



Tecnológico de Monterrey

Maestría en Inteligencia Artificial Aplicada

Proyecto Integrador

Dra. Grettel Barceló Alonso

Dr. Luis Eduardo Falcón Morales

Mtra. Verónica Sandra Guzmán de Valle

Conformación de equipos y planteamiento del proyecto

Jonathan Garza Bennet | A01793038

Rafael José Mateo Comprés | A01793054

Fernando Maytorena Espinosa de los Monteros | A01110946

Domingo 12 de mayo de 2024

Índice

Introducción.....	4
Adaptando LLMs a Nuevos Dominios.....	5
2.1. Etapas de Adaptación de Dominio	5
2.1.1. Domain Adaptive Pretraining (DAPT).....	5
2.1.2. Model Alignment	5
2.2. Parameter-Efficient Finetuning (PEFT)	6
2.2.1. Motivación y Ventajas.....	6
2.2.2. Métodos PEFT	6
2.2.3. Enfoque de Dos Etapas: Combinando Adapters para DAPT y Tareas Downstream.....	7
Feature Engineering y Dataset para DAPT	8
3.1. Recopilación y Procesamiento de Datos de Dominio.....	8
3.2. Procesamiento de Datos para nuestro Proyecto.....	8
3.2.1. Descripción del Dataset Existente	8
3.2.2. Limpieza y Filtrado de Datos	8
3.2.3. Tokenización y Formatos de Input.....	9
3.3. Entrenando el Adapter de DAPT.....	9
Feature Engineering y Dataset para Model Alignment	10
4.1. Dataset de Instrucciones.....	10
4.2. Modelo Bonito para Generación de Instrucciones.....	10
Model Alignment con Instruction Tuning.....	16
5.1. Adaptación a Tareas Específicas.....	16
5.2. Entrenando el Adapter de Tareas Downstream	16
5.3. Combinando los Adapters de DAPT y Tareas Downstream.....	17
Propuesta de Modelo y Experimentos	17
6.1. Arquitectura Propuesta.....	17
Evaluación y Discusión	17
7.1. Métricas de evaluación.....	17

Conclusiones y Trabajo Futuro	17
Bibliografía	18

Introducción

Los Modelos de Lenguaje de Gran Escala (LLMs, por sus siglas en inglés) han demostrado un desempeño impresionante en una amplia gama de tareas de Procesamiento de Lenguaje Natural (NLP) (Brown et al., 2020; Chowdhery et al., 2022). Estos modelos son típicamente entrenados en grandes cantidades de texto no estructurado de diversas fuentes, lo que les permite capturar conocimiento general y habilidades de lenguaje robustas (Devlin et al., 2019; Liu et al., 2019). Sin embargo, cuando se trata de dominios especializados como la medicina, el derecho o la ingeniería, los LLMs genéricos a menudo se quedan cortos en comparación con modelos específicos de dominio (Beltagy et al., 2019; Gururangan et al., 2022).

La adaptación de LLMs a nuevos dominios es un área de investigación activa y de gran importancia práctica. Permite aprovechar el vasto conocimiento capturado por estos modelos y aplicarlo a tareas específicas de dominio, mejorando así el desempeño y abriendo nuevas posibilidades de aplicación (Xu et al., 2022). Sin embargo, dado el enorme tamaño de los LLMs modernos, con cientos de miles de millones de parámetros (Brown et al., 2020; Chowdhery et al., 2022), la adaptación eficiente de estos modelos presenta desafíos significativos en términos de costo computacional y de almacenamiento (Liu et al., 2019).

En este trabajo, proponemos un framework para la adaptación eficiente de LLMs a nuevos dominios, centrándonos en el dominio legal mexicano. Nuestro enfoque combinará técnicas de vanguardia en Finetuning Eficiente en Parámetros (PEFT, por sus siglas en inglés) (He et al., 2021), generación de conjuntos de instrucciones (Mishra et al., 2022) y alineación de modelo con Instruction Tuning (Sanh et al., 2022)

Nuestro proyecto busca realizar las siguientes contribuciones:

1. Desarrollar un framework integral para la adaptación eficiente de dominio de LLMs, que combina PEFT, generación de instrucciones y alineación de modelos.
2. Un LLM adaptado al dominio legal mexicano, que establezca un nuevo estado del arte en tareas clave de este dominio.
3. Un análisis de las ventajas y compensaciones de diferentes métodos PEFT para la adaptación de dominio, con un enfoque en la eficiencia computacional.
4. Conocimientos prácticos sobre las mejores prácticas para la generación de conjuntos de instrucciones y la alineación de modelos para LLMs adaptados.

