Universidad Nacional de Ingeniería (UNI)
Escuela Profesional de
Ciencia de la Computación
Sílabo 2027-I

1. CURSO

CS3E2. Ingenieria Aplicada de LLMs (Electivo)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS3E2. Ingenieria Aplicada de LLMs
2.2 Ciclo	:	CS300 Ciclo.
2.3 Créditos	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Electivo
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	▪ AI264. Aprendizaje Profundo. (7^{mo} Sem) ▪ AI262. Procesamiento de Lenguaje Natural.

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Los modelos de lenguaje grandes (LLM) se han convertido en una tecnología fundamental para construir aplicaciones de software modernas; sin embargo, diseñar sistemas de manera efectiva alrededor de ellos requiere habilidades que van mucho más allá del aprendizaje automático o la ingeniería de software clásicos: prompting, generación aumentada por recuperación, ajuste fino eficiente en parámetros, flujos de trabajo agénticos con uso de herramientas, y las prácticas de evaluación, servicio y gobernanza que hacen que una aplicación basada en LLM sea confiable y esté lista para producción.

Este curso brinda a los estudiantes experiencia práctica a lo largo de toda la pila de ingeniería de aplicaciones LLM. Los estudiantes comienzan repasando la arquitectura transformer, la tokenización y el pipeline de preentrenamiento/inferencia que sustentan a los LLM modernos, y luego avanzan a través de la ingeniería de prompts, la generación aumentada por recuperación y el ajuste fino eficiente en parámetros (LoRA/PEFT) como las tres técnicas principales para adaptar el comportamiento de un LLM a una tarea específica sin reentrenarlo desde cero.

El curso aborda luego los flujos de trabajo agénticos y las arquitecturas de uso de herramientas, en los que un LLM planifica y ejecuta tareas de múltiples pasos usando herramientas externas, así como los métodos de evaluación (benchmarks, detección de alucinaciones, pruebas de seguridad/red-teaming) necesarios para determinar si un sistema basado en LLM está lo suficientemente maduro para desplegarse. El curso cierra con LLM Ops (servicio, optimización de latencia/costo, caché, cuantización), prácticas de IA responsable (sesgo, privacidad, seguridad ante inyección de prompts, gobernanza), y un proyecto capstone en el que los estudiantes integran estas técnicas en una sola aplicación de extremo a extremo basada en LLM.

5. OBJETIVOS

- Explicar la arquitectura, el preentrenamiento y el pipeline de inferencia de los modelos de lenguaje grandes (LLM) modernos.
- Diseñar prompts efectivos y pipelines de generación aumentada por recuperación (RAG) para fundamentar la salida de un LLM en conocimiento externo.
- Aplicar técnicas de ajuste fino eficiente en parámetros (LoRA/PEFT) para adaptar un LLM preentrenado a una tarea o dominio específico.
- Diseñar flujos de trabajo agénticos basados en LLM que usan herramientas, y evaluar la calidad, seguridad y robustez de sistemas basados en LLM.

- Operar aplicaciones basadas en LLM de forma responsable en producción, atendiendo las compensaciones de servicio/costo, los riesgos de seguridad y los requisitos de gobernanza, e integrar estas técnicas en una aplicación de extremo a extremo desplegable.

6. RESULTADOS DEL ESTUDIANTE

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Usage)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Usage)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Assessment)

7. TEMAS

<table>
<tr><td colspan="2">Unidad 1: Fundamentos de los Modelos de Lenguaje Grandes (8 horas)</td></tr>
<tr><td colspan="2">Resultados esperados: AG-C07</td></tr>
<tr><td>Temas</td><td>Aprendizaje esperado (Learning Outcomes)</td></tr>
<tr><td>

Esquemas de tokenización en subpalabras (BPE, WordPiece, tokenización a nivel de byte) y su efecto en el tamaño del vocabulario, la longitud de secuencia y la cobertura multilingüe Procesamiento del Lenguaje Natural.
Arquitectura transformer de tipo decodificador para modelado de lenguaje autorregresivo: enmascaramiento de autoatención causal, codificaciones posicionales, y la pila de bloques transformer usada en los LLM modernos Arquitecturas Transformer y Modelos Generativos.
El objetivo de preentrenamiento (predicción del siguiente token) sobre grandes corpus de texto no etiquetado, y las leyes de escalamiento que relacionan el tamaño del modelo, el tamaño de los datos y el cómputo con la pérdida de preentrenamiento Entrenamiento y Optimización de Redes Neuronales.
Inferencia autorregresiva y estrategias de decodificación: decodificación voraz (greedy), búsqueda en haz (beam search), temperatura y muestreo top-k/top-p, y la caché KV que hace eficiente la decodificación incremental.

