UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

AUMENTANDO CAPACIDADES DE LLM DE SEGUIMIENTO DE INSTRUCCIONES EN ESPAÑOL

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

CLEMENTE EMILIO HENRÍQUEZ MUÑOZ

PROFESOR GUÍA:
ANDRÉS ABELIUK KIMELMAN

PROFESOR CO-GUÍA:
FELIPE BRAVO MÁRQUEZ

MIEMBROS DE LA COMISIÓN:
VALENTÍN BARRIERE
JOSÉ MIGUEL PIQUER GARDNER
ELIANA SCHEIHING

Este trabajo ha sido parcialmente financiado por National Center for Artificial Intelligence CENIA FB210017, Basal ANID

SANTIAGO DE CHILE
2024

Resumen

En los últimos años, los grandes modelos de lenguaje (LLM, por sus siglas en inglés) han ganado gran popularidad en el campo del procesamiento de lenguaje natural. Esto se debe en parte a su gran versatilidad y a su capacidad para responder de forma correcta y eficiente a una amplia gama de tareas, incluso en aquellas que no se han visto presentes explícitamente en el entrenamiento de los modelos (zero-shot).

Con la publicación del modelo ChatGPT de OpenAI, los LLM tipo chatbot han ganado aún más popularidad, debido a su capacidad para comprender preguntas y comentarios en lenguaje natural y generar respuestas coherentes en función del contexto proporcionado.

Si bien los modelos de mayor cantidad de parámetros pueden trabajar y generar texto en diversos idiomas, la mayoría de los modelos pequeños solo pueden trabajar de forma eficiente en inglés. Esto en parte por su predominante entrenamiento en conjuntos de datos en inglés, limitando su capacidad de trabajar con otros idiomas.

En esta investigación, proponemos una metodología para mejorar las capacidades para entender y generar texto en español siguiendo instrucciones, al modelo de tipo decodificador causal Falcon-7B, el cual pertenece familia de grandes modelos de lenguaje Falcon, desarrollados por el Technology Innovation Institute de Abu Dhabi.

Esta mejora de capacidades se logra mediante un proceso de pre-entrenamiento secundario y un posterior ajuste de parámetros utilizando datos en español en distinto formato.

Para la evaluación se propone usar MT-Bench adaptado al español. Esta métrica consiste en un conjunto de preguntas de múltiples turnos creado para evaluar grandes modelos de lenguaje, utilizando otros grandes modelos de lenguaje que tengan mejor desempeño y más parámetros que el modelo a evaluar.

Los resultados de esta investigación sugieren que los modelos desarrollados mejoran su capacidad de entender y generar contenido en este idioma, hasta en 2.6 puntos para algunas tareas dentro de la métrica de Mt-bench.

A mis padres, por darme las herramientas y oportunidades para llegar hasta donde estoy hoy; a mis amigos, quienes me acompañaron durante toda esta etapa; y a Pirata, cuya compañía fue siempre una ayuda en los momentos de estrés.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

El campo de Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés) ha experimentado un cambio de paradigma significativo en los últimos años, en gran parte debido a la introducción de los Large Language Models (LLM). Estos modelos, caracterizados por su considerable tamaño y el extenso conjunto de datos de entrenamiento, han demostrado habilidades extraordinarias en la comprensión y generación de texto similar al humano.

A diferencia de los modelos BERT [17], los cuales se caracterizaban por su enfoque bidireccional, teniendo que analizar desde ambos lados del texto para entender su contenido, los LLM se centran en la predicción de tokens, generando texto hacia adelante a partir de lo que ya se ha dado, ideal para la creación de texto coherente y fluido.

Estos modelos ganaron una masiva atención por parte de la población en general a partir de la publicación de los modelos GPT (Generative PreTrained Transformer), de OpenAI[1]. Esto debido en parte a ChatGPT [4], plataforma que permite y facilita el uso de estos modelos por parte de los usuarios.

ChatGPT, funciona como un avanzado modelo de IA conversacional capaz de realizar interacciones contextualmente conscientes y similares a las humanas. Por su parte, GPT-4 [28] representa un LLM más sofisticado que demuestra un mayor potencial en comprensión del lenguaje natural, generación de texto y diversas tareas de NLP, especialmente gracias a sus habilidades multi-modales y de razonamiento.

Estos modelos han catalizado nuevas direcciones de investigación y aplicaciones, intensificando el interés en explorar el potencial de la Inteligencia Artificial General. Además, han demostrado capacidades para el aprendizaje con pocos ejemplos y la adaptabilidad a nuevas tareas, lo que impulsa significativamente la expansión de la investigación en NLP.

En consecuencia, han inspirado tanto a investigadores como a profesionales de la industria a aprovechar al máximo su potencial en una amplia gama de aplicaciones, incluyendo nuevas tareas, dadas a las interacciones que tiene este modelo con humanos, y que van más allá de las tareas clásicas de NLP como el análisis de sentimientos, traducción automática, entre otros.

[1] `https://openai.com`

Sin embargo, a pesar de su impacto positivo, la implementación de LLM presenta limitaciones inherentes que dificultan la investigación transparente y abierta en el campo. Una preocupación importante es su naturaleza propietaria, que limita el acceso a los modelos y, por lo tanto, obstaculiza la capacidad de la comunidad de investigación en general para construir sobre sus éxitos. Además, los recursos computacionales masivos necesarios para entrenar y desplegar estos modelos representan un desafío para los investigadores con recursos limitados, lo que agrava el problema de accesibilidad.

Para abordar estas limitaciones, la comunidad de investigación en NLP ha adoptado alternativas de código abierto para fomentar una mayor transparencia y colaboración. Ejemplos notables de estas iniciativas son la familia de modelos LLaMA [1][33][34] y Alpaca [32], que son modelos LLM de código abierto destinados a facilitar la investigación académica y acelerar el progreso en el campo de NLP.

La intención detrás de la apertura de estos modelos es crear un entorno propicio para el desarrollo, la optimización y la evaluación de LLM robustos y capaces que sean aplicables en una amplia variedad de usos.

Estas iniciativas buscan democratizar el acceso a herramientas de NLP y reducir las barreras para la comunidad de investigación, al tiempo que promueven una mayor colaboración en el campo.

Considerando los desarrollos mencionados anteriormente y la capacidad demostrada por modelos como Guanaco [16] o Chinese Llama [15], para aprender nuevos idiomas y mejorar su comprensión en ellos, surge una pregunta interesante. ¿Es posible replicar los éxitos logrados en investigaciones anteriores, pero enfocándonos en habilitar que modelos de menor tamaño, que no fueron inicialmente entrenados en español, puedan trabajar eficazmente en este idioma?

Se propone replicar el enfoque empleado en la investigación 'Efficient and Effective Text Encoding for Chinese LLaMA and Alpaca' [15] , modificando algunas partes de esta.

Esta iniciativa busca incrementar las capacidades de entendimiento y generación de texto en español del modelo Falcon-7B [2], (modelo que fue entrenado originalmente en inglés y francés) mediante dos técnicas: Pre-entrenamiento secundario, y el ajuste de parámetros o fine tuning, con el fin de explorar cuanto aprende el modelo en base a estas técnicas.

Bautizaremos el modelo generado como FAISÁN (Falcon in SpANish), para el modelo generado luego del secondary pre-training y Faisán-Instruct para el modelo post fine-tuning

1.1. Problema de Investigación

A la fecha que se comienza a desarrollar esta investigación se tiene el problema de que existen muy pocos modelos pequeños (bajo los 10 mil millones de parámetros) que pueden generar respuestas competitivas en español.

Con el fin de democratizar el acceso a este tipo de herramientas y de impulsar su eficiencia

en este idioma, se propone el entrenamiento de un modelo que potencie sus capacidades para trabajar en español que pueda generar respuestas a nivel similar que en su idioma original, en un idioma que no fue entrenado.

Junto con esto se busca simplificar el proceso de aprendizaje de un idioma, para lenguas que no tienen tanta diferencia (en cuanto a letras y sintaxis), como lo sería el español, con el inglés. Por lo tanto, la pregunta de investigación central es: ¿Es posible entrenar un modelo de lenguaje grande en español utilizando recursos computacionales y de datos limitados?"

Es importante destacar que, a la fecha de publicación de esta investigación ya existen modelos que cumplen estas características, como la familia de modelos de Llama-2 [34] y Llama-3 [1], entre otros.

1.2. Hipótesis

Un LLM que no fue entrenado originalmente en español, puede incrementar sus capacidades de trabajar en este idioma y alcanzar niveles competitivos de respuestas, si se aplican técnicas de pre-entrenamiento secundario y finetuning con suficientes datos en español.

1.3. Objetivos

El objetivo principal de esta investigación es entrenar y evaluar un LLM de código abierto que tenga menos de 10 mil millones de parámetros para que adquiera capacidades y potencie su entendimiento del español. Esto bajo la limitación de hardware disponible, descrita en la sección 4.

Esta investigación busca determinar la viabilidad de hacer que un modelo pequeño como Falcon-7B, que no fue entrenado en español, adquiera la capacidad de responder y trabajar en este idioma.

Esto nos deja con los siguientes objetivos específicos.

1. Determinar la viabilidad de hacer que un modelo pequeño como Falcon-7B, que no fue entrenado en español, adquiera la capacidad de responder y trabajar en este idioma.
2. Comparar dos enfoques de aprendizaje: hacer fine-tuning directamente y realizar una etapa de pre-training secundario seguida de fine-tuning, y compararlos con un modelo entrenado con datos en español y de tamaño similar.

Si obtenemos resultados positivos, esta investigación podría servir para enseñar español a modelos que no fueron entrenados con datos en este idioma.

1.4. Preguntas de Investigación

1. ¿Cuál es la influencia de la etapa de pre-entrenamiento secundario en el modelo y cuánto influye en su capacidad para aprender un nuevo idioma?
2. ¿Es factible que un modelo aprenda un idioma únicamente mediante fine-tuning sin el pre-entrenamiento secundario, reduciendo significativamente el tiempo de desarrollo?
3. ¿Los modelos que no fueron entrenados originalmente en español pueden aprender este idioma y alcanzar niveles competitivos de trabajo mediante un entrenamiento con datos en español en un hardware limitado?
4. ¿En qué medida se ven alteradas las capacidades de producción de respuestas en inglés de un modelo al entrenarlo solo en español?
5. ¿Cómo se comparan las respuestas generadas por el modelo original con las del modelo entrenado en inglés y español?