Adaptando LLMs a Nuevos Dominios

El proceso de adaptar un LLM pre-entrenado a un nuevo dominio generalmente implica dos etapas principales: 1) Domain Adaptive Pretraining (DAPT) y 2) Model Alignment (MA) (Gururangan et al., 2020)

2.1. Etapas de Adaptación de Dominio

2.1.1. Domain Adaptive Pretraining (DAPT)

DAPT implica continuar el pre-entrenamiento del LLM en un gran corpus de texto específico del dominio objetivo (Gururangan et al., 2020). Esto permite al modelo capturar información específica del dominio y ajustar sus representaciones internas en consecuencia. Se ha demostrado que DAPT mejora significativamente el rendimiento en una variedad de tareas específicas de dominio, como clasificación de texto clínico (Gururangan et al., 2020), análisis de sentimientos en finanzas (Araci, 2019) y muchas otras.

Sin embargo, DAPT también presenta algunos desafíos. Primero, requiere un gran conjunto de datos de texto específico del dominio, que puede ser difícil y costoso de obtener. Segundo, el pre-entrenamiento continuo de LLMs de gran escala es computacionalmente intensivo, a menudo requiriendo días o semanas en hardware especializado (Li et al., 2021). Finalmente, existe el riesgo de sobre-especializar el modelo al nuevo dominio, potencialmente degradando su rendimiento en tareas más generales - un fenómeno conocido como "olvido catastrófico" (Luo et al., 2024; McCloskey & Cohen, 1989).

2.1.2. Model Alignment

El objetivo de la etapa de *Model Alignment* es adaptar aún más el LLM a las tareas objetivo (*downstream tasks*) dentro del dominio de interés. Esto generalmente implica hacer finetuning al modelo en un conjunto más pequeño de ejemplos etiquetados para cada tarea (Xu et al., 2023). Sin embargo, dado el gran número de parámetros en los LLMs, el finetuning estándar a menudo resulta en sobreentrenamiento (*overfitting*), especialmente cuando los conjuntos de datos de las tareas objetivo son pequeños (Zhang et al., 2021).

Un desarrollo reciente prometedor en alineación de modelos es el *Instruction Tuning* (Ouyang et al., 2022; Sanh et al., 2022; Wei et al., 2023). En *Instruction Tuning*, el modelo es afinado en un conjunto de datos de "instrucciones", que consisten en pares de entradas en lenguaje natural y salidas deseadas. Esto permite un ajuste más eficiente del modelo para seguir instrucciones y realizar tareas específicas. *Instruction Tuning* ha demostrado una fuerte capacidad de generalización y transferencia a tareas no vistas durante el entrenamiento (Mishra et al., 2022).

2.2. Parameter-Efficient Finetuning (PEFT)

2.2.1. Motivación y Ventajas

Dadas las limitaciones del Domain Adaptive Pretraining (DAPT) por medio del pre-entrenamiento continuo discutidas anteriormente, los métodos PEFT han surgido como una alternativa prometedora para la adaptación eficiente de LLMs (He et al., 2021; Hu et al., 2022). La idea central de PEFT es agregar un pequeño número de parámetros entrenables al modelo pre-entrenado y actualizar solo esos parámetros durante el finetuning, manteniendo fijos los pesos originales (Houlsby et al., 2019).

Las principales ventajas de PEFT sobre el finetuning completo son:

- **Eficiencia Computacional:** Al actualizar solo una fracción de los parámetros, PEFT reduce drásticamente el costo computacional y de almacenamiento del finetuning (Li et al., 2021). Esto permite adaptar LLMs mucho más grandes en hardware más modesto.
- **Evitar el Overfitting:** Dado que el número de parámetros entrenables es mucho menor, PEFT es menos propenso al overfitting, incluso con conjuntos de datos pequeños (He et al., 2021). Esto es especialmente relevante para dominios especializados donde los datos etiquetados son escasos.
- **Preservar el Conocimiento Pre-entrenado:** Al mantener la mayor parte de los pesos del modelo fijos, PEFT preserva mejor las habilidades aprendidas durante el pre-entrenamiento (Hu et al., 2022). Esto mitiga el riesgo de "olvido catastrófico" y permite una transferencia más efectiva al nuevo dominio.
- **Modularidad y Composicionalidad:** Muchos métodos PEFT, como Adapters (Houlsby et al., 2019) y LoRA (Hu et al., 2022), aprenden módulos independientes para cada tarea o dominio. Estos pueden ser fácilmente compuestos o intercambiados, permitiendo una gran flexibilidad y reusabilidad (Karimi Mahabadi et al., 2021; Pfeiffer et al., 2020).

2.2.2. Métodos PEFT

Dentro de PEFT, se han propuesto varios métodos diferentes que varían en términos de cómo se agregan y entrenan los parámetros adicionales. Revisamos brevemente algunos de los más populares (para un estudio más detallado ver (Xu et al., 2023)).

