UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

URKU: Adaptación de LlaMA 2 para la generación de texto en Kichwa usando técnicas de Low-Rank Adaptation (LoRA)

James Armando León Caranqui

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 23 de noviembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

URKU: Adaptación de LlaMA 2 para la generación de texto en Kichwa usando técnicas de Low-Rank Adaptation (LoRA)

James Armando León Caranqui

Daniel Riofrío, PhD

Quito, 7 de diciembre de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: James Armando León Caranqui

Código: 213782

Cédula de identidad: 0604621185

Lugar y fecha: Quito, 4 de octubre de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En este estudio, introducimos una metodología innovadora para entrenar modelos de lenguaje de gran escala (LLMs, por sus siglas en inglés) en idiomas con recursos digitales limitados, utilizando técnicas de Adaptación de Bajo Rango (LoRA). Centrado en el idioma Kichwa—un idioma lingüísticamente desatendido pero culturalmente rico, hablado por más de medio millón de personas en Ecuador—nuestro enfoque aprovecha la robustez de la arquitectura Transformer, conocida por su eficacia en tareas de traducción automática. Dada la escasez de datos en Kichwa, empleamos técnicas de LoRA para ajustar un modelo pre-entrenado, superando así los desafíos típicamente asociados con idiomas de bajos recursos.

Para evaluar el rendimiento de nuestro modelo, nos basamos en un benchmark desarrollado en colaboración con antropólogos, inspirado en la estructura de las evaluaciones de Flores-200. Nuestros hallazgos revelan que el modelo establece un nuevo estándar en la traducción de Kichwa, superando significativamente a los modelos anteriores, que en su mayoría fueron entrenados en dialectos quechuas peruanos. Más allá de su aplicación inmediata en la traducción automática, nuestro trabajo aclara la aplicabilidad más amplia de las técnicas de LoRA en el entrenamiento de LLMs para idiomas de bajos recursos, extendiendo así el alcance de tecnologías para resumir texto y responder a preguntas a comunidades kichwa hablantes. Además, esta investigación tiene un gran potencial para facilitar la creación y accesibilidad de contenido en el idioma nativo de estas comunidades. Ya sea en el desarrollo de herramientas de traducción automática para Kichwa o en la creación de recursos educativos, nuestro modelo sirve como piedra angular para la preservación lingüística y cultural de este idioma nativo.

Palabras clave: LoRA, Transformer, modelado de lenguaje de bajos recursos, Kichwa, traducción automática, LlaMA 2.

ABSTRACT

In this study, we introduce a groundbreaking methodology for training large language models (LLMs) on languages with limited digital resources, utilizing Low-Rank Adaptation (LoRA) techniques. Focused on the Kichwa language—a linguistically underserved yet culturally rich language spoken by over half a million individuals in Ecuador and neighboring regions—our approach leverages the robustness of the Transformer architecture, renowned for its efficacy in machine translation tasks. Given the scarcity of Kichwa data, we employ LoRA techniques to fine-tune a pre-existing Transformer model, thereby overcoming the challenges typically associated with low-resource languages.

To assess the performance of our model, we rely on a specialized benchmark developed in collaboration with anthropologists, inspired by the structure of the Flores-200 benchmark. Our findings reveal that the model sets a new standard in Kichwa translation, significantly outpacing prior models mostly trained on Peruvian Quechua dialects. Beyond its immediate application in machine translation, our work elucidates the broader applicability of LoRA techniques in training LLMs for low-resource languages, thereby extending the reach of technologies like text summarization and question-answering to these communities. Moreover, this research holds substantial promise for the Kichwa-speaking community by facilitating the creation and accessibility of content in their native language. Whether it's the development of machine translation tools for Kichwa or the creation of educational resources, our model serves as a cornerstone for linguistic and cultural preservation.

Keywords: LoRA, Transformer, low-resource language modeling, Kichwa, machine translation, LlaMA 2

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

INTRODUCCIÓN

Contexto y Justificación

El escenario socio-lingüístico en Ecuador es un vivo reflejo de un extenso mosaico pluricultural, donde un 3,86% de la población equivalente a cerca de 654.316 individuos mantienen viva la lengua indígena Kichwa. A pesar de que esta cifra exhibe una tendencia decreciente, es imperativo resaltar que el 7,69% de la población ecuatoriana (1.301.887 personas) se asocia con comunidades indígenas, mientras que un notable 43% (7.336.265 personas) se identifica como mestizos con algún grado de ascendencia Kichwa. En términos absolutos, aproximadamente 9 millones de personas en el país conservan una intrínseca conexión con la herencia cultural y lingüística indígena, dentro de una población total que se aproxima a los 17 millones de habitantes [1].