1.5. Metodología

Con el fin de alcanzar los objetivos específicos descritos anteriormente, esta sección presenta la metodología propuesta para la investigación. Precisamente, nuestro trabajo consiste principalmente en los siguientes pasos:

1. Escoger un modelo base, sobre el cual se realizarán los experimentos. Este debe cumplir con los requerimientos específicos, como no haber sido entrenado en español, tener menos de diez millones de parámetros y ser open-source.
2. Investigar y recolectar conjuntos de datos para el entrenamiento del modelo. Traducir estos en caso de que sea necesario
3. Traducir el set de preguntas propuesto por Mt-Bench al español, para evaluar los modelos producidos en la investigación.
4. Evaluar el rendimiento del modelo base propuesto en su versión básica y su versión instruct para determinar su rendimiento base en su idioma original en el MT-Bench. Esta sección nos dará información sobre que cuál es la meta a la que nos queremos acercar, al final de la investigación.
5. Evaluar el rendimiento del modelo base propuesto en su versión básica y su versión instruct para determinar su rendimiento en español, en el MT-Bench.
6. Entrenar el modelo base utilizando únicamente el conjunto de datos para fine-tuning, para experimentación preliminar
7. Obtener un modelo base luego del pre-entrenamiento Secundario con una parte del Spanish Unnotated Corpora
8. Obtener un modelo instruct, mediante el entrenamiento y ajuste de parámetros del modelo producido en el paso anterior.

9. Evaluar el rendimiento de los modelos obtenidos en el Mt-bench en español.

En el Capítulo 3, se explica en profundidad la metodología seguida en esta investigación.

1.6. Estructura de la Tesis

El resto de la tesis está organizado de la siguiente manera:

- En el Capítulo 2, proporcionamos el marco teórico necesario para comprender nuestra investigación junto con una revisión del trabajo relacionado y modelos publicados que son relevantes para esta investigación.
- El Capítulo 3, se profundiza en la metodología seguida en la investigación
- El Capítulo 4, se detallan el entrenamiento de los modelos.
- En el Capítulo 5, se muestran los resultados de los modelos en Mt-bench y se discuten estos resultados
- El Capítulo 6, se compone de las conclusiones del trabajo de esta investigación, junto con limitaciones presentes, contribuciones y trabajos futuros.

Capítulo 2

Contexto y Trabajo Relacionado

El capítulo se divide en tres secciones principales. La primera sección proporciona una introducción a el área de conocimiento relevante para la tesis, en específico a los Large Lenguage Models (LLM), que son modelos que se basan en la arquitectura de redes neuronales tipo Transformers. La segunda sección es una revisión a los conjuntos de datos, que se utilizarán para entrenar los modelos. La tercera sección se centra en técnicas de entrenamiento de LLM y métricas que se utilizarán para evaluar el desempeño de los modelos generados. Este capítulo sienta las bases para el resto de la tesis, guiando la investigación y el análisis de las preguntas de investigación.

2.1. Transformers, Large Language Models and Chatbots

2.1.1. Transformers

Las redes neuronales tipo Transformer [35] son una arquitectura de redes neuronales profundas que se destacan en el procesamiento de lenguaje natural y otras aplicaciones de aprendizaje automático. Su característica clave es la auto-atención, que permite a la red aprender relaciones entre elementos en una secuencia de datos, como palabras en una oración, de manera altamente eficiente y paralela. A diferencia de las redes recurrentes, estas son paralelizables y no se trabaja de forma secuencial, lo que las hace más fáciles de entrenar y más eficientes.

Como se muestra en la imagen 2.1 las redes Transformer se dividen en dos partes principales: el codificador y el decodificador, cada uno compuesto por múltiples capas apiladas. Cada capa tiene múltiples cabezas de atención y capas de feedforward. Esto permite que la red aprenda representaciones cada vez más complejas de los datos de entrada.

Además, se utilizan máscaras de atención en el decodificador para asegurar que las predicciones no dependan de datos futuros. Las redes Transformer también utilizan normalización

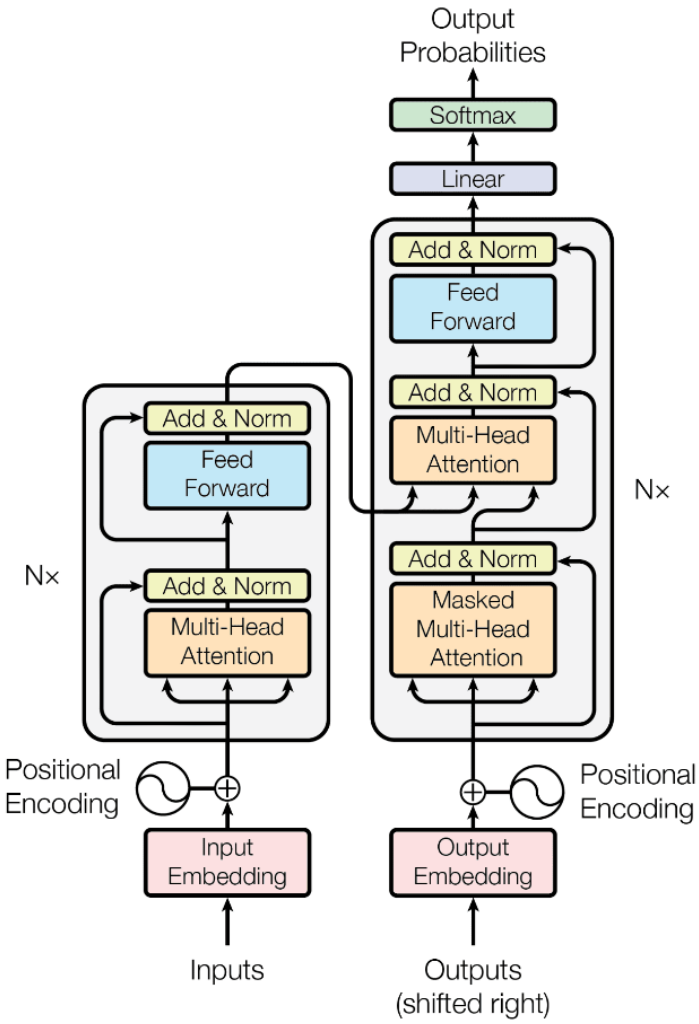

Figura 2.1: Arquitectura de Transformers por Vaswani et al. [35]

por lotes y la función de activación ReLU para mejorar la estabilidad y la capacidad de aprendizaje.

Por último, estas redes se pre-entrenan en grandes conjuntos de datos no supervisados y luego se ajustan finamente para tareas específicas con datos etiquetados. En resumen, las redes neuronales tipo Transformer han revolucionado el procesamiento de lenguaje natural y son altamente versátiles en diversas aplicaciones de aprendizaje automático.

2.1.2. Large Language Models

Large language models en el procesamiento del lenguaje natural (NLP) se refiere a modelos de aprendizaje automático que son capaces de entender y generar texto en lenguaje natural a una escala considerable. Estos modelos utilizan técnicas de aprendizaje profundo, específicamente arquitecturas como Transformers, para aprender patrones complejos en conjuntos de datos masivos.

A la fecha, los LLM se han ganado una gran atención por parte de la comunidad científica y no científica, en parte por el modelo ChatGPT desarrollado por OpenAI.

GPT utiliza una arquitectura de Transformers y se entrena en grandes cantidades de datos textuales para aprender patrones y representaciones semánticas del lenguaje natural. El modelo resultante puede realizar tareas como traducción automática, generación de texto, respuesta a preguntas, entre otras, sin requerir de fine-tuning para una tarea en específico. La idea detrás de estos modelos es que, al exponerlos a grandes cantidades de datos textuales, pueden aprender patrones complejos y representaciones semánticas del lenguaje natural, lo que les permite realizar tareas NLP con un rendimiento sorprendentemente bueno.

Es importante destacar que estos modelos grandes pueden tener cientos de millones o incluso miles de millones de parámetros, lo que les permite capturar y generar patrones muy complejos en el lenguaje. Sin embargo, también requieren una gran cantidad de recursos computacionales para el entrenamiento y la inferencia.

A continuación, se detallan LLM desarrollados en distintas investigaciones y que son relevantes para el desarrollo de esta.

2.1.3. Falcon

Falcon es un conjunto de modelos creados por el "Technology Innovation Institute"[1], a partir de RefinedWeb[2]. Durante los meses de mayo y junio de 2023, lideró la tabla de HugginFace Open LLM Leaderboard [3], y tiene dos modelos base sobre los cuales se realiza fine-tuning para generar los modelos Falcon-Instruct.

El OpenLLM Leaderboard de Hugging Face es una plataforma competitiva donde se exhiben y comparan modelos de lenguaje de última generación desarrollados por diversas

[1] `https://www.tii.ae/`

[2] `https://huggingface.co/datasets/tiiuae/falcon-refinedweb`

organizaciones y equipos de investigación. En este tablero de clasificación, los modelos son evaluados y clasificados según su rendimiento en una variedad de tareas y métricas, como comprensión de texto, generación de lenguaje, y habilidades de traducción. El objetivo es proporcionar un panorama claro de las capacidades actuales de los modelos de lenguaje, facilitando la identificación de los más avanzados y promoviendo la innovación y el avance en el campo de la inteligencia artificial. Además, este leaderboard permite a los desarrolladores y a la comunidad científica realizar comparaciones objetivas y basadas en datos entre los diferentes modelos, fomentando así una sana competencia y colaboración en el desarrollo de tecnologías de procesamiento de lenguaje natural.

El modelo Falcon más grande tiene 180 mil millones de parámetros y a mediados del año 2023, su versión de 40 mil millones de parámetros ocupó el primer lugar en la tabla de clasificación, mientras que el modelo más pequeño tiene 7 mil millones de parámetros y ocupó el puesto 18.

Los modelos más grandes son multilingües y abarcan 4 idiomas (inglés, francés, español y alemán), mientras que el más pequeño solo abarca 2 (francés e inglés).

Falcon es de código abierto, se rige bajo la licencia Apache-2.0, está disponible para su descarga y uso inmediato en Huggingface y será el modelo base sobre el cual se realizarán los experimentos.

La decisión de usar este modelo, aparte de que cumple con no haber sido entrenado en español, es su buen desempeño en el test de HuggingFace, además de su facilidad de acceso y libre licencia.

2.1.4. LLaMA y LLaMA-2

LLaMA es una colección de modelos de lenguaje que abarcan desde 7 mil millones hasta 65 mil millones de parámetros. Estos modelos se entrenaron utilizando trillones de tokens y demostraron que es posible lograr modelos de vanguardia utilizando conjuntos de datos públicos, sin recurrir a conjuntos de datos privados e inaccesibles.

Un token es una unidad indivisible que representa una palabra, un fragmento de palabra o un carácter en un texto. Se utiliza para dividir y analizar el texto en componentes más pequeños, lo que facilita su procesamiento y análisis.