</td><td>

Explicar cómo un transformer de tipo decodificador procesa una secuencia de entrada tokenizada para producir la distribución de probabilidad del siguiente token [Explicar]
Comparar esquemas de tokenización en subpalabras en términos de tamaño de vocabulario, longitud de secuencia y manejo de palabras fuera de vocabulario [Comparar]
Aplicar diferentes estrategias de decodificación (voraz, búsqueda en haz, muestreo) a un LLM preentrenado y analizar el efecto sobre la calidad y diversidad de la salida [Aplicar]
Analizar cómo el tamaño del modelo, el tamaño de los datos de entrenamiento y el presupuesto de cómputo afectan la pérdida de preentrenamiento de un LLM, dado un conjunto de mediciones de leyes de escalamiento [Analizar]

</td></tr>
<tr><td colspan="2">Lecturas : [Vas+17], [Ras24]</td></tr>
</table>

Unidad 2: Técnicas de Ingeniería de Prompts (6 horas)	
Resultados esperados: AG-C09	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Prompting de cero ejemplos (*zero-shot*) y de pocos ejemplos (*few-shot*): formular una tarea solo con instrucciones frente a proporcionar demostraciones de entrada/salida en contexto. ▪ Prompts de cadena de pensamiento (*chain-of-thought*) y razonamiento paso a paso, y autoconsistencia entre múltiples rutas de razonamiento muestreadas. ▪ Prompts de sistema/rol y formateo de salida estructurada (por ejemplo, solicitar respuestas restringidas a JSON o XML). ▪ Sensibilidad y robustez del prompt: cómo pequeños cambios de redacción u orden afectan la salida del modelo, y el refinamiento iterativo de prompts Ingeniería de Bucles de Agentes (Loop Engineering).	▪ Diseñar un prompt de pocos ejemplos con demostraciones en contexto para una tarea dada de clasificación o extracción [Diseñar] ▪ Aplicar prompting de cadena de pensamiento para mejorar el desempeño de un modelo en una tarea de razonamiento de múltiples pasos [Aplicar] ▪ Evaluar la sensibilidad de un prompt a cambios de redacción y orden, y refinarlo para reducir la variabilidad de la salida [Evaluar]
Lecturas : [Whi+23], [Wei+22a]	

Unidad 3: Generación Aumentada por Recuperación (RAG) (8 horas)	
Resultados esperados: AG-C09	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Estrategias de fragmentación de documentos y embedding vectorial denso de fragmentos de texto para recuperación semántica. ▪ Búsqueda de similitud vectorial (similitud coseno/producto punto, índices de vecinos más cercanos aproximados) y selección de base de datos/índice vectorial Sistemas NoSQL, Bases de Datos Semiestructuradas y No Estructuradas. ▪ El pipeline de recuperar-y-generar: embedding de la consulta, recuperación de los top-k resultados, ensamblaje del contexto, y construcción del prompt para el modelo generador. ▪ Patrones avanzados de RAG: reordenamiento (*reranking*) de los fragmentos recuperados, recuperación híbrida (dispersa+densa), y recuperación de múltiples saltos/iterativa Procesamiento del Lenguaje Natural.	▪ Diseñar una estrategia de fragmentación y embedding para una colección de documentos dada, de modo que soporte recuperación semántica [Diseñar] ▪ Implementar un pipeline de recuperar-y-generar que fundamente la respuesta de un LLM en el contexto recuperado [Implementar] ▪ Comparar estrategias de recuperación densa, dispersa e híbrida en términos de exhaustividad y relevancia para un corpus dado [Comparar] ▪ Evaluar la fundamentación (*groundedness*) de las respuestas de un sistema RAG frente a los documentos fuente recuperados [Evaluar]
Lecturas : [Lew+20], [Gao+24]	

