Temario del Curso

Introducción a los Large Language Models

Universidad Nacional Autónoma de México

Facultad de Ciencias

Materia: Proyecto I

Carrera: Matemáticas Aplicadas (Plan 2017)

Grupo: 6012
Modalidad: Presencial
Horario: Lunes a Viernes, 17:00 a 18:00
Profesor: Francisco Peez Carbajal
Ayudante: Jose Eduardo Rodriuez Barrios

Objetivo del Seminario-Taller

Este seminario-taller tiene como propósito que los estudiantes **desarrollen un proyecto de titulación** en el área de modelos de lenguaje grande (LLMs), integrando fundamentos teóricos con aplicaciones prácticas. El curso busca fomentar una comprensión profunda de los LLMs, desde su arquitectura hasta su implementación en tareas específicas.

A través de la lectura crítica de artículos científicos, la síntesis de conocimiento especializado y el desarrollo práctico de un proyecto, los estudiantes:

- **Explorarán** el estado del arte en modelos de lenguaje, desde enfoques neuronales iniciales hasta los LLMs contemporáneos.
- **Diseñarán e implementarán** un proyecto aplicado que resuelva una tarea específica (clasificación, generación, resumen, o chatbot).
- **Desarrollarán** una solución integral, desde la preparación del corpus hasta el despliegue del modelo.
- **Experimentarán** con arquitecturas avanzadas y técnicas de ajuste eficiente (LoRA, QLoRA, PEFT).
- **Documentarán** su trabajo con rigor técnico y científico, siguiendo estándares académicos.
- **Presentarán** sus avances y resultados de manera profesional, con énfasis en la comunicación efectiva.

Alcance y Enfoque

Este curso cubre el **espectro completo de los modelos modernos de lenguaje**:

- **Transformers y Arquitectura Base:** mecanismos de atención, embeddings y entrenamiento auto-supervisado
- **Modelos fundacionales:** BERT, GPT, T5, LLaMA, Falcon, Mistral
- **Fine-tuning eficiente:** LoRA, QLoRA, prompt-tuning, adapters
- **Evaluación y fairness:** métricas de desempeño, sesgos y responsabilidad ética
- **Aplicaciones:** chatbots, clasificación, resumen, generación de texto y RAG (Retrieval-Augmented Generation)

Formato del Curso

El curso se estructura como un **seminario-taller interactivo**, donde:

- Los estudiantes analizan y presentan artículos científicos clave, fomentando el pensamiento crítico.

- Se realizan discusiones grupales para conectar conceptos teóricos con aplicaciones prácticas.
- Cada estudiante desarrolla un proyecto individual, con mentoría personalizada en cada fase.
- Las sesiones incluyen retroalimentación continua para garantizar el progreso del proyecto.
- El proyecto final se diseña para ser un trabajo de titulación, con estándares de calidad académica.

Pre-requisitos

Para un aprovechamiento óptimo del curso, se requiere que los participantes cumplan con los siguientes requisitos:

Conocimientos Matemáticos

- **Álgebra lineal:** vectores, matrices y transformaciones lineales
- **Probabilidad y estadística:** nociones básicas
- **Conocimientos de regresión logística y aprendizaje de máquina:** deseable

Habilidades Técnicas

- Capacidad para leer artículos técnicos en inglés
- Programación en Python
- Familiaridad con herramientas como PyTorch, Hugging Face Transformers, y GitHub

Herramientas de Apoyo

- Aula virtual (Google Classroom)
- Repositorio individual en GitHub para código y reportes

Estructura del Seminario-Taller

El seminario-taller se organiza en cuatro fases, integrando teoría y práctica de manera progresiva:

Fase 1: Fundamentos y Estado del Arte (Semanas 1–4)

Objetivo: Comprender los principios fundamentales de los LLMs y su evolución histórica.

- Lectura y análisis de papers seminales: *Attention is All You Need* (Vaswani et al., 2017), *BERT* (Devlin et al., 2019), *GPT* (Brown et al., 2020).
- Elaboración de fichas bibliográficas y mapas conceptuales para sintetizar el conocimiento.
- Discusión sobre capacidades emergentes en LLMs a gran escala (Zhao et al., 2023).