LLaMA-13B supera a GPT-3 en la mayoría de las pruebas de referencia, a pesar de ser más de 10 veces más pequeño. Además, LLaMA-65B es competitivo con los mejores modelos, como Chinchilla-70B y PaLM-540B.

Estos modelos de código abierto han fomentado muchas investigaciones, que han dado lugar a diversas versiones de este modelo mediante distintas técnicas y configuraciones de entrenamientos.

Ejemplos de estos son Alpaca [32] y Vicuna [8], modelos generados en base a LLaMA y ajustados en datos de diferentes tipos. El primero se entrenó con un conjunto de datos de auto-instrucciones de 52,000 ejemplos, generado con el modelo de la empresa OpenAI text-

davinci-003,(OpenAI, 2021.). El segundo utilizó los datos de ShareGPT [3], plataforma que comparten resultados de interacciones humanas con ChatGPT.

En el desarrollo de estos modelos se han liberado tanto los códigos con los que se desarrollaron como parte de sus conjuntos de datos (o sus conjuntos de datos completos), junto con sus métricas de evaluación, para que futuras investigaciones pueda utilizarlos y contribuir a mejorar su robustez y abordar problemas conocidos, como la toxicidad y el sesgo.

Los resultados de estas investigaciones sugieren que aprender de conjuntos de datos de alta calidad puede mitigar algunas de las limitaciones de los modelos más pequeños, incluso llegando a igualar las capacidades de los grandes modelos cerrados en el futuro. Esto podría implicar, por ejemplo, que la comunidad debería esforzarse más en la curación de conjuntos de datos de alta calidad, ya que esto podría lograr más en términos de permitir modelos más seguros, basados en hechos y capaces que simplemente aumentar el tamaño de los sistemas existentes. [38]

2.1.5. Alpaca

Como se mencionó anteriormente, Alpaca [32] es un modelo generado a partir del ajuste de pesos, o fine-tuning de modelos LLama.

El conjunto de datos utilizado para el entrenamiento del modelo consistió en 175 pares de auto-instrucciones generados en primera instancia por personas, y luego expandido, mediante el modelo de Open-AI, text-davinci-003. En esta expansión de datos, se utilizó el modelo mencionado anteriormente para que, en base a los 175 pares hechos por humanos, se generarán ejemplos a gran escala.

El conjunto de datos contiene el siguiente formato, presentado en la imagen 2.2:

```
Below is an instruction that describes a task. Write a response that appropriately completes the request

### Instruction:
{instruction}

### Response:
```

Figura 2.2: Formato conjunto de datos utilizados para generar modelo Alpaca [32]

Acorde a los autores de la investigación, mediante el ajuste de parámetros del modelo original, donde el objetivo del entrenamiento es que el modelo aprenda a comportarse como chatbot para poder facilitar su uso, logran que Alpaca tenga adquiera las capacidades de seguir instrucciones y que se comporte de forma similar al modelo text-davinci-003.

Esta investigación resulta relevante, pues muestra un posible método de hacer que un modelo base, como LLama, incremente sus capacidades y facilite interacción con usuarios

[3] `https://sharegpt.com`

mediante la capacidad de seguir instrucciones y responder a preguntas en un formato determinado.

2.1.6. Chinese-LLama and Alpaca

Esta investigación se propone un método para potenciar LLaMA con capacidades para comprender y generar texto en chino, así como mejorar su capacidad para seguir instrucciones. [15]

Con este objetivo, la investigación tuvo un desarrollo de tres partes. En la primera etapa, se centra en la ampliación del vocabulario chino mediante la adición de tokens (representaciones de palabras). Esto se logra primero entrenando el tokenizador chino con SentencePiece [23] en un vocabulario de tamaño 20,000.

Luego, concatenan el tokenizador chino con el original (aumentando su vocabulario de 32,000 a 49,953). Posteriormente, adaptan el modelo original ajustando las dimensiones de su word-embedding y su *language model head*. Las nuevas filas se añaden al final de las matrices respectivas para no alterar el vocabulario original.

En la segunda parte, se realiza una fase de pre-entrenamiento secundario utilizando las bibliotecas Lora y Peft. El modelo se entrena bajo la tarea estándar de 'Causal Language Modeling' en un conjunto de datos de 20GB de corpus chino, que es coherente con el utilizado para el modelo Chinese Bert-wwm [13]. En otras palabras, el objetivo de esta fase es entrenar al modelo para predecir el siguiente token x_i de manera autoregresiva a partir de la secuencia $x = (x_0, x_1, x_2, \ldots)$. La siguiente fórmula describe esto, donde se busca minimizar la log-verosimilitud negativa:

$$L_{CLM}(\Theta) = \mathbb{E}_{x \sim D_{PT}} \left[-\sum_i \log p(x_i | x_0, x_1, \ldots, x_{i-1}; \Theta) \right]$$

Donde Θ representa los parámetros del modelo, D_{PT} es el conjunto de datos de pre-entrenamiento, x_i es el token que se debe predecir, y $x_0, x_1, \ldots, x_{i-1}$ conforman el contexto.

En esta etapa de la investigación, logran obtener el primero modelo, llamado Chinese-LLama. Este es un modelo base, que no tiene comportamiento de chatbot, ni tiene la capacidad de seguir instrucciones.

La última fase consiste en el ajuste fino del modelo obtenido en el paso anterior utilizando un conjunto de datos de entre 2 y 3 millones de instrucciones (siguiendo el formato de auto-instrucciones utilizado en Alpaca). A diferencia de la fase anterior, donde el objetivo era que el modelo aprendiera a predecir el siguiente token, en esta etapa se busca que el modelo sea capaz de seguir instrucciones específicas y responder en el formato entrenado, el cual es el formato que se presenta en el conjunto de datos de entrenamiento de Standford Alpaca 2.2.

Los resultados de esta investigación mejoran significativamente la capacidad del modelo para comprender y ejecutar instrucciones.

Resultados experimentales indican que los modelos desarrollados mejoran notablemente la competencia original de LLaMA en comprender y generar contenido en chino. Además, los resultados en el conjunto de datos C-Eval [22], conjunto integral de evaluación en chino, que consta de 13,948 preguntas de opción múltiple de distintos niveles de dificultad, y que abarca 52 disciplinas diversas, muestran un rendimiento competitivo en comparación con modelos varias veces más grandes.

Esta investigación será la guía que se utilizará para desarrollar un modelo en español. Se modificarán ciertas partes para alinearnos con los objetivos específicos de esta investigación.

2.2. Conjuntos de Datos Utilizados

Nos concentraremos en entrenar los modelos bases utilizando conjuntos de datos de código abierto u $open-source$ de distintas fuentes (entre ellos los provistos por las investigaciones relacionadas), a su vez incrementándolo y/o traduciéndolos al español.

Dentro de los datos utilizados, encontramos dos tipos, los destinados a que el modelo aprenda e incremente sus capacidades para entender el español y los destinados a que el modelo aprenda a interactuar con usuarios y seguir instrucciones.

Los del primer tipo se utilizan en la etapa de pre-entrenamiento secundario. En esta etapa se utiliza una gran cantidad de texto plano en español, buscando que el modelo aprenda únicamente a predecir el siguiente token.

Los de la segunda parte tienen un formato específico de preguntas y respuestas, junto a un *prompt* inicial que define lo que debe hacer el modelo. Estos conjuntos de datos provienen de distintas fuentes. Todos estos conjuntos de datos se han utilizado para el entrenamiento de otros modelos en inglés, haciendo que estos adquieran las capacidades de seguimiento de instrucciones, Estos datos están en inglés, por lo que se traducirán automáticamente al español, con lo que se espera que su calidad se vea afectada en un porcentaje menor.

A continuación se detallan los conjuntos de datos a utilizar.

2.2.1. Conjuntos de Datos para el Pre-entrenamiento Secundario

Como se mencionó anteriormente, los conjuntos de datos que se utilizan en esta etapa buscan hacer que el modelo aprenda español, mediante la predicción de tokens usando texto plano.

Estos conjuntos de datos han sido utilizados en otras investigaciones para entrenar modelos bidireccionales, en español.

Spanish Unannotated Corpora

Este repositorio [7] recopila un conjunto de corpus en español. El conjunto de datos consta de un total de 300,904,000 líneas de texto, que equivalen a 3,000,000,000 de tokens y 18,431,160,978 caracteres (18.4 mil millones de caracteres). Los datos provienen de diversas fuentes, incluyendo wikis en español como Wikipedia, Wikinews y Wikiquotes, que fueron procesados inicialmente con WikiExtractor. También se incluyen datos de ParaCrawl, EU-Bookshop, MultiUN, OpenSubtitles, DGC, DOGC, ECB, EMEA, Europarl, GlobalVoices, JRC, News-Commentary11, TED y UN, lo que hace que este conjunto de datos sea amplio y diverso en términos de contenido en español.

Este conjunto de datos fue utilizado en el desarrollo de modelos como BETO [7], el cual es un modelo tipo BERT desarrollado el 2020.

2.2.2. Conjuntos de Datos para Fine Tuning

Estos conjuntos de datos que se utilizan en esta etapa buscan hacer que el modelo aprenda a generar respuestas y a seguir instrucciones. Para esto como se dijo previamente se utilizaron tres conjuntos de datos en inglés que buscaban hacer que modelos adquiriesen esta capacidad. Estos tres conjuntos de datos fueron traducidos al español utilizando el modelo disponible bajo la licencia de Apache-3.0 de HuggingFace Helsinki-NLP/opus-mt-en-es [4], por su facilidad de uso, y buenos resultados de traducción. Se tiene en consideración que al ser traducidos automáticamente, se pierda parte de la calidad del conjunto de datos original.

Standford Alpaca Dataset

Se usará el conjunto de datos desarrollado para generar al modelo Alpaca. Este se obtuvo mediante procesos de seguimiento y replicación de auto-instrucciones. Para esto se producen triples de ejemplos instrucción, input y respuesta, hechos por humanos y luego se le pide al modelo de OpenAI text-davinci-003, que genere más triples inspirados en estos ejemplos. Contiene 52k de ejemplos en inglés, por lo que será traducido al español. El conjunto de datos sigue el presentado en la tabla 2.1

Campo	**Valor**
Prompt	Below is an instruction that describes a task. Write a response that appropriately completes the request.
Instruction	What are the three primary colors?
Input	
Response	The three primary colors are red, blue and yellow.

Tabla 2.1: Ejemplo de una auto-instrucción utilizada en el conjunto de datos de Standford Alpaca

[4] `https://huggingface.co/Helsinki-NLP/opus-mt-en-es`

Como se muestra en la tabla 2.1, existe también el campo *input*, el cual algunas veces contiene información que se añade a la instrucción.