Unidad 4: Ajuste Fino y Métodos Eficientes en Parámetros (8 horas)	
Resultados esperados: AG-C07,AG-C09	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Ajuste fino completo de un LLM preentrenado frente al costo y los riesgos de sobreajuste de actualizar todos sus parámetros con un conjunto de datos pequeño de la tarea objetivo Técnicas de Regularización para Redes Profundas. ▪ Adaptación de bajo rango (LoRA) y métodos relacionados de ajuste fino eficiente en parámetros (PEFT) basados en adaptadores: actualizaciones de pesos de bajo rango, ubicación de adaptadores, y fusión de adaptadores en el modelo base. ▪ Ajuste fino cuantizado (por ejemplo, QLoRA) para adaptar modelos grandes con memoria de GPU limitada. ▪ Ajuste por instrucciones y ajuste fino basado en preferencias (por ejemplo, RLHF/DPO) para alinear el comportamiento de un modelo ajustado con las instrucciones y preferencias deseadas.	▪ Comparar el ajuste fino completo y el ajuste fino eficiente en parámetros basado en LoRA en términos de costo computacional, huella de memoria y desempeño en la tarea [Comparar] ▪ Implementar una corrida de ajuste fino con LoRA sobre un LLM preentrenado para una tarea objetivo [Implementar] ▪ Determinar un rango y ubicación de adaptador apropiados para un presupuesto y tarea de ajuste fino dados [Determinar] ▪ Explicar cómo el ajuste por instrucciones y el ajuste fino basado en preferencias cambian el comportamiento de un modelo base respecto a su estado preentrenado [Explicar]
Lecturas : [Hu+22], [Det+23]	

Unidad 5: Ingeniería de Bucles de Agentes (Loop Engineering) (8 horas)	
Resultados esperados: AG-C09	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ La ingeniería de bucles como disciplina: diseñar el ciclo iterativo observar-actuar-verificar que ejecuta un agente, en contraste con escribir un único prompt ▪ El patrón Gauntlet Loop (Matt Shumer, 2026): un agente líder descompone un objetivo y un estándar de calidad en las partes más pequeñas y evaluables posibles, y despliega agentes constructores y agentes críticos con contexto limpio ▪ Separación constructor/crítico: un agente que crea un resultado suele ser peor juzgando objetivamente su propia calidad que explicándola, de ahí la necesidad de un crítico independiente ▪ Uso de herramientas por parte del agente (lectura y escritura de archivos, terminal, búsqueda, ejecución de pruebas) como forma de actuar sobre el repositorio en lugar de limitarse a generar texto ▪ Protocolo de Contexto de Modelo (MCP) como interfaz estandarizada entre agentes y fuentes de datos o herramientas: servidores, recursos, prompts y herramientas expuestas ▪ Especificación ejecutable: artefactos versionados en el repositorio (especificaciones, planes y listas de tareas) que el agente consume y actualiza	▪ Explicar en qué se diferencia la ingeniería de bucles de la ingeniería de un único prompt [Explicar] ▪ Diseñar un bucle de tipo Gauntlet Loop (agente constructor + agente crítico independiente + comparación ciega contra un estándar de calidad) para refinar iterativamente un resultado generado por IA [Diseñar] ▪ Comparar niveles de autonomía de un agente (asistencia en línea, edición multi-archivo, ejecución no supervisada) en términos de riesgo y beneficio [Comparar] ▪ Diseñar y configurar un servidor MCP que exponga a un agente los recursos y acciones necesarios para una tarea de desarrollo [Diseñar]
Lecturas : [Yao+23]	

Unidad 6: Evaluación de Sistemas LLM: Benchmarks, Alucinación y Seguridad (6 horas)	
Resultados esperados: AG-C02,AG-C07	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Benchmarks estandarizados para LLM (por ejemplo, benchmarks de conocimiento y razonamiento multitarea) y métricas de evaluación automática para generación abierta (por ejemplo, LLM-como-juez) Aprendizaje Automático: Datos, Representaciones y Evaluación. ▪ Protocolos de evaluación humana: juicios de preferencia por pares, calificación basada en rúbricas, y acuerdo entre evaluadores. ▪ Alucinación: causas, técnicas de detección (por ejemplo, verificación de hechos contra una fuente recuperada, verificaciones de autoconsistencia), y estrategias de mitigación Agentes de Codificación basados en LLM. ▪ Pruebas adversarias y de seguridad (*red teaming*): intentos de *jailbreak*, pruebas de toxicidad/sesgo, y conjuntos de benchmarks de seguridad Redes Neuronales y Ética.	▪ Diseñar un protocolo de evaluación (selección de benchmarks más una rúbrica de evaluación humana) para una aplicación basada en LLM dada [Diseñar] ▪ Detectar contenido alucinado en la salida de un LLM verificándolo contra documentos fuente recuperados [Detectar] ▪ Evaluar la robustez de una aplicación basada en LLM frente a un intento de *red teaming*/*jailbreak* y proponer mitigaciones [Evaluar (valorar)]
Lecturas : [LBL+23]	