Desarrollo del proyecto:

- Definición del problema y tarea lingüística (clasificación, QA, resumen, etc.).
- Recolección y exploración del corpus de texto.
- Configuración del entorno de desarrollo.
- Establecimiento de baselines con modelos preentrenados.

Fase 2: Procesamiento, Fine-tuning y Experimentación (Semanas 5–8)

Objetivo: Implementar y experimentar con técnicas avanzadas de ajuste eficiente.

- Lectura y análisis de técnicas de tokenización y fine-tuning (LoRA, QLoRA, adapters).
- Implementación del pipeline de entrenamiento y experimentación con modelos recientes (LLaMA, Falcon, Mistral).
- Discusión sobre optimización de hiperparámetros y estrategias de validación.

Desarrollo del proyecto:

- Comparación de arquitecturas y técnicas de ajuste eficiente.
- Documentación de resultados y análisis crítico de experimentos.

Fase 3: Evaluación, Interpretabilidad y Fairness (Semanas 9–12)

Lecturas y síntesis:

- Métricas: perplejidad, BLEU, ROUGE, F1, exactitud

- Interpretabilidad: visualización de atención y saliency maps
- Fairness y ética: sesgos en LLMs (Gallegos et al., 2023), alucinaciones (Huang et al., 2024) y uso responsable
- Análisis de robustez y generalización en diferentes dominios

Desarrollo del proyecto:

- Evaluación cuantitativa y cualitativa del modelo
- Análisis de errores y diagnóstico
- Estudio de sesgos y robustez del modelo
- Implementación de técnicas de interpretabilidad
- Comparación con baselines y modelos del estado del arte

Fase 4: Deployment, Documentación y Presentación (Semanas 13–16)

Lecturas y síntesis:

- Despliegue de modelos: APIs con FastAPI, Streamlit o Gradio
- Técnicas de optimización y compresión (quantization, pruning)
- Casos de estudio: RAG (Lewis et al., 2020), agentes y pipelines de inferencia
- Consideraciones éticas y sostenibilidad de los LLMs

Desarrollo del proyecto:

- Selección y optimización del modelo final
- Implementación de un demo funcional (API/chatbot/app)
- Aplicación de técnicas de compresión y optimización
- Elaboración del reporte técnico (formato de tesis, 50–80 páginas)
- Presentación final y defensa del proyecto ante el grupo

Temas de Ayudantía

La ayudantía del curso está diseñada para proporcionar **soporte técnico y práctico** a los estudiantes en el desarrollo de su proyecto. Las sesiones de ayudantía se enfocan en herramientas, técnicas y buenas prácticas esenciales para la implementación exitosa del proyecto de titulación.

Herramientas y Entorno de Desarrollo

1. **Git y GitHub desde cero**
   Control de versiones, manejo de repositorios, branching, commits y colaboración en proyectos.
2. **Markdown y documentación**
   Sintaxis Markdown para README, documentación de código y reportes técnicos.
3. **Docker y contenedores para ML**
   Creación de ambientes reproducibles, dockerización de modelos y despliegue containerizado.
4. **Entorno de desarrollo Python**
   Configuración de ambientes virtuales, gestión de dependencias (pip, conda), notebooks y IDEs.
5. **Google Colab y recursos GPU**
   Uso efectivo de Colab, gestión de sesiones, acceso a GPU/TPU y mejores prácticas.

Bibliotecas y Frameworks

6. **Primeros pasos con Hugging Face**
   Introducción al ecosistema: Hub, modelos preentrenados, tokenizers y configuración básica.
7. **Introducción a Ollama**
   Instalación, configuración y ejecución local de modelos de lenguaje.
8. **Uso de Hugging Face**
   Pipelines, fine-tuning, evaluación de modelos y manejo de datasets.
9. **Casos de uso y personalización de Ollama**
   Implementación de casos prácticos, customización de modelos y optimización.

Desarrollo de Aplicaciones

10. **Pipeline de datos**
    Preprocesamiento, limpieza, tokenización y manejo eficiente de datasets grandes.
11. **FastAPI para modelos**
    Creación de APIs REST para servir modelos, endpoints, validación y documentación automática.

12. **Gradio/Streamlit**
    Desarrollo de interfaces interactivas para demostración de modelos y prototipos.