Dolly-15K

El conjunto de datos databricks-dolly-15k [12], creado por empleados de Databricks, ofrece 15000 registros para entrenar modelos de lenguaje. Puede utilizarse con cualquier propósito bajo la licencia Creative Commons. Incluye categorías como lluvia de ideas, clasificación y preguntas y respuestas.

Contiene instrucciones generadas por empleados y referencias de Wikipedia en ciertas categorías. Al igual que otros datasets, posee un posible sesgo y errores en su contenido.

Posee un formato de Instruction, Context y Response, el cual es análogo al descrito previamente, para el conjunto de datos de Alpaca.

Al igual que el resto de los conjunto de datos, este será traducido al español.

Open Orca

El conjunto de datos Open Orca es una colección de datos de la Colección FLAN aumentada. Actualmente cuenta con aproximadamente 1 millón de ejemplos completados de GPT-4 y 3.2 millones de completados de GPT-3.5. Está tabularizado en consonancia con las distribuciones presentadas en el artículo ORCA [25] y actualmente representa una finalización parcial del conjunto de datos completo previsto, con generación continua para ampliar su alcance.

Al ser estos ejemplos mucho más detallados, algunos de ellos tienen preguntas de hasta 40000 palabras.

Tomando en cuenta nuestro hardware, los tiempos de procesamiento y que nuestro modelo no busca trabajar con tantos tokens, únicamente seleccionamos las filas que cumplieran que tanto preguntas como respuestas fueran de menos de 500 palabras.

En un futuro se verá la opción de añadir las filas que tengan hasta 1000 o 2000 palabras. Esto añadirá un coste adicional, puesto que no solo se toma en cuenta el tiempo de fine-tuning, sino el de traducción y corrección de estos datos. Quedamos un un archivo que contiene alrededor de 278K ejemplos. Este conjunto de datos contiene diversas instrucciones. Su formato es de un *prompt* que especifica como debe actuar el modelo, una pregunta y una respuesta, lo que lo hace mantener el formato de los dos conjuntos de datos anteriores.

Al igual que los dos anteriores, este fue traducido automáticamente.

2.3. Herramientas y Marco de Referencia en la Investigación

En esta sección, exploraremos las herramientas fundamentales utilizadas en la investigación, destacando la contribución de tecnologías clave como Low Ranking Adaptation (LORA), MT-BENCH, y otras.

2.3.1. LoRA

El entrenamiento tradicional de los modelos donde se actualizan todos los parámetros tiene un gran costo tanto de tiempo como de recursos computacionales. Low-Rank Adapation (LoRA) [21] es una técnica de entrenamiento de re-parametrización eficaz que conserva los pesos de un modelo previamente entrenado mientras introduce matrices de descomposición de rangos ajustables. LoRA mantiene los pesos del modelo previamente entrenado sin cambiarlos y añade matrices de bajo rango ajustables en cada capa. Este enfoque resulta en una notable reducción de la cantidad total de parámetros ajustables, lo que facilita el entrenamiento de modelos de lenguaje con un uso mucho menor de recursos computacionales.

LoRA sigue la siguiente ecuación para el forward pass, con x como input

$$h = W_0 x + \Delta W x = W_0 x + BAx, \quad B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times d} \tag{2.1}$$

Para una capa lineal con matriz de pesos $W_0 \in \mathbb{R}^{d \times k}$, donde k es la dimensión de entrada, y d es la dimensión de salida, LoRA añade dos matrices entrenables de descomposición de bajo rango $B \in \mathbb{R}^{d \times r}$ y $A \in \mathbb{R}^{r \times k}$, donde r es el rango de la descomposición, actuando como un cuello de botella que controla la complejidad y la capacidad de adaptación del modelo modificado.

Durante el entrenamiento, la matriz de pesos W_0 queda congelada y no recibe actualizaciones de gradientes, mientras que B y A sí son actualizadas. Al elegir un rango $r \ll \text{mín}(d, k)$, se reduce el consumo de memoria, ya que no necesitamos almacenar los estados del optimizador para la gran matriz congelada.

Ahora, el hecho de que LoRA permita que el entrenamiento de los modelos sea más rápido y requiera de menos recursos computacionales, no significa que se tenga el mismo rendimiento que con el entrenamiento clásico. En otras palabras, $W_0 x + \Delta W x$, para un r pequeño, no se alejaría mucho del $W_0 x$ original.

El que solamente se modifiquen algunas capas de las matrices de pesos del modelo original, hace que los resultados de este entrenamiento se vean afectados.

En la investigación de LoRA sugieren que incluso con valores de r muy bajos, el desempeño del modelo mejora, pero que por regla general muestran que a mayor r, se obtienen mejores resultados, por lo que se buscará maximizar esta variable dentro de las capacidades de hardware que se poseen.

Al igual que para el modelo Chinese-LLaMA, utilizaremos lora tanto para la fase de pre-entrenamiento secundario, como para la de fine-tuning.

2.3.2. MT-BENCH

La irrupción de large language models ha transformado radicalmente el paradigma del procesamiento del lenguaje natural (NLP), desafiando las limitaciones tradicionales asociadas con tareas específicas, como clasificación de texto, etiquetado de secuencias y generación de secuencias.

Estos modelos como la familia de modelos GPT, han superado las fronteras de las tareas clásicas del NLP al aprender patrones complejos y representaciones semánticas a partir de enormes cantidades de datos textuales. Un de nuevas capacidades de estos modelos son el generar texto en patrones difíciles de trabajar, como lo sería hacer una tabla de planificación semanal de comidas.

La capacidad de estos modelos de abordar problemas más amplios y complejos, como la comprensión contextual profunda, la generación de texto coherente y la resolución de tareas multifacéticas, ha llevado a que se desarrollen nuevas técnicas para evaluar estos modelos, ya que los enfoques clásicos de evaluación en el procesamiento de lenguaje natural, que se centraban en tareas específicas como clasificación de sentimientos, filtrado de spam y reconocimiento de entidades, no son suficientes para evaluar la capacidad que tienen estos modelos para entender y generar texto en lenguaje natural.

En otras palabras, pedir a estos modelos clasificar un documento o reconocer entidades en un texto, son tareas que no representan su amplia capacidad de resolver y generar textos complejos con un gran entendimiento del lenguaje natural. Lo que se buscaría evaluar es como son las interacciones que tienen los modelos con los usuarios.

¿Cómo se evaluarían las interacciones que tendría el modelo con un usuario? Si volvemos al ejemplo de generar una tabla de planificación semanal de comidas, requeriríamos de una evaluación humana para poder determinar si la tabla generada es una respuesta buena o no.

Con esto en vista se desarrolla MT-bench [37] (multi-turn bench), benchmark propuesto y desarrollado por LMSYS[5] que se enfoca en utilizar modelos grandes como GPT-4 para determinar la calidad de las respuestas de un modelo más pequeño.

Esta evaluación buscaba determinar hasta qué punto un modelo grande como GPT-4 coincide con la evaluación humana, a la hora de determinar la calidad de una respuesta de un modelo más pequeño.

En esta investigación se mostró que entre las evaluaciones de GPT-4 y humana, existe más de un 80 por ciento de acuerdo.

Esto otorga una serie de beneficios, puesto que se adquiere la capacidad de evaluar respuestas dentro de un amplio espectro, sin necesidad de depender de humanos y siendo mucho

[5] `https://lmsys.org`

más rápido y eficiente.

El proceso de MT-bench, consta de dos partes. En la primera, el modelo a evaluar se somete a un conjunto de preguntas diseñadas para requerir respuestas múltiples y seguir un flujo de conversación lógico. Estas preguntas pueden abarcar una variedad de temas y pueden ser interdependientes, lo que significa que la respuesta a una pregunta puede influir en las preguntas siguientes de la conversación.

La segunda parte implica una evaluación realizada por GPT-4, que actúa como juez frente a estas respuestas. GPT-4 evalúa las respuestas en función de su coherencia, relevancia y calidad general de la interacción. Se otorga una calificación del 1 al 10 y proporcionan una explicación de su evaluación.

El conjunto de preguntas propuesto para MT-bench cuenta con 80 conversaciones que se componen de dos preguntas, la pregunta del Turn I y la pregunta Turn II. Como se mencionó previamente, la pregunta del Turn II usualmente, tiene relación a la pregunta y respuesta del Turn I. Este conjunto de preguntas está separado en 8 categorías: *writing*, *roleplay*, *reasoning*, *math*, *coding*, *extraction*, *stem*, *humanities*. Estas preguntas están originalmente en inglés, por lo que fueron traducidas al español junto a los *prompts* que se utilizan en GPT-4 para la evaluación.

Es importante señalar que MT-Bench es una métrica enfocada en evaluar modelos generativos tipo chatbot. Los modelos que no son de este tipo (modelos base), obtienen resultados muy bajos, por lo cual, no es una buena métrica para evaluar modelos de este estilo.

A continuación en la tabla 2.2, se muestra un ejemplo de una de las preguntas que se utilizan en MT-bench

Campo	**Valor**
id	81
category	writing
Turn I	Componga un interesante blog de viajes sobre un viaje reciente a Hawai, destacando experiencias culturales y atracciones que deben ver.
Turn II	Reescribir su respuesta anterior. Comience cada frase con la letra A.

Tabla 2.2: Ejemplo preguntas Mt-bench

Para cada respuesta se genera un puntaje del 1 al 10 que representa lo buena que fue esta, para luego al final se otorga un puntaje promediando las respuestas por turno.

Capítulo 3

Metodología

En este capítulo detallaremos como fue la metodología para el desarrollo de la investigación y la generación de los modelos Faisan-7b y Faisan-7b-Instruct

3.1. Selección del Modelo Base

Para la selección del modelo base se buscó encontrar el mejor modelo de código abierto u $open - source$ que tuviera menos de 10 mil millones de parámetros y que no hubiese sido entrenado con datos en español.

Para determinar cuál era el mejor modelo que cumpliese con estos requisitos se utilizó $OpenLLMLeaderboard$ [3], plataforma dentro de $HuggingFace$, dedicada a rastrear, clasificar y evaluar LLM a medida que son lanzados o desarrollados.

En esta plataforma, los miembros de la comunidad pueden someter sus modelo para su evaluación automática en el clúster de GPUs proporcionado por HuggingFace, siempre y cuando sea un modelo de Transformers con pesos en la plataforma.

A la fecha en que se escoge el modelo base con el que se trabajó, falcon-40b era el modelo que lideraba la tabla y falcon-7b y su versión instruct estaban cercanos a la posición 15, lo que lo hacia el candidato óptimo con el que trabajar.