La declinación en el número de hablantes nativos de Kichwa no emerge como un fenómeno aislado, sino que se revela como el reflejo de contextos pragmáticos. Ecuador, pese a sus loables esfuerzos por promover la inclusión, se caracteriza por una predominancia lingüística hispana. Esta realidad impulsa a aquellos individuos que persiguen un futuro urbano y profesional a adquirir un dominio competente del español. En este contexto, se observa una preferencia hacia la transmisión del español a las nuevas generaciones como vehículo para acceder a una educación superior y, en consecuencia, a oportunidades ampliadas de movilidad social. Esta dinámica ha catalizado una espiral descendente en el número de hablantes nativos de Kichwa, exacerbando la marginalización de aquellos que no poseen un dominio del español.

Ante la intersección de esta problemática social con los notables avances en las tecnologías de procesamiento de lenguaje natural (NLP, por sus siglas en inglés), se descifra una oportunidad inédita. Se vislumbra una brecha social que podría ser mitigada mediante la implementación de modelos de lenguaje sofisticados, tales como el modelo LlaMa 2, para brindar soporte a

comunidades lingüísticamente desatendidas. El impulso subyacente reside en adaptar dicho modelo para funcionar eficazmente en escenarios de "datos escasos", un aspecto primordial para lenguajes con recursos digitales limitados.

El fulcro de este estudio es evaluar el rendimiento del modelo adaptado en aplicaciones prácticas, permitiendo que hablantes nativos de Kichwa califiquen tanto la generación como la traducción de textos. Concomitantemente, se aspira a diseñar un marco de evaluación automática anclado en métricas de vanguardia, proporcionando un método de evaluación replicable y confiable. En esencia, este trabajo se avoca a adaptar tecnologías preexistentes para atender casos lingüísticos desatendidos, pero de una importancia cultural y social indiscutible. En las secciones subsiguientes, se desglosará el marco teórico que sustenta este estudio, se delineará la metodología empleada, y se discutirán los hallazgos e implicaciones de la adaptación del modelo LlaMa 2 mediante la técnica de Low Rank Adaptation para la generación de textos en Kichwa.

Objetivos Generales y Específicos

Generales

- Adaptar el Modelo LlaMA 2 al Idioma Kichwa mediante Técnicas de Procesamiento de Lenguaje Natural (NLP): Este objetivo enfatiza la adaptación y optimización del modelo para un procesamiento eficaz del Kichwa.
- Diseñar un Protocolo de Evaluación Riguroso para Asegurar la Eficacia y Precisión del Modelo Adaptado: Involucra el establecimiento de métricas y pruebas para una evaluación objetiva y replicable.

Específicos

- Crear un Corpus de Alta Calidad Siguiendo los Principios de las Arquitecturas de Transformadores en Aprendizaje Profundo: Centrado en generar un conjunto de datos robusto y representativo.
- Utilizar la Técnica de Adaptación de Bajo Rango (Low-Rank Adaptation) para Facilitar la Adaptación del Modelo a Lenguajes de Recursos Limitados: Busca una metodología eficaz para la adaptación sin comprometer la calidad.
- Establecer un Benchmark Específico para Evaluar la Generación y Traducción de Textos por el Modelo Adaptado: Enfocado en la creación de pruebas estandarizadas para evaluar el rendimiento del modelo.
- Validar 'URKU', el Modelo Adaptado, con Hablantes Nativos y Expertos en Kichwa: Introduce una evaluación humana, permitiendo una valoración tanto técnica como interactiva del modelo.

Alcance y Limitaciones

Alcance

1. **Desarrollo de un Modelo de Lenguaje entrenado en Kichwa ecuatoriano:** El núcleo de este proyecto radica en adaptar y afinar un modelo de lenguaje robusto entrenado por Meta, LlaMA 2, para el procesamiento de texto en Kichwa, uno de los idiomas indígenas del ecuador.
2. **Evaluación Rigurosa:** El consejo de profesionales en el idioma ha permitido la construcción de un corpus de datos robusto que permitirá explorar el desempeño del modelo afinado en tareas de generación y traducción de texto a partir de criterios de evaluación a discreción de Kuymi Tambaco, hablante nativa y catedrática del idioma.

3. **Contribución a la Comunidad Lingüística:** A través del desarrollo y la evaluación de este modelo, se busca contribuir al enriquecimiento digital del Kichwa, facilitando su uso y, especialmente, impulsando su preservación en la era digital como testimonio de la tradición indígena ecuatoriana.

Limitaciones

1. **Recursos Limitados:** La disponibilidad de fuentes de datos digitales para el Kichwa es en extremo limitada, lo cual puede impactar en la capacidad del modelo para aprender y generalizar efectivamente.
2. **Diversidad Dialectal:** Más allá de los recursos disponibles para la creación del corpus, es esencial atender a la variedad de dialectos que comprenden el Kichwa. Esto puede presentar desafíos en la construcción de un modelo representativo lo mismo que funcional. En consecuencia, existe un 'trade-off' importante entre representatividad y funcionalidad.
3. **Expertise Tecnológico para trabajos futuros:** La necesidad de conocimiento especializado para la adaptación y evaluación del modelo podría representar una barrera para futuras iteraciones y adaptaciones comunitarias del modelo resultante de este proyecto.