Documentación Final

13. **Preparación de presentación final (LaTeX)**
    Estructuración de documentos académicos, manejo de bibliografía, figuras y formato de tesis.

Formato de las Ayudantías

Las sesiones de ayudantía se llevan a cabo de manera complementaria al curso principal y están diseñadas para:

- Resolver dudas técnicas específicas sobre implementación
- Proporcionar tutoriales prácticos de herramientas clave
- Ofrecer mentoría personalizada para cada proyecto
- Facilitar la resolución de problemas de código y debugging
- Apoyar en la configuración de ambientes y resolución de dependencias

Evaluación del Seminario-Taller

La evaluación se centra en el **desarrollo integral del proyecto de titulación** y la **apropiación de conocimiento** a través de actividades prácticas y teóricas.

Actividad	Fase	Descripción	%
Presentaciones de Papers	1	Análisis crítico y síntesis de artículos relevantes, conectando conceptos teóricos con el proyecto.	20 %
Avances del Proyecto	2–3	Entregas parciales: definición del problema, implementación baseline, experimentación y despliegue.	30 %
Documentación Técnica	1–4	Reporte progresivo con estructura de tesis, incluyendo metodología, experimentos y análisis.	20 %
Proyecto Final	4	Producto funcional completo: modelo entrenado, sistema desplegado y presentación profesional.	30 %

Cuadro 1: Distribución de la evaluación del seminario-taller

Componentes del Proyecto de Titulación

El proyecto final debe ser un trabajo completo y de calidad suficiente para servir como tesis, incluyendo:

1. **Revisión bibliográfica** exhaustiva del estado del arte en LLMs
2. **Planteamiento del problema** específico en procesamiento de lenguaje natural
3. **Metodología** detallada incluyendo preprocesamiento, arquitectura y fine-tuning
4. **Implementación** completa y reproducible (repositorio GitHub documentado)
5. **Experimentación** rigurosa comparando diferentes enfoques y técnicas
6. **Evaluación** completa incluyendo métricas, interpretabilidad y análisis de sesgos
7. **Sistema desplegado** funcional (API, chatbot o aplicación web)
8. **Documentación técnica** completa con formato de tesis (50–80 páginas)
9. **Análisis ético** y consideraciones de uso responsable

Criterios de Evaluación Específicos

Para Presentaciones de Papers:

- Comprensión profunda de conceptos técnicos y matemáticos
- Capacidad de síntesis y explicación clara de metodologías complejas

- Análisis crítico de fortalezas, limitaciones y contribuciones
- Conexión efectiva con el estado del arte y proyecto personal
- Participación activa en discusiones técnicas

Para el Proyecto:

- Rigor metodológico en experimentación y evaluación
- Implementación técnica correcta y eficiente
- Originalidad en el planteamiento o enfoque del problema
- Calidad de la documentación técnica y reproducibilidad
- Análisis crítico de resultados y comparación con literatura

Políticas del Seminario-Taller

- La calificación mínima aprobatoria es de 60 %
- Se requiere asistencia mínima del 80 % y participación activa
- Los avances deben entregarse en las fechas establecidas
- El proyecto es individual, aunque se fomenta la discusión colaborativa
- Cada estudiante debe mantener un repositorio de GitHub actualizado
- Las presentaciones de papers son obligatorias y se asignan rotativamente
- Se espera honestidad académica; el plagio resultará en reprobación automática

Bibliografía

Artículos Científicos Fundamentales

Arquitecturas Fundacionales

1. **Vaswani, A., et al.** (2017). *Attention is All You Need.* NeurIPS.
   Paper fundamental que introduce la arquitectura Transformer basada únicamente en mecanismos de atención.

2. **Devlin, J., et al.** (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* NAACL.
   Modelo bidireccional que revoluciona la comprensión de texto mediante pre-entrenamiento masivo.

3. **Brown, T., et al.** (2020). *Language Models are Few-Shot Learners.* NeurIPS.
   GPT-3 con 175B parámetros demuestra capacidades de few-shot learning sin fine-tuning adicional.

4. **Raffel, C., et al.** (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* JMLR.
   T5 unifica todas las tareas de NLP como problemas texto-a-texto.