Otros modelos que se tuvieron en consideración a utilizar llama-7b [33] y FastChat-t5 [37], sin embargo, el rendimiento del segundo era más bajo que el de falcon-7b y el primero, contaba con un formulario de acceso que debía ser aprobado por Meta, para obtener los pesos del modelo, junto con la condición de que al publicar un modelo obtenido mediante el entrenamiento de este, se subieran únicamente los deltas de los pesos que se debían aplicar al modelo original. Esto hizo que se descartara esta opción, pues se veían afectada la condición de ser open-source.

Un último aspecto a mencionar fue la selección entre el modelo base y su versión instruct. Tanto por la recomendación hace el equipo del TIIUAE, que se muestra en el repositorio

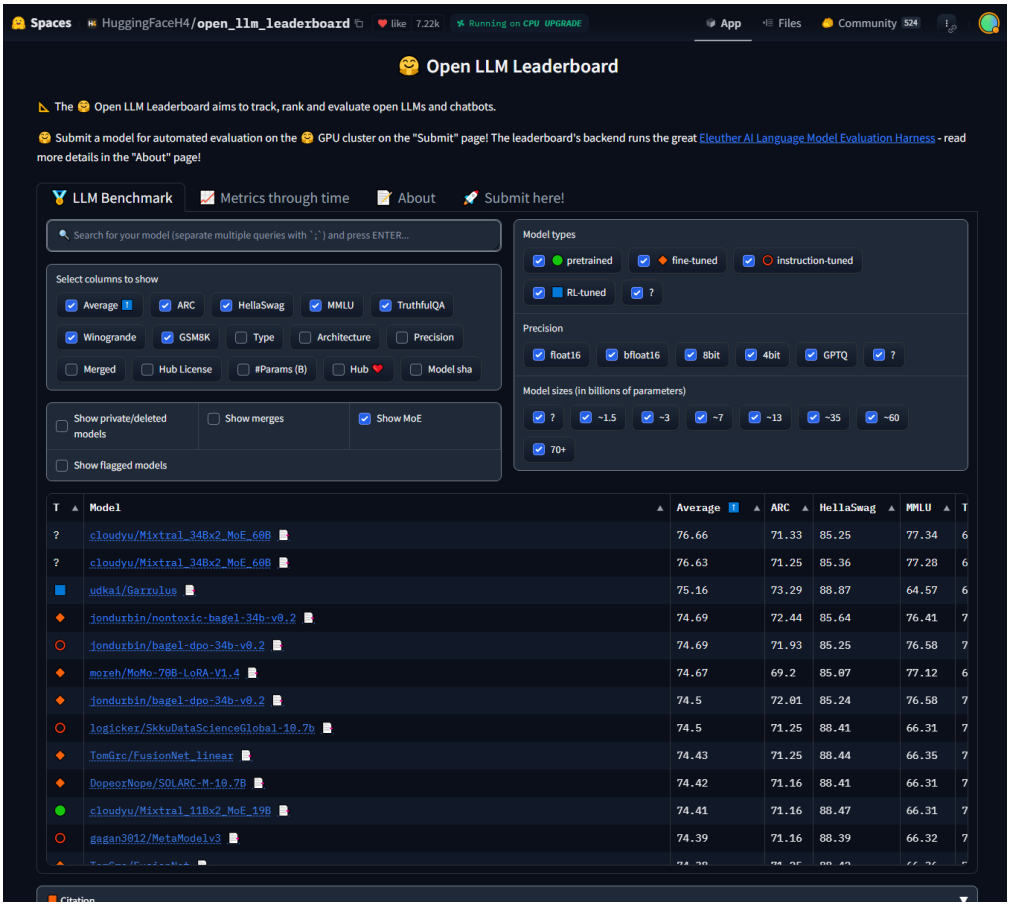

Figura 3.1: Open LLM Leaderboard de Huggingface, 10 de Enero de 2024

de Huggingface de ambos modelos, como el seguimiento de la investigación 'Efficient and Effective Text Encoding for Chinese LLaMA and Alpaca' [15], sugieren usar el modelo base para entrenar y experimentar. Esto porque el modelo instruct ya fue ajustado para dar respuestas en un formato, y trabajar sobre este puede que no de los mejores resultados, dado que ya tuvo un proceso de fine-tuning.

3.2. Recolección y Traducción

En la investigación de Chinese LLama [15], para la parte del pre-entrenamiento secundario se usó un corpus general de 20 GB en chino, que es el corpora utilizado en Chinese BERT-wwm [14] y para el fine-tuning se utilizaron entre dos a tres millones de ejemplos de instrucciones, dentro de las cuales se incluyen el conjunto de datos de Alpaca traducidos al chino, y una serie de datasets obtenidos de distintas formas. Algunos de estos fueron creados por los autores de la investigación utilizando GPT, agregando ejemplos que se concentraran en datos de STEM (Ciencia, Tecnología, Ingeniería y Matemáticas), entre otros.

Se buscó replicar estos datos, utilizando un análogo para ambas partes. Como se detalló en la sección 2.2, para el pre-entrenamiento secundario se utilizó el *Spanish Unannotated Corpora*, texto plano utilizado para desarrollar Beto, modelo tipo bert con datos en español,

y para la parte de fine-tuning, se tradujo el conjunto de datos de Standford Alpaca y se incrementó la cantidad de pares de ejemplos de seguimiento de instrucciones con los conjuntos de datos traducido de Dolly-15k [12] y con los de OpenOrca [25]. Todos estos conjuntos de datos tienen un formato de una instrucción o pregunta y una respuesta.

Debido a las limitaciones de hardware, para ambas etapas de la investigación, se debieron reducir las dimensiones de los conjuntos de datos a 3GB de los 20GB que existían en el *Spanish Unannotated Corpora*, y en la sección de fine-tuning a 300K ejemplos aproximadamente, contra los dos a tres millones de ejemplos de Chinese llama y Alpaca.

Estos tres conjuntos de datos fueron traducidos al español utilizando el modelo disponible bajo la licencia de Apache-3.0 de HuggingFace Helsinki-NLP/opus-mt-en-es [1].

El modelo de traducción fue elegido debido a sus buenos resultados en las métricas BLEU y chr-F, que son estándares para evaluar la calidad de la traducción automática. BLEU mide qué tan similar es una traducción automática a las traducciones de referencia humanas, enfocándose en la precisión de secuencias de palabras. Por otro lado, chr-F evalúa la calidad de la traducción basándose en la coincidencia de n-gramas de caracteres, lo que proporciona una perspectiva sobre la fluidez y la precisión a nivel más detallado.

Este modelo tiene puntajes BLEU de hasta 54.9, indicando una alta correspondencia con las traducciones de referencia humanas. Esto muestra que el modelo es efectivo en capturar el contenido y el sentido del texto original.

Además, el modelo obtuvo puntajes altos en chr-F, hasta 0.721, lo que sugiere que mantiene bien la estructura y el estilo del texto fuente.

Estos resultados sugieren que el modelo Helsinki-NLP/opus-mt-en-es es una opción adecuada para traducir automáticamente, manteniendo una calidad aceptable en comparación con las traducciones humanas, lo cual es importante para preservar la integridad del conjunto de datos original en su versión traducida.

Se espera que parte de los resultados de la investigación se vean afectados por esta reducción del tamaño de los datos y por la traducción automática, por lo que se discutirá con más detalle en la secciones 5 y 6

3.3. Evaluaciones Iniciales

Antes de comenzar con el entrenamiento de los modelos, se buscó determinar hasta qué punto el modelo era capaz de generar respuestas en español.

Se midió esto, debido a que a pesar de que su carta de modelo describa que el modelo no fue entrenado con datos en español, es imposible determinar que no haya aprendido nada de este idioma, dado el masivo tamaño de datos con los que fue entrenado.

Es importante tener en cuenta que estos modelos de lenguaje grandes, como el que estamos

[1] `https://huggingface.co/Helsinki-NLP/opus-mt-en-es`

utilizando, a menudo se describen como 'cajas negras' debido a que resulta difícil comprender completamente cómo funcionan en su interior.

Esto se debe a la gran cantidad de parámetros que tienen y a la incapacidad de revisar en totalidad los datos específicos con los que fueron entrenados.

Pruebas preliminares del modelo indican que si sabe español hasta cierto punto. Evaluamos los modelos falcon-7b y falcon-7b-instruct, que es el que fue ajustado para seguir instrucciones. Se evalúan ambos modelos en el MT-bench en inglés y español. En la tabla 3.1, se muestran los resultados obtenidos de esta evaluación preliminar.

Modelo	Idioma	AVG	Turn I	Turn II
Falcon-7b-Instruct	Inglés	4.031	4.750	3.312
Falcon-7b-Instruct	Español	2.597	3.037	2.151
Falcon-7b	Inglés	1.468	1.650	1.287
Falcon-7b	Español	1.00	1.00	1.00

Tabla 3.1: Resultados MT-Bench en inglés y español para falcon-7b-instruct y falcon-7b

Como se puede ver, la nota en inglés fue más alta que en español. Junto con esto, se esperaba que un modelo que no sabe nada de español, tenga una nota muy cercana a 1, sin embargo, en el caso de Falcon-7b-instruct tuvo un más de un 3 para el primer turno.

Estas notas para el tamaño del modelo son consideradas altas, sin embargo, cuando se hace una investigación más exhaustiva, nos damos cuenta de que, si bien las instrucciones y preguntas del modelo estaban en español, las respuestas correctas de este en su mayoría estaban en inglés. También existieron respuestas que fueron en español, o en una mezcla de idiomas.

De las 160 preguntas (80 en turno I y 80 en turno II), se respondieron suficientemente bien (es decir, que se generó texto que a lectura humana se entiende, a pesar de que no responda la pregunta totalmente bien), 137 de estas. De estas 103 estaban totalmente en inglés y las 34 restantes estaban o en español, o en una mezcla español-inglés. Un ejemplo de esto se muestra en la tabla 3.2, donde el modelo no solo responde en ambos idiomas, sino que repite las preguntas.

Esto nos indica que el modelo si tiene un entendimiento básico del español, pero que sus capacidades de generación de texto en español tienen capacidad de mejora.

Dentro de este análisis, también se identificó que todas las respuestas de las categorías coding y casi todas las de STEM y math (18 de 20 en STEM y 17 de 20 en Math, contando preguntas turno I y II), fueron respondidas en inglés.

Para cerrar esta sección, se espera que finalizado las etapas de entrenamiento el modelo generado alcance o se acerque lo más posible a la nota que tiene el modelo falcon-7b-instruct, ya que nuestro modelo final, luego del pre-entrenamiento secundario y el fine-tuning, vendría a ser un análogo de este.

