstracciones en frameworks	5
2.2. Model Context Protocol	5
2.3. Estado del arte en arquitecturas de agentes LLM	6
2.3.1. Arquitectura RAG	7
2.3.2. Arquitecturas de interacción entre agentes	7
Eranskina / apéndice	9
Bibliografía	11

Índice de figuras

2.1.	Ejemplo de interacción de un modelo LLM con una herramienta externa.	4
2.2.	Esquema de funcionamiento del Model Context Protocol.	6

Índice de tablas

Lista de Algoritmos



Introducción

LLMs han avanzado mucho, las apps que usan gen IA están surgiendo blah blah blah.

Los agentes ya hace mucho, pero ahora se están popularizando agentes LLM. Para esto están surgiendo frameworks blah blah blah.

Además se suele trabajar con ajustes de modelos, sistemas RAG para sacarle más provecho a las aplicaciones.

En este contexto LKS Next quiere investigar sobre las diferentes arquitecturas de agentes y sus aplicaciones. Para ello se dispone del escenario de onboarding.

a

Antecedentes

Resumen de diferentes partes:

- Ajuste de agentes ->LoRa? ->comentar lo de los agentes instruct. - El protocolo MCP - El estado del arte en arquitecturas de agentes LLM y sistemas RAG - - El estado del arte en agentes integrados a proyectos software + contexto de onboarding quizás en la introducción.
- Redes neuronales?

2. Agentes LLM (Fundamentos y funcionamiento básico)

¿Qué son los agentes LLM? 2.1. Modelos LLM 2.2. Interacción con herramientas 2.3. Abstracciones en frameworks

2.1. Agentes LLM

Los agentes de Inteligencia Artificial son programas informáticos que implementan modelos computacionales para ejecutar diversas funciones específicas del contexto en el que se aplican. Tras siete décadas y media de investigación, los esfuerzos en el campo se han focalizado en agentes basados en Grandes Modelos de Lenguaje (LLM).

2.1.1. Modelos LLM

Los LLM son redes neuronales especializadas en el procesamiento del lenguaje natural, basados en la arquitectura Transformer. Esta arquitectura se fundamenta en el mecanismo de atención, el cual transforma la representación del texto para incorporar información contextual de manera escalable al hardware.

Para comprender el funcionamiento de estos agentes, resulta imprescindible asimilar previamente conceptos como la tokenización y las representaciones vectoriales del lenguaje.

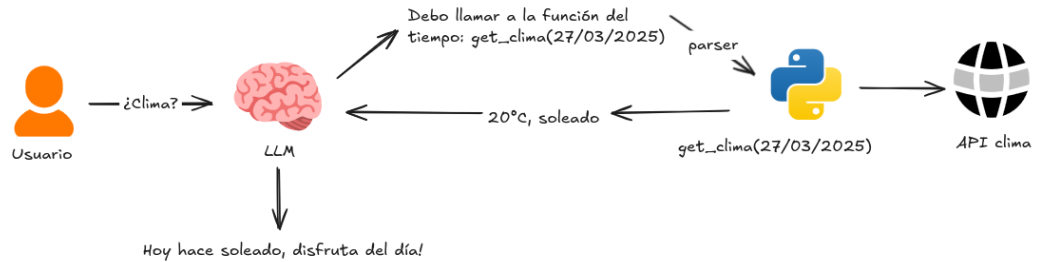


Figura 2.1: Ejemplo de interacción de un modelo LLM con una herramienta externa.

Tokens Los tokens constituyen la unidad mínima de texto que el modelo puede procesar. Dado que dichos modelos operan sobre estructuras matemáticas, requieren transformar el lenguaje natural en representaciones matriciales. Para lograr esta conversión, el texto se segmenta en dichas unidades mínimas, que pueden corresponder a caracteres individuales, fragmentos de texto o palabras completas. El conjunto íntegro de estas unidades reconocibles por el modelo configura su vocabulario.

Representaciones vectoriales Constituyen vectores numéricos de dimensionalidad fija que codifican la semántica inherente a cada token. Estos vectores pueden comprender desde 768 dimensiones en arquitecturas como BERT-base hasta superar las 16.000 dimensiones en los modelos más avanzados del estado del arte. Por ejemplo, una dimensión específica podría especializarse en representar conceptos abstractos. En este contexto, la representación vectorial del token “animal” contendría un valor más elevado en dicha dimensión que la correspondiente al término “gato”, reflejando su mayor grado de abstracción conceptual.