Unidad 7: Servicio de LLM, Optimización de Latencia y Costo (8 horas)	
Resultados esperados: AG-C09	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Arquitectura de servicio de LLM: agrupamiento de solicitudes (*batching*), *batching* continuo/dinámico, y el papel de la caché KV en el rendimiento de la decodificación incremental Pruebas de Rendimiento y Evaluación Comparativa. ▪ Cuantización de modelos (por ejemplo, cuantización post-entrenamiento int8/int4) y sus compensaciones entre huella de memoria, latencia y calidad de salida Entrenamiento y Optimización de Redes Neuronales. ▪ Estrategias de caché para sistemas de servicio de LLM: caché de prompt/respuesta, caché semántica, y reutilización de prefijo/caché KV entre solicitudes. ▪ Compensaciones de optimización de costo y latencia: selección del tamaño del modelo, elección del acelerador de hardware, y enrutamiento de solicitudes entre niveles de modelos Pruebas de Rendimiento y Evaluación Comparativa.	▪ Explicar cómo el *batching* continuo y la reutilización de la caché KV mejoran el rendimiento de un sistema de servicio de LLM [Explicar] ▪ Aplicar una técnica de cuantización a un LLM preentrenado y medir las compensaciones resultantes de latencia, memoria y calidad [Aplicar] ▪ Diseñar una estrategia de caché para una carga de trabajo de servicio de LLM dada, con el fin de reducir llamadas de inferencia redundantes [Diseñar] ▪ Estimar el costo de servicio de una aplicación basada en LLM dado un volumen de solicitudes esperado, el tamaño del modelo y el hardware [Estimar]
Lecturas : [Kwo+23]	

Unidad 8: Seguridad en IA y Hardware (6 horas)	
Resultados esperados: AG-C02	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Impacto de la IA en la seguridad y la privacidad: usar la IA para reforzar las defensas y abordar las capacidades adversariales aumentadas debido a la IA Aplicaciones e Impacto Social de la IA, Aplicaciones Adicionales e Impacto Social de la IA, Sociedad, ética y Profesión, Diseño de Sistemas. ▪ Riesgos de seguridad en la construcción y operación de sistemas de IA/aprendizaje automático (por ejemplo, sesgo algorítmico, sesgo del corpus de conocimiento, sesgo del corpus de entrenamiento, violación de derechos de autor) Aplicaciones e Impacto Social de la IA, Aplicaciones Adicionales e Impacto Social de la IA, Formulación del Aprendizaje Automático y Evaluación, Aprendizaje Profundo, RL y Ética, Introducción a la IA: Conceptos y Comportamiento Inteligente, Características del Problema y Naturaleza de los Agentes, Fundamentos de Aprendizaje Automático, Aprendizaje Automático: Datos, Representaciones y Evaluación, Redes Neuronales y Ética. ▪ Consideraciones éticas profesionales al diseñar sistemas seguros y mantener la privacidad; hacking ético Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. ▪ Implicaciones éticas de tecnologías emergentes (IA, automatización, privacidad de datos) ▪ Inyección de prompts directa e indirecta: instrucciones controladas por un atacante, incrustadas en la entrada del usuario o en contenido recuperado/retornado por una herramienta, que secuestran el comportamiento de una aplicación basada en LLM Inyección y Validación de Entrada. ▪ Gobierno de seguridad (security governance): objetivos organizacionales y evaluación general de riesgos.	▪ Mostrar cómo un adversario podría usar algoritmos de aprendizaje automático para reducir la seguridad de un sistema [Analizar] ▪ Aplicar un conjunto de regulaciones de privacidad para diseñar un sistema que proteja la privacidad [Aplicar] ▪ Evaluar las dimensiones éticas de un proyecto con impactos sociales o ambientales significativos en un contexto cultural diferente [Evaluar] ▪ Diseñar mitigaciones (jerarquía de instrucciones, filtrado de entrada/salida, acceso a herramientas con mínimo privilegio) para reducir la superficie de ataque de inyección de prompts de una aplicación LLM [Diseñar]
Lecturas : [OWA25], [Wei+22b]	