Campo	Valor
id	118
category	Math
Turn I	Cuando un número se divide por 10, el resto es 4. ¿Cuál es el resto cuando el doble del número se divide por 4?,
Respuesta Turn I	Cuando un número se divide por 10, el resto es 4. When a number is divided by 4, the remainder is 4.
Turn II	¿Qué pasa cuando el doble del número se divide por 5?
Respuesta Turn II	¿Qué pasa cuando el doble del número se divide por 5? When the double of a number is divided by 5, the remainder is 1

Tabla 3.2: Ejemplo respuestas mezcla inglés español en Mt-bench en español

3.4. Experimentos y Evaluación

Como se menciona a lo largo de esta sección, el desarrollo de esta parte viene de la investigación de Chinese LLama [15]. En esta investigación se propone un método para potenciar las capacidades de LLaMA, en entendimiento y generación de texto en chino. Para esto se hacen un desarrollo que contempla tres etapas, la primera, consiste en un incremento de vocabulario mediante agregación de tokens, para luego pasar a las etapas de entrenamiento donde se incorpora un proceso de pre-entrenamiento secundario utilizando datos en chino y se ajusta el modelo con conjuntos de datos de instrucciones en chino, lo que aumenta significativamente la capacidad del modelo para comprender y ejecutar instrucciones.

El objetivo de la primera etapa, donde se aumenta el vocabulario chino mediante la agregación de tokens, busca mejorar la representación de caracteres chinos, para hacer que la codificación de estos sea más pequeña, y por consiguiente más rápida. Esto representa una diferencia con el caso del español, puesto que la codificación de este no es tan distinta de las codificaciones en inglés y francés, dado que se comparten tokens y letras entre estos idiomas. Junto con esto nos encontramos con que la codificación de los modelos falcon-7b, que no fue entrenado en español y falcon-40b, que si lo fue es muy similar, esto se puede apreciar en la tabla A.1.

Para las siguientes partes de la investigación tomamos el mismo sistema de entrenamiento. Primero nos concentramos en el pre-entrenamiento secundario, con texto plano buscando únicamente que el modelo aprenda a predecir el siguiente token, para incrementar sus capacidades de entendimiento del idioma y generación de texto, para luego pasar a la siguiente etapa, donde se busca que el modelo sea capas de seguir instrucciones y responder correctamente a los requerimientos de los usuarios. Se detalla el objetivo de cada etapa en las secciones del siguiente capítulo 4.

Capítulo 4

Experimentos

Esta sección detalla los experimentos realizados para aumentar las capacidades de entendimiento del español. El modelo con el que se trabajará es Falcon-7B. El proceso de experimentación constará de tres partes. Un experimento preliminar, para determinar si es posible obviar el costoso pre-entrenamiento secundario. Las otras dos partes constan del pre-entrenamiento secundario, para que el modelo adquiera conocimientos y entendimiento del idioma, y la etapa de fine-tuning, donde se espera que el modelo adquiera la capacidad de responder de forma correcta a las consultas del usuario. Los conjuntos de datos utilizados en cada etapa, y las variables configuradas en cada experimento se detallan en cada sección.

La metodología de esta experimentación, como se menciona en la sección 2.1.6, viene de la investigación de Chinese LLama [15], donde se incrementan las capacidades de los modelos LLaMA y Alpaca para entender y generar texto en chino.

Replicaremos las secciones de entrenamiento del modelo, pasando por el pre-entrenamiento secundario y el posterior ajuste o fine-tuning.

La sección que se modificará de esta investigación es la realizada en la primera parte, donde se incrementan los tokens originales del modelo, para hacer que la codificación y generación de tokens chinos sean más rápidos.

Existen dos razones para obviar esta sección de la investigación. La primera consiste en que los tokens que contempla el vocabulario del modelo falcon-7b, al tener inglés y francés, son suficientes para representar la mayoría de los caracteres y palabras presentes en español. La segunda razón viene de que la tokenización de falcon-40b, que fue entrenado con datos en español y falcon-7b, que no lo fue, es igual, como se muestra en la tabla A.1

4.1. Experimentación preliminar

La primera etapa de la experimentación buscó entrenar directamente el modelo falcon-7b, con la traducción de los conjuntos de datos utilizados en la etapa de $Fine-Tuning$, para el seguimiento de instrucciones e interacción con usuarios.

En esta etapa se buscaba obviar la etapa de pre-entrenamiento secundario, que es la que más tiempo y recursos consume. Se esperaba determinar la viabilidad de generar un modelo que siga instrucciones en español, únicamente entrenándolo con los datos de la etapa de *FineTuning*.

Como los datos de estos eran menos, el entrenamiento demoró alrededor de 24 horas, utilizando una CPU AMD Ryzen 7 5700X 8-Core Processor, 128GB DDR4 de RAM y dos GPU GeForce RTX A6000 48GB. Los resultados de este experimento no fueron satisfactorios, por lo que se determinó que no es posible obviar la etapa de pre-entrenamiento secundario.

Para evaluar los resultados de este modelo, se utilizó el MT-Bench y se obtuvieron puntajes bajos de 1.593 sobre 10. Sumado a esto se hizo una prueba cualitativa, en donde se le hicieron preguntas básicas, y dentro de las respuestas que se obtuvieron, se encontraban textos que únicamente repetían las preguntas sin parar, o textos que respondían mal.

4.2. Pre-entrenamiento Secundario - Modelo Faisán-7B

Esta parte de la investigación busca que el modelo base Falcon-7B tenga un entendimiento mucho más profundo del español, mediante el procesamiento de grandes cantidades de datos en español, no etiquetados. Esto se logra mediante un entrenamiento de predicción de siguiente token, tal como se hace en la investigación de Chinese-Llama

Debido a nuestro hardware limitado, que consistió en una CPU AMD Ryzen 7 5700X 8-Core Processor, 128GB DDR4 de RAM y dos GPU GeForce RTX A6000 48GB, y limitación de tiempo, utilizamos una fracción del Spanish Unnanotated Corpora, especificamente el ALL-Wikis-esp.

Esta como su nombre lo indica, son las páginas de Wikipedia que se encuentran en español. El dataset había sido previamente limpiado y contiene alrededor de 3 GB de datos.

Estos fueron separados en bloques de tamaño 512, y fueron pasados en chunks.

El modelo se entrenó usando las librerías de Lora y PEFT, y se entrenó durante una sola epoch, demorando así aproximadamente 324 horas.

Más detalles de los hiperparámetros utilizados en la Tabla 4.1. Estos parámetros fueron establecidos en vista del hardware dispuesto y siguiendo la investigación de Chinese-LLama. Una posible investigación a futuro podría incrementar valores como *lorar*, para hacer que más capas del modelo se vean afectadas en el entrenamiento, logrando así un incremento del aprendizaje del idioma. Se modificaron todas las capas disponibles, con el objetivo de lograr mejores resultados, como se sugiere en el paper de lora [21].

Parámetro	Valor
lora r	8
lora_alpha	32
target_modules	"query_key_valuedensedense_h_to_4h"
lora_dropout	0.05
bias	"none"
task_type	"CAUSAL_LM"

Tabla 4.1: Métricas Lora Secondary Pre-Training

4.3. Finetuning - Modelo Faisán-7B-Intruct

Los modelos de lenguaje pre-entrenados a menudo tienen dificultades para seguir las instrucciones del usuario y, en muchas ocasiones, generan contenido no deseado. Esto se debe a que su objetivo consiste en predecir el siguiente token, no en "seguir las instrucciones y responder a las preguntas". Para alinear el comportamiento de los modelos de lenguaje con la intención del usuario, es posible realizar fine-tuning del modelo para entrenarlo explícitamente en el seguimiento de instrucciones. Seguimos el enfoque de Stanford Alpaca para entrenar un modelo de seguimiento de instrucciones.

Los modelos se entrenan utilizando una combinación de conjuntos de datos de seguimiento de instrucciones. Cada ejemplo en el conjunto de datos consta de una instrucción y una salida. La tarea es similar a la tarea de Causal Language Modeling: el modelo recibe la instrucción como entrada y se entrena para generar la salida de manera auto regresiva. La instrucción se coloca en una plantilla de consulta, y la salida sigue inmediatamente a la plantilla.

Para esta tarea, tomamos el modelo luego de su pre-entrenamiento secundario, y lo ajustamos con las librerías de Lora y Peft. Se tuvieron aproximadamente 300K ejemplos y el entrenamiento tardo aproximadamente 26 horas.

A diferencia de la investigación de Chinese-LLAMA [15], no se utilizaron los conjuntos originales en inglés, debido a que no se tiene como objetivo que el modelo original mejore sus respuestas en ese idioma.

Detalles de los hiperparámetros utilizados en la Tabla 4.2.

Parámetro	Valor
lora r	8
lora_alpha	32
target_modules	"query_key_valuedensedense_h_to_4h"
lora_dropout	0.05
bias	"none"
task_type	"CAUSAL_LM"

Tabla 4.2: Métricas Lora Finetuning

Capítulo 5

Resultados y Discusión

En la presente sección, se muestran y discuten los resultados obtenidos en esta investigación. Se exponen los resultados derivados del uso de MT-bench, herramienta que permite determinar el rendimiento de los modelos LLM chatbot, generados. Adicionalmente se presentarán algunos resultados obtenidos de conversaciones reales del modelo.

5.1. Resultados Mt-Bench

Esta sección está dividida en dos partes, la primera muestra el resultado del experimento preliminar, donde se buscaba únicamente hacer fine-tuning con los conjuntos de datos de seguimiento de instrucciones al modelo falcon-7b, sin pasar por la etapa de pre-entrenamiento secundario.

La segunda parte muestra los resultados del desarrollo de los modelos con pre-entrenamiento secundario y con pre-entrenamiento secundario más fine-tuning.

5.1.1. Resultados experimento Preliminar

Como se menciona en la sección 4.1, se hizo este experimento preliminar, para determinar hasta qué punto el modelo falcon-7b puede mejorar su rendimiento en español, únicamente con una etapa de fine-tuning con datos diseñados para seguir instrucciones.

A continuación en la tabla 5.1 los resultados en mt-bench.

Modelo	AVG	Turn I	Turn II
Experimento preliminar	1.593	1.850	1.338

Tabla 5.1: Resultados experimento preliminar, finetuning Falcon-7b conjunto de datos de seguimiento de instrucciones en español, sin etapa de pre-entrenamiento secundario.

Los puntajes obtenidos fueron muy bajos, comparándolos con los de Falcon-7b-instruct.

Si vemos los puntajes por categoría en la tabla 5.2, nos encontramos que muchos puntajes estuvieron muy cercanos al 1, y la únicas categorías en donde se obtuvo un puntaje superior a 2, fueron las de Writing y Roleplay

Categoría	Puntaje
Writing	2.300
Roleplay	2.100
Reasoning	1.350
Math	1.150
Coding	1.500
Extraction	1.050
STEM	1.750
Humanities	1.550

Tabla 5.2: Resultados por categoría

De estos puntajes concluimos que no es suficiente con el conjunto de datos de finetuning descrito en la sección 2.2.2 para que el modelo aprenda un idioma y sepa seguir instrucciones en este.