- **Adapters (Houlsby et al., 2019; Pfeiffer et al., 2020):** Los Adapters insertan capas de proyección de rango completo en cada capa del modelo pre-entrenado. Solo los parámetros del Adapter son entrenados durante el finetuning.

- LoRA (Hu et al., 2022): LoRA aprende matrices de rango bajo que se suman a los pesos originales en cada capa. Esto permite una actualización de grano fino con muy pocos parámetros adicionales.
- Prefix-Tuning (Li & Liang, 2021): Prefix-Tuning prepende un conjunto pequeño de "tokens de prefijo" entrenables a la secuencia de entrada. Solo los embeddings de los tokens de prefijo son actualizados.
- BitFit (Zaken et al., 2021): BitFit entrena solo las bias terms en el modelo pre-entrenado, dejando los otros parámetros fijos. A pesar de su simplicidad, puede lograr un desempeño sorprendentemente fuerte.
- Compacter (Mahabadi et al., 2021): Compacter combina Adapters con matrices de reparametrización hipercompleja de rango bajo para una mayor eficiencia de parámetros.
- QLoRA (Dettmers et al., 2022): QLoRA es una extensión de LoRA que cuantiza el modelo base a baja precisión (4-8 bits) mientras mantiene los pesos LoRA en precisión completa (FP32). Esto permite una mayor eficiencia computacional y de memoria durante el ajuste y la inferencia, al tiempo que mantiene un rendimiento comparable al LoRA de precisión completa. QLoRA es particularmente adecuado para ajustar LLMs muy grandes en hardware limitado.

Para el desarrollo de nuestro proyecto, debemos definir cuál de las técnicas PEFT sería más apta para el entrenamiento de nuestro modelo dadas las limitantes existentes de recursos y hardware.

2.2.3. Enfoque de Dos Etapas: Combinando Adapters para DAPT y Tareas Downstream

Un desarrollo reciente y prometedor en la adaptación eficiente de dominio de LLMs es el uso de un enfoque de dos etapas que combina adapters PEFT para el conocimiento de dominio y las tareas downstream (Gema et al., 2023). La idea central es entrenar primero un adapter PEFT en un corpus de dominio grande para capturar las características y el conocimiento generales del dominio (etapa de DAPT). Luego, se entrena un segundo adapter PEFT en un conjunto de datos de instrucciones orientado a tareas específicas de downstream (etapa de Model Alignment).

Durante la inferencia, ambos adapters se combinan con el modelo base para producir un LLM adaptado al dominio y a las tareas. Este enfoque aprovecha los beneficios de eficiencia de los métodos PEFT mientras permite una adaptación flexible y modular a diferentes dominios y tareas.

Gema et al. (2023) demuestran la efectividad de este enfoque en el dominio clínico, utilizando LLaMA como modelo base. Entrenan un "Clinical LLaMA-LoRA" adapter en un gran corpus de notas clínicas para la adaptación de dominio, y un "Downstream LLaMA-LoRA" adapter en un conjunto de datos de instrucciones para tareas clínicas específicas. La combinación de estos

adapters con LLaMA base logra un rendimiento superior en varias tareas clínicas de downstream, superando a los modelos de referencia.

En nuestro proyecto, exploraremos un enfoque similar para el dominio legal mexicano. En la Sección 3, discutimos el Feature Engineering y Dataset Creation para entrenar nuestro adapter de DAPT. En la Sección 5, cubrimos el entrenamiento de nuestro adapter de tareas downstream. Finalmente, demostraremos el desempeño obtenido al combinar un modelo base, un adaptador de dominio, y un adaptador de tareas objetivo en nuestros experimentos.

Feature Engineering y Dataset para DAPT

3.1. Recopilación y Procesamiento de Datos de Dominio

La recopilación y procesamiento de datos de dominio de alta calidad es fundamental para el éxito de la adaptación de dominio de LLMs. Los datos deben ser relevantes, diversos y representativos del dominio objetivo. Las fuentes de datos comunes incluyen literatura académica, informes técnicos, manuales, sitios web y foros especializados.

El proceso de recopilación de datos a menudo implica técnicas como web scraping, extracción de texto de PDFs o imágenes, y llamadas a APIs de bases de datos. Es importante considerar los aspectos legales y éticos de la recopilación de datos, como los derechos de autor y la privacidad.

3.2. Procesamiento de Datos para nuestro Proyecto

3.2.1. Descripción del Dataset Existente

Para nuestro proyecto, utilizaremos un dataset existente que encontramos en MDPI Open Access Journal, perteneciente al research paper titulado “Unified, Labeled, and Semi-Structured Database of Pre-Processed Mexican Laws”. Este dataset contiene más de 300 folders, donde cada folder representa una ley mexicana a nivel federal, y los cuales contienen al menos una variedad de archivos de tipo PDF, JSON y TXT. Aunque esto nos ahorra un esfuerzo significativo en la recopilación de datos, aún necesitamos procesar y preparar los datos para el pre-entrenamiento de DAPT.