2.1.2. Interacción con herramientas externas

Los agentes LLM poseen la capacidad de interactuar con diversas herramientas como búsquedas web, bases de datos o interfaces de usuario. Fundamentalmente, un LLM solo genera tokens de texto, por lo que la integración de herramientas se implementa mediante palabras clave que el modelo puede incluir en su salida. Para ello, en el texto de entrada se especifica el esquema de la función a utilizar y, si decide emplearla, el modelo generará el texto correspondiente. Posteriormente, se procesa la respuesta para extraer llamadas a funciones si las hubiese.

La interacción con herramientas es típicamente alternante. Tras realizar la llamada a la herramienta, la salida de esta se utilizará como entrada para el siguiente mensaje del modelo. La figura 2.1 ilustra el esquema de un agente con acceso a una API del clima. Como el modelo carece de información climática en tiempo real, se le indica en el prompt la posibilidad de invocar esta función. Al incluir la llamada en su texto de salida, se ejecuta la función y su respuesta se transmite al modelo para generar el resultado final.

2.1.3. Abstracciones en frameworks

Aunque los agentes LLM son una tecnología de reciente surgimiento, ya se han desarrollado frameworks que estandarizan su implementación. Estas estructuras de trabajo, como LangChain o LangGraph, ofrecen abstracciones de alto nivel para reutilizar funcionalidades comunes presentes en la mayoría de sistemas de agentes. Las funcionalidades principales que estos frameworks proporcionan son:

- **Gestión de modelos:** La ejecución de LLM requiere dominio del modelo empleado, ya que cada uno posee tokenizadores específicos y esquemas propios de entrada/salida. Los frameworks ofrecen interfaces unificadas, facilitando el uso de diversos modelos sin conocimientos técnicos precisos.
- **Interacción conversacional:** La comunicación con los modelos se efectúa mediante un esquema conversacional, donde el modelo recibe un texto de entrada y genera una respuesta correspondiente. Las respuestas y entradas se concatenan secuencialmente para preservar el contexto de la conversación, cada consulta subsiguiente incorpora todos los intercambios precedentes.
- **Uso de herramientas externas:** El desarrollador únicamente debe especificar la función que desea incorporar, toda la complejidad de la interacción se abstrae en el framework.
- **Interacción entre agentes:** Los agentes pueden establecer comunicación entre sí, permitiendo la construcción de sistemas con mayor complejidad. Algunos frameworks establecen protocolos que definen las modalidades de comunicación entre los distintos agentes.

2.2. Model Context Protocol

todo: referencias a las docs.

El Model Context Protocol (MCP), desarrollado por Anthropic, estandariza la comunicación entre agentes LLM y herramientas. Permite que aplicaciones diversas ofrezcan herramientas a agentes externos sin exponer detalles de implementación. Comparable al modelo OSI, el MCP opera en un nivel de abstracción inferior a los frameworks, proporcionando una capa de interoperabilidad.

La figura 2.2 ilustra el esquema operativo del protocolo. En este contexto, los desarrolladores de Jira y GitHub han implementado un servidor MCP que proporciona las herramientas disponibles al cliente MCP. Este servidor realiza la traducción de las interacciones necesarias con las API de Jira y GitHub, permitiendo que el agente LLM únicamente requiera conocer el esquema funcional que debe utilizar. Por su parte, el cliente MCP se encarga de administrar la comunicación con los diferentes servidores, facilitando que el agente acceda directamente a las herramientas disponibles.

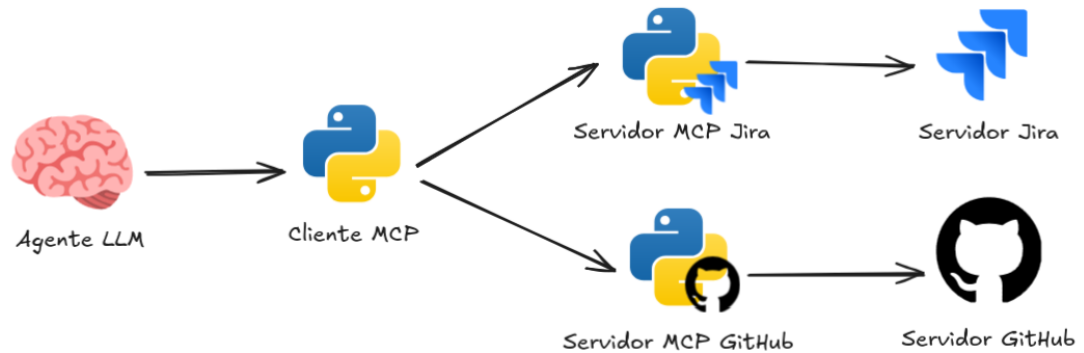


Figura 2.2: Esquema de funcionamiento del Model Context Protocol.