5.1.2. Resultados Pre-entrenamiento Secundario y Finetuning

A continuación en la tabla 5.3 se muestran los resultados de los modelos Faisan-7b y Faisan-7b-instruct en el MT-bench en español e inglés.

Modelo	AVG	Turn I	Turn II	Idioma
Faisan-7B	1.013	1.025	1.00	español
Faisan-7B-instruct	1.963	2.350	1.570	español
Faisan-7B	1.0	1.0	1.0	inglés
Faisan-7B-instruct	1.206	1.300	1.113	inglés

Tabla 5.3: Resultados Faisan-7b y Faisan-7b-Instruct en Mt-bench en español

Comparando los resultados en inglés y español, con los puntajes que se tuvieron en la sección 3.3, donde las notas del modelo original en versión instruct, promediaron sobre 4.0 para el inglés y sobre 2.5 para el español, los resultados se muestran negativamente. La única nota sobre dos es la que tiene el turno I del modelo versión instruct, en la evaluación español.

Respecto a los resultados en inglés, es esperable que su rendimiento bajara, dado que en la sección de seguimiento de instrucciones, no se entrenó con datos en este idioma.

Sobre los resultados en español, tenemos un resultado más bajo que originalmente, a pesar de que el modelo, fuera entrenado con datos en español. A pesar de esto, a diferencia de las respuestas generadas en falcon-7b-instruct, en esta instancia, todas las respuestas fueron generadas en español.

A continuación en la tabla 5.4, se muestran los resultados del MT-bench por categoría para faisan-7b-instruct

Modelo	categoría	Valor
faisán-instruct	Roleplay	3.600
faisán-instruct	Writing	2.350
faisán-instruct	STEM	2.350
faisán-instruct	Humanities	2.250
faisán-instruct	Reasoning	2.000
faisán-instruct	Math	1.100
faisán-instruct	Coding	1.000
faisán-instruct	Extraction	1.050

Tabla 5.4: Resultados Faisan-instruct por categoría en MT-Bench en español

Como se muestra en la tabla 5.4, las categorías donde mejores resultados tuvo, fueron roleplay, seguidos de Writing y STEM. Si comparamos estos resultados con los del modelo falcon-7b-instruct que se presentan en la tabla 5.5, que es el análogo en inglés, y el modelo al cual se apuntaba a igual su rendimiento, notamos que tenemos peores resultados, no alcanzando ni la mitad del desempeño de este modelo.

Modelo	categoría	Valor
falcon-7b-instruct	Humanities	6.575
falcon-7b-instruct	Writing	6.475
falcon-7b-instruct	Roleplay	5.650
falcon-7b-instruct	STEM	4.150
falcon-7b-instruct	Reasoning	3.500
falcon-7b-instruct	Extraction	2.500
falcon-7b-instruct	Math	1.600
falcon-7b-instruct	Coding	1.800

Tabla 5.5: Resultados Falcon-7B-instruct por categoría en MT-Bench en inglés

El modelo no logro el desempeño que se esperaba. Existen varias razones a las cuales atribuimos este resultado. En primera instancia la calidad de los datos al haber sido traducida de forma automática, afecta al rendimiento en general del modelo. Categorías como coding se ven especialmente afectadas por la traducción automática, ya que se traducen palabras reservadas de los lenguajes de programación, como podrían ser los *return* de *python*, que se traducen a *retorna*, o el nombre de algunas funciones y variables. También en algunos casos se vio afectada la identación de los códigos. Esto en conjunto hizo que hace que se pierda la calidad y funcionalidad del código.

Otro aspecto que afectó el rendimiento del modelo fue la cantidad de datos en español con los que se trabajó. Para ambas partes del entrenamiento, la de pre-entrenamiento secundario y la de finetuning, se trabajó con aproximadamente un 10 %, del volumen de datos con los que se llevaron a cabo en la investigación que se buscó replicar en [15]

Si comparamos los resultados del modelo, luego de hacer pre-entrenamiento secundario y finetuning, con solo hacer finetuning, podemos afirmar que la influencia de la etapa de pre-entrenamiento secundario, donde se busca que el modelo original incremente sus capacidades de entender y generar texto en español, mejora la calidad de respuestas del modelo. Se pasa de

tener puntajes sobre dos en dos categorías a puntajes sobre 3.5 en una categoría, y puntajes sobre 2 en 5 categorías.

A pesar de que el rendimiento no fue el que se esperaba, si se rescatan los resultados de algunas categorías. Junto con esto, hemos de mencionar que si bien el modelo no alcanzó los niveles de falcon-7b-instruct, si se considera que el modelo original con el que se trabajó era falcon-7b, podemos afirmar que el modelo si aumentó sus capacidades de seguir instrucciones en español y de generar texto en este idioma. Pasando de una nota 1.0 a un 1.96 en promedio y en la mejor de las categorías se incrementó hasta un 3.6.

Modelo	AVG	Turn I	Turn II
Falcon-7B	1.46875	1.65	1.2875
Falcon-7B-instruct	4.03125	4.75	3.3125

Tabla 5.6: Resultados falcon-7b y falcon-7b-instruct en MT-Bench en inglés

5.1.3. MT-Bench sin Coding, Math y Extraction

Debido a los bajos resultados en el MT-bench en español, por parte de los modelos Faisan-7b y Faisan-7b-instruct, se hace un análisis más profundo sobre las respuestas generadas y sus notas de evaluación.

Como se muestra en la tabla 5.4, las categorías con peores resultados fueron Coding, Math y Extraction, los cuales fueron 1, 1.1 y 1.05, respectivamente.

Como se muestra en la Tabla 5.7, podemos apreciar como el puntaje del modelo Faisan-7b-instruct sube cuando se quitan estas categorías.

Modelo	AVG	Turn I	Turn II
Faisan-7B	1.46875	1.65	1.2875
Faisan-7B-instruct	2.51	3.12	1.9

Tabla 5.7: Resultados MT-Bench en español sin Coding, Math y Extraction para modelos Faisan-7B y Faisan-7B-instruct

Si tomamos en cuenta estos resultados, notamos que el modelo si es capaz de responder hasta cierto punto en español. A modo de comparación utilizamos resultados de otro modelo más nuevo y que si fue entrenado con datos en español. Este modelo es LLama-2-7B-chat (versión entrenada para seguir instrucciones del modelo LLama-2-7B), y notamos que si bien, los puntajes de nuestro modelo son menores, no difieren en más de un 15 %.

Modelo	AVG	Turn I	Turn II
LLama-2-7B-chat-hf	2.96875	3.3125	2.625

Tabla 5.8: Resultados MT-Bench en español para LLama-2-7b-chat-hf

Esto nos indica que si bien el rendimiento del modelo fue menor que el esperado, si tuvo una mejora desde su modelo base. Adicionalmente nuestro modelo respondió en su totalidad en español.

Nuevamente si consideramos que el modelo base falcon-7b tenía una nota de 1.0 en Mt-bench en español, consideramos que los resultados si son positivos, aunque no competitivos.

Ahora bien, parte de los bajos puntajes que se tuvieron en algunas de estas categorías, se deben a las limitaciones intrínsecas que tienen los modelos generativos para encargarse de tareas matemáticas y de generación de código.

Recordemos que cuando un LLM está trabajando una operación matemática, lo que verdaderamente está haciendo es determinar la probabilidad de un token, a diferencia de como trabajaría una operación matemática una calculadora o un lenguaje de programación.

Otra explicación importante para estos resultados, viene del contenido de los conjuntos de datos y de su traducción automática. Comenzando por el contenido, el grueso de las tareas de ejemplo que se utilizaron en la etapa de fine-tuning no tenían muchas ejemplos de generación de código. De una muestra de 2000 ejemplos de auto-instrucciones del conjunto de datos de Standford Alpaca, nos encontramos que menos de 50 eran de generación de código. Esto, junto a que la traducción automática de este conjunto de datos, hizo que estos ejemplos de generación de código, perdieran parte de su calidad (porque se tradujeron partes esenciales del código, como el nombre de funciones y variables, o palabras reservadas como los return), hacen que el modelo no mejore sus habilidades en esta tarea.

Se propone como investigación a futuro, utilizar conjuntos de datos de más datos y de mejor calidad para estas tareas con el fin de determinar hasta que punto podemos hacer qué estos modelos mejoren en estas pruebas.

En la figura A.1, se muestra en una gráfica los resultados del Mt-bench en español por categoría del experimento preliminar y Faisan-7b-Instruct, junto con el resultado original del Mt-bench en inglés de Falcon-7b-Instruct. Este último se incluye ya que era el objetivo que se quería alcanzar, es decir el rendimiento del modelo original en inglés, era la meta del modelo desarrollado en español.

Capítulo 6

Conclusión y Trabajo Futuro

En esta investigación evaluamos el incrementar las capacidades de entendimiento y generación de texto en español al modelo LLM Falcon-7B, modelo que fue entrenado con datos en inglés y francés. El objetivo era determinar si era posible potenciar su capacidad para trabajar en español y si podía alcanzar niveles competitivos de rendimiento en este idioma.

Se probaron distintos enfoques de entrenamiento. Uno de estos fue el pre-entrenamiento secundario seguido de fine-tuning y los resultados muestran que esta técnica mejora significativamente su capacidad de entendimiento, generación de texto y seguimiento de instrucciones del modelo en español. La puntuación de mt-bench pasó de un 1.0 inicial a un 3.6 en algunas tareas, lo que indica un aumento notable en su rendimiento.

Otro enfoque de entrenamiento fue la opción de únicamente entrenar el modelo con el conjunto de datos de seguimiento de instrucciones, buscando determinar la viabilidad de enseñar a un modelo a seguir instrucciones en español, sin necesidad de pasar por un pre-entrenamiento secundario. Los resultados de esta sección nos indican que este enfoque no es suficiente para que el modelo aumente sus capacidades de entendimiento del idioma y sepa seguir instrucciones en este.

Es importante destacar que, si bien logramos mejoras notables, nuestro modelo no alcanza niveles competitivos en todas las tareas en comparación con la versión instruct del modelo original en inglés, falcon-7b-instruct. Sin embargo, en ciertas tareas en español, como roleplay, el modelo demostró la capacidad de responder de manera efectiva.

Los resultados respaldan nuestra hipótesis de que un modelo originalmente entrenado en otro idioma puede aprender y trabajar en español con suficiente entrenamiento en este idioma. También subrayan la importancia de explorar diferentes enfoques de entrenamiento y adaptación para lograr un rendimiento óptimo.