Unidad 9: Capstone de Aplicación Basada en LLM (6 horas)	
Resultados esperados: AG-C02,AG-C09	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
▪ Integrar un componente de prompting/RAG/agéntico, una capa de servicio y un bucle de evaluación en una sola aplicación de extremo a extremo basada en LLM Prácticas de Codificación. ▪ Refinamiento iterativo guiado por evaluación de una aplicación basada en LLM: usar los resultados de evaluación y la retroalimentación de usuarios para guiar cambios en los prompts, la recuperación o el ajuste fino Construcción a Escala y Proceso. ▪ Consideraciones de despliegue para una aplicación basada en LLM: empaquetado, monitoreo en producción, y un plan de reversión/respaldo ante fallas del modelo o la infraestructura.	▪ Integrar componentes de prompting, recuperación y/o ajuste fino con una capa de servicio en una aplicación de extremo a extremo basada en LLM que funcione [Integrar] ▪ Usar los resultados de evaluación y la retroalimentación de usuarios para refinar iterativamente una aplicación basada en LLM desplegada [Usar] ▪ Planear una estrategia de despliegue y reversión para una aplicación basada en LLM, considerando el monitoreo y los modos de falla [Planear]
Lecturas : [Alt24]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

Sesiones Teóricas:
Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

Sesiones de Laboratorio:
Para verificar que los alumnos hayan alcanzado el logro planteado para cada una de las unidades de aprendizaje, realizarán actividades que les permita aplicar los conocimientos adquiridos durante las sesiones de teoría y se les propondrá retos que permitan evaluar el desempeño de los alumnos.

Exposiciones individuales o grupales:
Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

Lecturas:
A lo largo del curso se proporcionan diferentes lecturas, las cuales podrían ser evaluadas. El uso de herramientas Online permite a cada estudiante acceder a la información del curso, e interactuar fuera de aula con el profesor y con los otros estudiantes.

La nota final NF depende de de las siguientes notas:

- EP: Examen Parcial.
- EF: Examen Final.
- P: Evaluacion permanente (Promedio de prácticas y Trabajos calificados).

La nota final NF se calcula así:

$$NF = \frac{EP + EF + P}{3}$$

10. BIBLIOGRAFÍA BÁSICA

[Vas+17] Ashish Vaswani et al. ■Attention Is All You Need■. En: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[Lew+20] Patrick Lewis et al. ■Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks■. En: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Hu+22] Edward J. Hu et al. ■LoRA: Low-Rank Adaptation of Large Language Models■. En: *International Conference on Learning Representations (ICLR)*. 2022.

[Wei+22a] Jason Wei et al. ■Chain-of-Thought Prompting Elicits Reasoning in Large Language Models■. En: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[Wei+22b] Laura Weidinger et al. ■Taxonomy of Risks posed by Language Models■. En: *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*. 2022.

[Det+23] Tim Dettmers et al. ■QLoRA: Efficient Finetuning of Quantized LLMs■. En: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[Kwo+23] Woosuk Kwon et al. ■Efficient Memory Management for Large Language Model Serving with PagedAttention■. En: *ACM Symposium on Operating Systems Principles (SOSP)*. 2023.

[LBL+23] Percy Liang, Rishi Bommasani, Tony Lee et al. ■Holistic Evaluation of Language Models■. En: *Transactions on Machine Learning Research (TMLR)* (2023).

[Whi+23] Jules White et al. ■A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT■. En: *arXiv preprint arXiv:2302.11382* (2023).

[Yao+23] Shunyu Yao et al. ■ReAct: Synergizing Reasoning and Acting in Language Models■. En: *International Conference on Learning Representations (ICLR)*. 2023.

[Alt24] Valentina Alto. *Building LLM Powered Applications: Create Intelligent Apps and Agents with Large Language Models*. Packt Publishing, 2024.

[Gao+24] Yunfan Gao et al. ■Retrieval-Augmented Generation for Large Language Models: A Survey■. En: *arXiv preprint arXiv:2312.10997* (2024).

[Ras24] Sebastian Raschka. *Build a Large Language Model (From Scratch)*. Manning Publications, 2024.

[OWA25] OWASP Foundation. *OWASP Top 10 for Large Language Model Applications*. `https://owasp.org/www-project-top-10-for-large-language-model-applications/`. 2025.