El protocolo ofrece dos modos de operación para establecer la comunicación entre cliente y servidor:

- **Comunicación SSE:** El protocolo Server-Sent Events (SSE) establece un canal de comunicación unidireccional sobre HTTP desde el servidor hacia el cliente. Proporciona actualizaciones en tiempo real con capacidad de streaming. En el protocolo MCP, el cliente efectúa solicitudes para la ejecución de herramientas en el servidor mediante HTTP, a lo que el servidor puede responder mediante eventos SSE.
- **Comunicación STDIO:** El protocolo de entrada y salida estándar (STDIO) facilita la comunicación bidireccional entre cliente y servidor a nivel de proceso en el sistema operativo. Este mecanismo permite el intercambio de información en formato JSON a través de los canales estándar del sistema. Su diseño, orientado principalmente a entornos locales, restringe la conexión a un único cliente por servidor al limitarse a la comunicación entre dos procesos.

La aplicación de escritorio claude-desktop de Anthropic constituye un reflejo del potencial del protocolo. Esta plataforma ofrece la posibilidad de interactuar con servidores preconfigurados mediante una configuración mínima. Implementando el protocolo STDIO, la aplicación ejecuta los servidores distribuidos por terceros a través del gestor de paquetes UV o Docker. Al incorporar un cliente MCP en la aplicación, consigue integrar las herramientas disponibles en la interfaz de chat con los modelos de Anthropic.

2.3. Estado del arte en arquitecturas de agentes LLM

La comunidad científica ha desarrollado diferentes arquitecturas de agentes para sacar el máximo provecho al rendimiento de los modelos disponibles. Es de destacar la arquitectura RAG, la cual complementa la entrada del modelo con la recuperación de documentos relevantes. Por otro lado, se han realizado diversos esfuerzos en investigar diferentes arquitecturas de comunicación, estrategias de planning o sistemas de memoria, entre otros.

2.3.1. Arquitectura RAG

Los modelos LLM poseen un conocimiento restringido a los datos con los que fueron entrenados. Para superar esta limitación, la arquitectura RAG (Retrieval-Augmented Generation) complementa la generación del LLM mediante la recuperación de información relevante desde repositorios de conocimiento externos. La figura x ilustra un ejemplo de su funcionamiento.

La recuperación de documentos relevantes se implementa mediante diversas estrategias: expresiones regulares, búsqueda de trigramas [ref], palabras clave, entre otras. No obstante, el enfoque predominante consiste en la indexación vectorial. En este método, los documentos se transforman en vectores, generalmente mediante LLMs especializados en codificación, denominados Embedders. Al representar los documentos en un espacio vectorial, es posible recuperar aquellos semánticamente más pertinentes mediante la comparación del vector de consulta con los vectores de los documentos indexados, utilizando métricas como la distancia coseno.

2.3.1.1. Estrategias RAG avanzadas

La mejora del rendimiento de la arquitectura RAG ha sido objeto de extensa investigación [1][2]. La optimización de esta estrategia se ha centrado en mejorar el procesamiento de documentos, sistemas de extracción, sistemas de recuperación y optimización del pipeline general:

- **procesado de documentos:** El procesamiento previo a la indexación de documentos determina la calidad de la información. Entre las estrategias más destacadas se encuentran optimizaciones como limpiar el ruido de los documentos, la optimización del tamaño de chunks para el embedder utilizado, o el sistema overlapping window, el cual consiste en superponer los chunks para evitar la pérdida de información en los límites de los mismos.

Adicionalmente, se han desarrollado técnicas de generación para mejorar la calidad de los documentos. Estas técnicas varían desde recuperar documentos directamente generados[3], recitar pasajes de evidencia[4], o utilizar un documento generado a partir de la query para buscar documentos relevantes[5]. Estas técnicas se pueden combinar con el algoritmo beam search para más efectividad[6].

- **sistemas de extracción:**
- **sistemas de recuperación:**
- **optimización del pipeline:**

2.3.2. Arquitecturas de interacción entre agentes

La interacción entre agentes LLM es también un campo de investigación activo.

Eranskina / apéndice

Eranskinak

Bibliografía

- [1] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. Ver página 7.
- [2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. Ver página 7.
- [3] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. Ver página 7.
- [4] Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. Recitation-augmented language models. Ver página 7.
- [5] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777. Association for Computational Linguistics. Ver página 7.
- [6] Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. Lift yourself up: Retrieval-augmented text generation with self-memory. Ver página 7.