Modelos Fundacionales Recientes

5. **Touvron, H., et al.** (2023). *LLaMA: Open and Efficient Foundation Language Models.* arXiv.
   Familia de modelos abiertos (7B-65B) que compite con modelos propietarios más grandes.

6. **Almazrouei, E., et al.** (2023). *The Falcon Series of Open Language Models.* arXiv.
   Modelos Falcon entrenados en datos curados con rendimiento competitivo y licencia abierta.

7. **Jiang, A. Q., et al.** (2023). *Mistral 7B.* arXiv.
   Modelo de 7B parámetros con innovaciones arquitectónicas que supera a modelos más grandes.

Técnicas de Fine-tuning Eficiente

8. **Hu, E. J., et al.** (2021). *LoRA: Low-Rank Adaptation of Large Language Models.* ICLR.
   Técnica de adaptación de bajo rango que reduce drásticamente parámetros entrenables.

9. **Dettmers, T., et al.** (2023). *QLoRA: Efficient Finetuning of Quantized LLMs.* arXiv.
   Combina quantización 4-bit con LoRA para fine-tuning eficiente en hardware limitado.

10. **Li, X. L., & Liang, P.** (2021). *Prefix-Tuning: Optimizing Continuous Prompts for Generation.* ACL.
    Método que optimiza vectores de prefijo continuos manteniendo parámetros del modelo fijos.

11. **Houlsby, N., et al.** (2019). *Parameter-Efficient Transfer Learning for NLP.* ICML.
Introducción de adapters como módulos ligeros insertados entre capas del modelo.

Evaluación, Interpretabilidad y Ética

12. **Zhao, W. X., et al.** (2023). *A Survey of Large Language Models.* arXiv.
Revisión comprehensiva del estado del arte, capacidades emergentes y desafíos de los LLMs.

13. **Gallegos, I. O., et al.** (2023). *Bias and Fairness in Large Language Models: A Survey.* arXiv.
Análisis exhaustivo de sesgos en LLMs y métodos de evaluación y mitigación.

14. **Huang, L., et al.** (2024). *A Survey on Hallucination in Large Language Models.* arXiv.
Estudio sistemático del problema de alucinaciones y estrategias de mitigación.

Aplicaciones Avanzadas

15. **Lewis, P., et al.** (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS.
RAG combina generación con recuperación de información para mejorar factualidad.

16. **Sennrich, R., et al.** (2016). *Neural Machine Translation of Rare Words with Subword Units.* ACL.
Introducción del algoritmo BPE para manejo de vocabulario abierto en NLP.

Libros de Texto y Referencias

17. **Tunstall, L., von Werra, L., & Wolf, T.** (2022). *Natural Language Processing with Transformers.* O'Reilly.
Guía práctica para implementación de Transformers con Hugging Face.

18. **Jurafsky, D., & Martin, J.** (2023). *Speech and Language Processing (3rd ed. draft).* Stanford University.
Texto fundamental de procesamiento de lenguaje natural.

19. **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning.* MIT Press.
Referencia fundamental sobre deep learning (disponible en línea).

Recursos Técnicos y Documentación

20. **Hugging Face Transformers Documentation:** `https://huggingface.co/docs/transformers/`
Documentación oficial de la biblioteca más utilizada para LLMs.

21. **Papers with Code – NLP Section:** https://paperswithcode.com/area/natural-language-processing
    Plataforma con papers, implementaciones y benchmarks actualizados.
22. **arXiv sections:** cs.CL (Computation and Language), cs.LG (Machine Learning)
    Repositorio de preprints con los avances más recientes en el campo.
23. **OpenAI Research:** `https://openai.com/research/`
    Publicaciones y avances de uno de los laboratorios líderes en LLMs.

Recursos de Implementación

24. **PyTorch Documentation:** `https://pytorch.org/docs/stable/index.html`
25. **Weights & Biases:** `https://wandb.ai/` (Para experiment tracking)
26. **MLflow:** `https://mlflow.org/` (Plataforma MLOps alternativa)
27. **FastAPI:** `https://fastapi.tiangolo.com/` (Para APIs de despliegue)
28. **Gradio:** `https://gradio.app/` (Para interfaces de usuario rápidas)