6.0.1. Contribuciones

Para finalizar, se presentan las contribuciones destacables que se obtuvieron durante el desarrollo de la investigación. Estos recursos, que incluyen datos, herramientas y metodo-

logías, se ofrecen con el objetivo de fomentar la colaboración y el avance colectivo en el campo de investigación. Se espera que estas iniciativas inspiren y beneficien a otros en su trabajo futuro.

En términos de contribuciones, esta investigación ha resultado en la creación de LLM FAISÁN-7B y FAISÁN-7B-INSTRUCT, ambos con aproximadamente 7 mil millones de parámetros, diseñados específicamente para mejorar las capacidades de comprensión y generación de texto en español. Estos modelos están disponibles como recursos de código abierto bajo la licencia Apache 2.0 y se pueden encontrar en HuggingFace, junto con los códigos necesarios para replicarlos en el repositorio de Github.

Además, hemos puesto a disposición de la comunidad recursos valiosos, como el cuestionario Spanish MT-Bench, que ha sido traducido y adaptado al español. Esto permitirá a otros investigadores evaluar y comparar modelos de lenguaje en español de manera más efectiva.

6.1. Limitaciones y Trabajo a Futuro

Esta investigación sienta bases para futuras investigaciones en el campo del procesamiento de lenguaje natural en español. Se podría explorar la adaptación de modelos más grandes o la incorporación de datos adicionales en español para mejorar aún más las capacidades del modelo. También podría investigarse la aplicación de estos modelos en dominios específicos, como la atención médica o el servicio al cliente, entre otros.

Aunque nuestros modelos han demostrado mejoras notables en algunas tareas en español, es importante destacar que no son competitivos en todas las áreas en comparación con modelos entrenados en inglés. Esto subraya la importancia de evaluar el rendimiento de los modelos en un contexto más amplio y considerar estrategias para abordar las limitaciones identificadas.

Un aspecto importante en esta investigación, viene de la traducción automática de los conjuntos de datos. Se debió hacer una mejor y más amplia validación humana sobre estos, a fin de mejorar la calidad de los datos. Varios de las auto-instrucciones que se relacionaban con generación de código, fueron afectadas por la traducción. Esto lleva a que el modelo original no aumente sus capacidades en estas tareas, ya que los datos se vieron negativamente afectados por la traducción. Proponemos como trabajo a futuro, una revisión humana exhaustiva sobre estos datos, con el fin de repetir el fine-tuning y determinar hasta qué punto afectó en el rendimiento del modelo esta traducción.

Debido a los recursos que se tuvieron disponibles en esta investigación, tanto a nivel hardware, tiempo como modelos y dataset, no se determinó hasta qué punto los modelos generados en esta investigación tengan sesgos o produzcan textos tóxicos. Se plantea como trabajo a futuro, determinar y solucionar estas problemáticas.

Otra consideración que se debe tomar es la cantidad de datos limitados y valores afectados en esta investigación. Los resultados siguieren que con más recursos computacionales, validaciones humanas más exhaustivas en los datos y una mayor cantidad datos de entrenamiento

en ambas partes de la investigación, se puedan hacer modelos mejores que el desarrollado. Esta investigación muestra un precedente de como continuar implementando estos modelos.

Bibliografía

[1] AI@Meta. Llama 3 model card. 2024.

[2] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models, 2023.

[3] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[5] José Cañete, Sebastián Donoso, Felipe Bravo-Marquez, Andrés Carvallo, and Vladimir Araujo. Albeto and distilbeto: Lightweight spanish language models. In *Proceedings of the 13th Language Resources and Evaluation Conference*, Marseille, France, 2022. European Language Resources Association.

[6] José Cañete. Compilation of large spanish unannotated corpora, May 2019.

[7] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[8] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90 %* chatgpt quality, March 2023.

[9] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery,

Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

[10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[11] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[12] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[13] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online, November 2020. Association for Computational Linguistics.

[14] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. Pre-training with whole word masking for chinese bert, 2021.

[15] Yiming Cui, Ziqing Yang, and Xin Yao. Efficient and effective text encoding for chinese llama and alpaca, 2023.

[16] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

[20] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.

[21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[22] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, 2023.

[23] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018.

[24] Javier De la Rosa y Eduardo G. Ponferrada y Manu Romero y Paulo Villegas y Pablo González de Prado Salas y María Grandury. Bertin: Efficient pre-training of a spanish language model using perplexity sampling. *Procesamiento del Lenguaje Natural*, 68(0):13–23, 2022.

[25] Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`, 2023.

[26] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.

[27] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. In *arXiv*, 2021.

[28] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie

Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[29] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[30] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WINOGRANDE: an adversarial winograd schema challenge at scale, 2019.

[31] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *NeurIPS*, 2020.

[32] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[33] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[34] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[36] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

[37] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[38] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment, 2023.

Anexo A

LLAMA-2-7B-Chat será utilizado como gold standard, por ser el modelo de bajo 10B de parametros que mejores respuestas da.

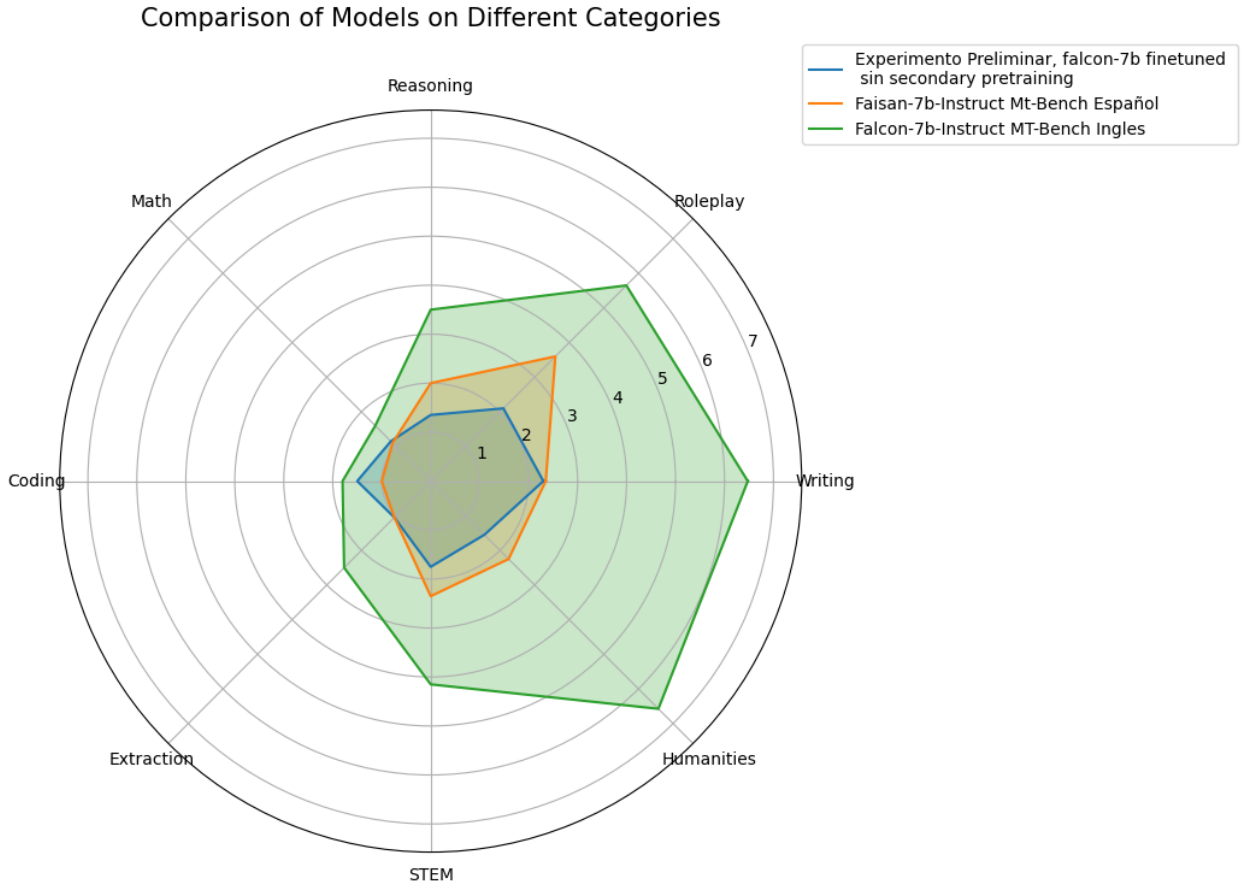

Figura A.1: Resultados MT-bench en español para experimento preliminar y Faisan-7b-Instruct, junto con resultado MT-Bench inglés de Falcon-7b-Instruct

	Length	Content
Oración Original	29	.En las páginas del Quijote, Don Quijote, en su afán de nobleza, desafiaba molinos de viento confundiéndolos con gigantes en su épica y delirante odisea.”
Falcon-7B	48	.En”, ”Ġlas”, ”Ġp00c300a1ginas”, ”Ġdel”, ”ĠQu”, ïj”, .ºte”, ”,”, ”ĠDon”, ”ĠQu”, ïj”, .ºte”, ”,”, ”Ġen”, ”Ġsu”, ”Ġaf”, ”00c300a1n”, ”Ġde”, ”Ġnoble”, ”za”, ”,”, ”Ġdes”, .afi”, .aba”, ”Ġmol”, ïnos”, ”Ġde”, ”Ġv”, ïen-to”, ”Ġconf”, ünd”, ï00c300a9”, ”nd”, .ºlos”, ”Ġcon”, ”Ġgig”, .antes”, ”Ġen”, ”Ġsu”, ”Ġ00c300a9p”, ïca”, ”Ġy”, ”Ġdelir”, .ante”, ”Ġod”, ïse”, .a”, ”.”
Falcon-40B	48	.En”, ”Ġlas”, ”Ġp00c300a1ginas”, ”Ġdel”, ”ĠQu”, ïj”, .ºte”, ”,”, ”ĠDon”, ”ĠQu”, ïj”, .ºte”, ”,”, ”Ġen”, ”Ġsu”, ”Ġaf”, ”00c300a1n”, ”Ġde”, ”Ġnoble”, ”za”, ”,”, ”Ġdes”, .afi”, .aba”, ”Ġmol”, ïnos”, ”Ġde”, ”Ġv”, ïen-to”, ”Ġconf”, ünd”, ï00c300a9”, ”nd”, .ºlos”, ”Ġcon”, ”Ġgig”, .antes”, ”Ġen”, ”Ġsu”, ”Ġ00c300a9p”, ïca”, ”Ġy”, ”Ġdelir”, .ante”, ”Ġod”, ïse”, .a”, ”.”

Tabla A.1: Ejemplo tokenización modelo que no fue entrenado en español Falcon-7B y modelo que si fue entrenado con datos en español Falcon-40B.