3.2.2. Limpieza y Filtrado de Datos

El primer paso en el procesamiento de nuestro dataset es la limpieza y el filtrado de datos. Esto implica tareas como:

- Eliminación de caracteres o secciones no deseadas (por ejemplo, marcas de formato, enlaces, etc.)
- Manejo de puntuación y casos especiales (por ejemplo, fórmulas matemáticas, código fuente, etc.)

- Filtrado de documentos irrelevantes o de baja calidad
- Deduplicación de contenido

Para esta etapa, utilizaremos bibliotecas de Python como Pandas, SpaCy, NumPy, Regular Expressions y Natural Language Toolkit (nltk), y escribiremos scripts personalizados para automatizar el proceso de limpieza.

3.2.3. Tokenización y Formatos de Input

Después de limpiar los datos, necesitamos tokenizarlos y formatearlos para el pre-entrenamiento. El tokenizador a utilizar dependerá en gran medida del modelo que seleccionemos para el entrenamiento, algunos ejemplos relevantes son Byte-Pair Encoding (BPE), Byte-level BPE, WordPiece, Unigram o SentencePiece.

Para el pre-entrenamiento, preprocesaremos los documentos en archivos JSONL, que es el formato esperado por una gran mayoría de modelos open-source como los modelos de la familia Mistral o Llama. Esto implica tareas como dividir los documentos en trozos de longitud fija, agregar tokens especiales, y crear ejemplos de pre-entrenamiento (por ejemplo, pares de oraciones adyacentes).

3.3. Entrenando el Adapter de DAPT

Con nuestro dataset de dominio preparado, el siguiente paso será entrenar nuestro *adapter* de DAPT para capturar las características y el conocimiento del dominio legal mexicano. Para ello, utilizaremos el enfoque PEFT para este propósito, por medio de una de las técnicas descritas en la sección 2.2.2.

Primero, debemos inicializar [DAPT_ADAPTER] con un set de hiperparámetros elegidos (por ejemplo, rango para LoRA, dimensión para Adapters, etc.). Luego, agregaremos este *adapter* a cada capa del modelo base. Durante el entrenamiento, congelamos los pesos del modelo base y solo actualizamos los parámetros de [DAPT_ADAPTER]. Esto nos permitirá capturar eficientemente el conocimiento específico del dominio con un número mínimo de parámetros adicionales.

Entrenamos el *adapter* utilizando nuestro dataset de dominio por un número bajo de épocas (de dos a cinco épocas debería ser suficiente para no sobreentrenar el *adapter*). Es muy importante realizar una serie de experimentos para encontrar el set óptimo de hiperparámetros como el tipo de optimizador, la tasa de aprendizaje, o el uso de decaimiento de la tasa de aprendizaje. Monitoreamos la pérdida del lenguaje en un conjunto de validación para evitar el sobreajuste.

El adaptador entrenado, [DAPT_ADAPTER], puede entonces combinarse con el modelo base para tareas posteriores. Esta combinación nos permite aprovechar el conocimiento general del modelo base junto con las características específicas del dominio aprendidas por [DAPT_ADAPTER].

Feature Engineering y Dataset para Model Alignment

Una vez que hemos entrenado [DAPT_ADAPTER], el siguiente paso implica continuar con la alineación del modelo. Esto plantea dos desafíos principales: 1) Crear un conjunto de datos de instrucciones alineado al formato de entrada-salida esperado por nuestro modelo y 2) Manejar el volumen potencialmente grande de datos de texto no estructurados en el dominio objetivo. Abordamos cada uno de estos a continuación.

4.1. Dataset de Instrucciones

El formato de un dataset de instrucciones es especialmente adecuado para LLMs porque coincide con su paradigma de entrenamiento en lenguaje natural (Sanh et al., 2022; Wei et al., 2022). Un ejemplo de instrucción para un LLM adaptado al dominio legal podría verse así:

Instrucción: De acuerdo con la siguiente transcripción de la audiencia judicial, ¿cuál fue la conclusión del juez con respecto al caso de fraude fiscal?

Transcripción: "En la sesión de hoy, después de revisar las pruebas presentadas por ambas partes y escuchar los argumentos finales, el juez dictaminó que el acusado no pudo demostrar de manera convincente la legitimidad de las transacciones financieras en cuestión..."

Respuesta: Según la transcripción proporcionada, el juez concluyó que el acusado no pudo demostrar de manera convincente la legitimidad de las transacciones financieras en cuestión, lo que sugiere que el caso de fraude fiscal no fue probado a favor del acusado.

La instrucción proporciona un contexto y una tarea claros, mientras que la respuesta ilustra el formato y estilo deseados. Al ajustar un LLM a un conjunto diverso de estos pares de instrucción-respuesta, podemos enseñarle a seguir instrucciones y generar respuestas relevantes para una variedad de tareas.

Sin embargo, crear manualmente tales conjuntos de datos de instrucciones es costoso y consume mucho tiempo, especialmente si queremos cubrir una amplia gama de tareas en un dominio. Las anotaciones existentes tampoco se transfieren fácilmente a este formato. Por lo tanto, desarrollar métodos para generar automáticamente conjuntos de datos de instrucciones a partir de datos no estructurados es un área de investigación activa (Mishra et al., 2022; Wang et al., 2022; Wei et al., 2022).

4.2. Modelo para Generación de Instrucciones

Para generar nuestro conjunto de datos de instrucciones, se tomó como referencia la librería Bonito de Python (Nayak et al., 2022), un modelo de generación de tareas condicionales del estado del arte. Bonito toma como entrada texto no anotado (por ejemplo, párrafos de artículos científicos, notas clínicas, registros de pacientes) junto con un tipo de tarea deseado (por ejemplo, clasificación de entidades, resumen, implicación textual). Genera pares de instrucción-respuesta relevantes para el tipo de tarea, condicionado en el texto de entrada.

Bonito se entrena en un conjunto de datos a gran escala de generación de tareas condicionales con atributos (CTGA), que contiene 1,65 millones de ejemplos creados remezclando conjuntos de datos de afinación de instrucciones existentes en "plantillas meta". Estas plantillas toman documentos como entrada y generan ejemplos donde la entrada es el documento y el tipo de tarea, y la salida consta de la instrucción y la respuesta. Por ejemplo, una plantilla meta para la clasificación de sentimientos podría verse así:

{texto_entrada}

Pregunta: ¿El texto anterior expresa un sentimiento {sentimiento}? ¿Sí o no?

Respuesta: {etiqueta}

Donde {texto_entrada} es un documento de revisión de productos, {sentimiento} es "positivo" o "negativo", y {etiqueta} es "Sí" o "No" correspondientemente.

Al aplicar estas plantillas meta a un gran corpus de documentos sin anotar, Bonito puede generar conjuntos de datos de instrucciones diversas y relevantes para el dominio. Sin embargo, de acuerdo con su documentación (Bach et al, s.f.), Bonito sólo maneja el idioma inglés lo cual representa la mayor barrera en su aplicación debido a que nuestro conjunto de datos se encuentra por completo en idioma español.

Como alternativa para la generación de los pares de instruction/output se consideró la posibilidad de utilizar el modelo Google Gemini 1.5 el cual tiene la capacidad de manejar varios idiomas, incluido el español, con una capacidad de recibir hasta 1,048,576 como entrada (Google, s.f.). La idea es seguir un procedimiento similar al implementado en equipo de BatsResearch en Bonito, empleando Gemini 1.5 como motor de extracción de información para generar los pares de instrucciones proveniente de nuestro conjunto de datos.

A continuación, se muestra un ejemplo de una solicitud a Google Gemini la cual contiene el contexto y ejemplos de las platillas de la información que se desea extraer empleando la información contenida en el documento legal en curso:

prompt = ""

Eres un experto en inteligencia artificial, trabajas en el desarrollo de Large Language Models.

Estás encargado del desarrollo de un modelo de lenguaje experto en el dominio de las leyes mexicanas, para lo cual estás creando un dataset de tipo instruction tuning.

Hasta el momento, has recabado más de 300 leyes mexicanas, de las cuales requieres extraer información relevante y generar el dataset con preguntas y respuestas basados en los documentos legislativos.

Tomando en cuenta el contenido de la ley que compartiré debajo, por favor genera 10 ejemplos de plantillas de tipo "Question Answering (QA)", "Summarization", "Legal Advice Generation" y "Legal Document Drafting", siguiendo el formato de pares instruction-output.

Ejemplo de Question Answering (QA):

[

{

"instruction": "Según el artículo 22 del Código Civil Federal, ¿desde qué momento entra el individuo concebido bajo la protección de la ley?",

"output": "De acuerdo al artículo 22 del Código Civil Federal, desde el momento en que un individuo es concebido, entra bajo la protección de la ley y se le tiene por

nacido para los efectos declarados en el presente Código."

},

{

"instruction": "¿Qué establece el artículo 23 del Código Civil Federal sobre el ejercicio de derechos de los incapaces?",

"output": "Según el artículo 23 del Código Civil Federal, la minoría de edad, el estado de interdicción y demás incapacidades establecidas por la ley, son

restricciones a la personalidad jurídica que no deben menoscabar la dignidad de la persona ni atentar contra la integridad de la familia; pero los incapaces pueden ejercitar sus derechos o contraer obligaciones por medio de sus representantes."

}

]

Ejemplos de Summarization:

[

{

"instruction": "Resume las obligaciones del Juez del Registro Civil respecto a las actas, según los artículos 43 a 49 del Código Civil Federal.",

"output": "De acuerdo a los artículos 43 a 49 del Código Civil Federal, el Juez del Registro Civil tiene las siguientes obligaciones respecto a las actas: asentar sólo lo

declarado para el acto y previsto en la ley; permitir la representación por mandatario especial; preferir testigos designados por los interesados; evitar falsificación o

inserción de circunstancias prohibidas; corregir vicios o defectos no sustanciales; dar testimonio de actas; y autorizar actas de su propia familia en la adscripción

más próxima."

},

{

"instruction": "Sintetiza lo que establece el Código Civil Federal en los artículos del 86 al 88 sobre la adopción plena.",

"output": "Los artículos 86, 87 y 88 del Código Civil Federal establecen que en la adopción plena se levantará un acta como si fuera de nacimiento, se harán

anotaciones en el acta de nacimiento originaria, la cual quedará reservada. No se publicará ni expedirá constancia alguna que revele el origen del adoptado ni su

condición, salvo providencia judicial. El artículo 88 fue derogado."

}

]

Ejemplos de Legal Advice Generation:

[

{

"instruction": "Si dos personas quieren contraer matrimonio siendo menores de edad, ¿qué requisitos deben cumplir según el Código Civil Federal?",

"output": "De acuerdo al Código Civil Federal, si dos personas menores de edad quieren contraer matrimonio, deben cumplir los siguientes requisitos: 1) Tener al

menos 16 años el hombre y 14 la mujer (art. 148); 2) Obtener el consentimiento de sus padres, abuelos paternos o tutores (arts. 149-155); 3) Solicitar la dispensa del

juez de lo familiar (art. 151); 4) Presentar su solicitud ante el Juez del Registro Civil (art. 97) con la documentación requerida (art. 98). Se recomienda consultar a un

abogado especializado para recibir orientación específica a su caso."

},

{

"instruction": "¿Qué pasos debe seguir una persona para solicitar una acta de nacimiento según el Código Civil Federal?",

"output": "Para solicitar un acta de nacimiento, el Código Civil Federal establece en sus artículos 54 a 76 el siguiente procedimiento: 1) Presentar al recién nacido

ante el Juez del Registro Civil en su oficina o lugar de nacimiento; 2) Los padres o abuelos declaran el nacimiento dentro de los 6 meses siguientes; 3) Si nace en

casa, el jefe de familia tiene obligación de dar aviso; 4) Si nace en un sanatorio, el director o administrador dará aviso; 5) Se levantará el acta de nacimiento con

asistencia de dos testigos, expresándose todos los datos requeridos por la ley. Es importante cumplir los plazos y proporcionar información veraz para evitar

contratiempos posteriores."

}

]

Ejemplos de Legal Document Drafting:

[

{

"instruction": "Redacta una cláusula para un contrato de compraventa de un inmueble, estipulando que las partes se someten a las leyes mexicanas según el

Código Civil Federal.”,

“output”: “CLÁUSULA DÉCIMA.- LEY APLICABLE. Para todo lo relativo a la interpretación, cumplimiento y ejecución del presente contrato, las partes acuerdan

someterse a las leyes federales de los Estados Unidos Mexicanos, en particular a las disposiciones del Código Civil Federal, según lo establecido en sus artículos 12,

13 y demás relativos y aplicables, renunciando expresamente a cualquier otra ley, fuero o jurisdicción que pudiera corresponderles en virtud de sus domicilios presentes o futuros.”

},

{

“instruction”: “Elabora una cláusula para un contrato de arrendamiento especificando que la renta se incrementará anualmente conforme al Código Civil Federal.”,

“output”: “CLÁUSULA DÉCIMA.- INCREMENTO DE LA RENTA. Las partes acuerdan que la renta pactada en la cláusula X del presente contrato se incrementará anualmente

de manera automática, aplicando el porcentaje de aumento que corresponda al Índice Nacional de Precios al Consumidor (INPC) publicado por el Banco de México.

Lo anterior con fundamento en lo dispuesto por el artículo 2448-J del Código Civil Federal. El arrendador notificará por escrito al arrendatario, con al menos 30

días de anticipación, el monto de la renta actualizada que entrará en vigor en la fecha de aniversario del contrato.”

}

]

Por cada registro del JSON que se encuentra a continuación, genera un par instruction-output de cada tipo (“Question Answering (QA)”, “Summarization”, “Legal Advice Generation” y “Legal Document Drafting”):

“““““

La solicitud anterior contiene una plantilla en la cual se le da contexto a Gemini sobre la tarea a realizar, y 4 ejemplos con las categorías de los pares instruction/output a generar: "*Question Answering (QA)*", "*Summarization*", "*Legal Advice Generation*" y "*Legal Document Drafting*". En total, se generarán 40 pares para cada ley con la que cuenta el conjunto de datos extraído de "*Unified, Labeled, and Semi-Structured Database of Pre-Processed Mexican Laws*". Con esto obtendremos un nuevo conjunto de entrenamiento para instrucciones y salidas.

Model Alignment con Instruction Tuning

5.1. Adaptación a Tareas Específicas

Con nuestro conjunto de datos de instrucciones generado, el siguiente paso será afinar un modelo base con este conjunto de datos enfocado en tareas objetivo, por medio del proceso denominado *Instruction Tuning* (IT) (Ouyang et al., 2022).

Durante el *Instruction Tuning*, el modelo aprende a interpretar las instrucciones en lenguaje natural y generar respuestas apropiadas en el estilo y formato deseados. Esto no solo mejora el rendimiento en las tareas objetivo reales, sino que también mejora las habilidades de "*few-shot*" o "*zero-shot*", donde el modelo puede seguir instrucciones nuevas basándose en patrones aprendidos de las instrucciones de entrenamiento (Sanh et al., 2022; Wei et al., 2022).

5.2. Entrenando el Adapter de Tareas Downstream

Después de entrenar nuestro *adapter* [DAPT_ADAPTER] para capturar el conocimiento del dominio, el siguiente paso será entrenar un segundo *adapter*, [DOWNSTREAM_ADAPTER], para las tareas objetivo. Para esto, utilizamos la técnica de *Instruction Tuning*, que implica ajustar el *adapter* en un dataset de instrucciones que consiste en pares de instrucciones en lenguaje natural y respuestas deseadas.

Para entrenar [DOWNSTREAM_ADAPTER], utilizaremos el mismo enfoque PEFT que para [DAPT_ADAPTER] (por ejemplo, LoRA). Congelamos los pesos del modelo base, y solo actualizamos los parámetros de [DOWNSTREAM_ADAPTER]. Esto nos permite capturar el conocimiento específico de la tarea de manera eficiente, mientras conservamos el conocimiento general del dominio aprendido en la etapa de DAPT.

Al igual que en el entrenamiento de [DAPT_ADAPTER], debemos prestar atención en encontrar el set óptimo de hiperparámetros para el entrenamiento [DOWNSTREAM_ADAPTER]. Una vez entrenado, [DOWNSTREAM_ADAPTER] puede combinarse con el modelo base y con [DAPT_ADAPTER] para inferencia en las tareas objetivo. Esta combinación de *adapters* nos permite aprovechar

tanto el conocimiento general del dominio como las habilidades específicas de la tarea para lograr un rendimiento óptimo.

5.3. Combinando los Adapters de DAPT y Tareas Downstream

Propuesta de Modelo y Experimentos

6.1. Arquitectura Propuesta

[Describir a mayor detalle la arquitectura propuesta de modelo base, más dos *adapters*]

Evaluación y Discusión

7.1. Métricas de evaluación

[Realizaremos una investigación de las principales metodologías de evaluación de LLMs, con un enfoque particular en evaluaciones para el dominio legal]

Conclusiones y Trabajo Futuro

[Presentaremos las principales conclusiones de nuestro proyecto, descubrimientos y aprendizajes más importantes, y cuáles serían las recomendaciones para continuar con el desarrollo de una metodología que facilite la adaptación de LLMs a dominios y tareas particulares.]

Bibliografia

- Araci, D. (2019). *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models* (arXiv:1908.10063). arXiv. <https://doi.org/10.48550/arXiv.1908.10063>
- Bach, S., Nayak, N., Yu, P., & Zheng-Xin, Y. (n.d.). *BatsResearch/bonito-v1 · hugging face*. BatsResearch/bonito-v1 · Hugging Face. <https://huggingface.co/BatsResearch/bonito-v1>
- Beltagy, I., Lo, K., & Cohan, A. (2019). SciBERT: A Pretrained Language Model for Scientific Text. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3615–3620). Association for Computational Linguistics. <https://doi.org/10.18653/v1/D19-1371>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). *Language Models are Few-Shot Learners* (arXiv:2005.14165). arXiv. <https://doi.org/10.48550/arXiv.2005.14165>
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S. & Fiedel, N. (2022). *PaLM: Scaling Language Modeling with Pathways* (arXiv:2204.02311). arXiv. <https://doi.org/10.48550/arXiv.2204.02311>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (arXiv:1810.04805). arXiv. <https://doi.org/10.48550/arXiv.1810.04805>
- Gema, A. P., Daines, L., Minervini, P., & Alex, B. (2023). *Parameter-efficient fine-tuning of LLaMA for the clinical domain* (arXiv:2307.03042). arXiv. <https://doi.org/10.48550/arXiv.2307.03042>

- Google. (n.d.). Gemini API docs and reference | google AI for developers | google for developers. Google. <https://ai.google.dev/gemini-api/docs>
- Gururangan, S., Card, D., Dreier, S., Gade, E., Wang, L., Wang, Z., Zettlemoyer, L., & Smith, N. A. (2022). Whose Language Counts as High Quality? Measuring Language Ideologies in Text Data Selection. In Y. Goldberg, Z. Kozareva, & Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 2562–2580). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.emnlp-main.165>
- He, J., Zhao, V. Y., Huang, J., Hu, E., shen, y., Chen, W., & Neubig, G. (2021). *Towards a unified view of parameter-efficient transfer learning* (arXiv:2110.04366). arXiv. <https://doi.org/10.48550/arXiv.2110.04366>
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M. & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (pp. 2790–2799). PMLR. <https://proceedings.mlr.press/v97/houlsby19a.html>
- Hu, E. J., shen, y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). *Lora: Low-rank adaptation of large language models* (arXiv:2106.09685). arXiv. <https://doi.org/10.48550/arXiv.2106.09685>
- Li, X. L., & Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 4582–4597). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.acl-long.353>
- Li, Y., Yao, T., Qin, T., Wei, F., & Liu, T. Y. (2021). *Efficient Transformers: A Survey* (arXiv:2009.06732). arXiv. <https://doi.org/10.48550/arXiv.2009.06732>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach* (arXiv:1907.11692). arXiv. <https://doi.org/10.48550/arXiv.1907.11692>
- Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., & Zhang, Y. (2024). *An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning* (arXiv:2308.08747). arXiv. <http://arxiv.org/abs/2308.08747>
- Mahabadi, R. K., Henderson, J., & Ruder, S. (2021). *Compacter: Efficient low-rank hypercomplex adapter layers* (arXiv:2106.04647). arXiv. <https://doi.org/10.48550/arXiv.2106.04647>

- McCloskey, M., & Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In G. H. Bower (Ed.), *Psychology of Learning and Motivation* (Vol. 24, pp. 109–165). Academic Press. [https://doi.org/10.1016/S0079-7421\(08\)60536-8](https://doi.org/10.1016/S0079-7421(08)60536-8)
- Mishra, S., Khashabi, D., Baral, C., & Hajishirzi, H. (2022). Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In S. Muresan, P. Nakov, & A. Villavicencio (Eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 3470–3487). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.acl-long.244>
- Nayak, N., Nan, Y., Trost, A., & Bach, S. H. (2022). *Learning to Generate Instruction Tuning Datasets for Zero-Shot Task Adaptation* (arXiv:2205.11701). arXiv. <https://doi.org/10.48550/arXiv.2205.11701>
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Aspell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). *Training language models to follow instructions with human feedback* (arXiv:2203.02155). <http://arxiv.org/abs/2203.02155>
- Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., & Gurevych, I. (2020). AdapterFusion: Non-destructive task composition for transfer learning. In P. Merlo, J. Tiedemann, & R. Tsarfaty (Eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (pp. 487–503). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.eacl-main.39>
- Pfeiffer, J., Vulić, I., Gurevych, I., & Ruder, S. (2020). MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7654–7673). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.emnlp-main.617>
- Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Bers, T., Biderman, S., Gao, L., Wolf, T. & Rush, A. M. (2022). *Multitask Prompted Training Enables Zero-Shot Task Generalization* (arXiv:2110.08207). arXiv. <https://doi.org/10.48550/arXiv.2110.08207>

- Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2022). *Self-instruct: Aligning language model with self generated instructions* (arXiv:2212.10560). arXiv. <https://doi.org/10.48550/arXiv.2212.10560>
- Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2022). Finetuned Language Models Are Zero-Shot Learners. In *The Tenth International Conference on Learning Representations*. <https://openreview.net/forum?id=gEZrGCozdqR>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (arXiv:2201.11903). arXiv. <http://arxiv.org/abs/2201.11903>
- Xu, L., Xie, H., Qin, S. Z. J., Tao, X., & Wang, F. L. (2023). *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment* (arXiv:2312.12148). arXiv. <https://doi.org/10.48550/arXiv.2312.12148>
- Zaken, E. B., Ravfogel, S., & Goldberg, Y. (2021). *BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language Models* (arXiv:2106.10199). arXiv. <https://doi.org/10.48550/arXiv.2106.10199>
- Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T. & Zettlemoyer, L. (2022). *OPT: Open Pre-trained Transformer Language Models* (arXiv:2205.01068). arXiv. <https://doi.org/10.48550/arXiv.2205.01068>