Estructura del Corpus de Datos

El corpus de datos constituye la piedra angular para la adaptación del modelo de lenguaje LlaMa 2 al Kichwa. A continuación, se desglosan los componentes principales de este corpus:

Recolección de Datos:

- **Fuentes Primarias:** Se ha hecho acopio de textos auténticos en Kichwa a partir de una variedad de fuentes, incluyendo literatura y transcripciones de entrevistas con hablantes nativos.
- **Fuentes Secundarias:** Se ha recurrido también a traducciones bilingües de textos entre el español y el Kichwa, proporcionando una rica fuente de datos paralelos. A su vez, se han utilizado entrevistas realizadas por Simeon Floyd como parte del proyecto 'Prometeo' de la secretaría de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador.

Preprocesamiento de Datos:

- **Normalización:** Los datos han sido normalizados para asegurar la consistencia en la ortografía y la gramática, facilitando así el entrenamiento del modelo. Este proceso ha sido manual, a pesar de usar expresiones regulares en la fase inicial.
- **Segmentación:** Se ha procedido a la segmentación del texto en sentencias y palabras, estructurando el corpus para una ingestión efectiva por parte del modelo. Al igual que en el apartado anterior, para asegurar la calidad de los datos se han recurrido a métodos manuales como complemento al procesamiento con expresiones regulares.

Almacenamiento y Accesibilidad:

- **Almacenamiento en ficheros:** El corpus se ha almacenado en tres diccionarios de tipo JSON estructurados en formato Alpaca, asegurando su accesibilidad y manejo eficiente para las etapas de entrenamiento y evaluación del modelo.

Este corpus, meticulosamente construído y estructurado, proporciona una fundación sólida para la adaptación de LlaMa 2 al Kichwa, promoviendo un entendimiento más profundo y una apreciación más rica de la lengua Kichwa en el ámbito digital. Así también, el formato de los datasets resultantes facilitan el entrenamiento de LoRAs al tener todos los datos segmentados.

ESTADO DEL ARTE

Evolución del Procesamiento de Lenguaje Natural (NLP)

El Procesamiento de Lenguaje Natural (NLP, por sus siglas en inglés) emerge como una disciplina que anida en la confluencia de la lingüística computacional y la inteligencia artificial, procurando facultar a las máquinas con una comprensión y generación de lenguaje humano que simula la capacidad interpretativa humana. A lo largo de las últimas décadas, esta disciplina ha atravesado un notable recorrido evolutivo, marcado por hitos que han redefinido los contornos del procesamiento de lenguaje y sus aplicaciones prácticas.

En las albores del NLP, las aproximaciones eran regidas por reglas heurísticas y gramáticas formales, cuyas limitaciones se evidenciaban en su rigidez y su incapacidad para generalizar a partir de ejemplos no vistos. Con el advenimiento de modelos probabilísticos y estadísticos, como los Campos Aleatorios Condicionales (CRF) propuestos por Lafferty et al., se inauguró una era donde la segmentación y etiquetado de datos de secuencia se tornó más eficaz, permitiendo una gama amplia de aplicaciones que abarcaban desde el análisis de sentimientos hasta la traducción automática [2]. La irrupción de los modelos de aprendizaje profundo, específicamente las redes neuronales recurrentes (RNN) y las redes neuronales convolucionales (CNN), facilitó la captura de relaciones semánticas y sintácticas de largo alcance en los textos, propiciando avances en tareas como la traducción automática y la generación de texto. Sin embargo, fue la llegada de los modelos de transformadores,

introducidos por Vaswani et al., lo que marcó un punto de inflexión en el campo del NLP, ofreciendo una arquitectura capaz de manejar dependencias de largo alcance de manera más eficaz y en paralelo, a diferencia de las arquitecturas precedentes [3].

Los transformadores catalizaron el desarrollo de modelos de lenguaje de gran escala como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer), que lograron capturar y generar lenguaje humano con una fidelidad sin precedentes. BERT, con su enfoque bidireccional, y GPT, con su enfoque autoregresivo, se destacaron en una plétora de tareas de NLP, estableciendo nuevos estándares de rendimiento. Con un enfoque en la eficiencia y la adaptabilidad, el modelo de lenguaje LlaMa 2 emerge como una alternativa robusta de código abierto, destaca por su capacidad en tareas de traducción y generación de texto, aún en escenarios de entrenamiento 'few-shot'[4]. LlaMa 2 se sitúa en una tradición de modelos que buscan balancear la capacidad de representación con la eficiencia computacional y la adaptabilidad a diferentes idiomas y dominios.

Avanzando cronológicamente, BERT representa un hito en el campo del NLP. Este modelo de representación de lenguaje pre-entrena representaciones bidireccionales profundas de texto no etiquetado, condicionando conjuntamente tanto el contexto izquierdo como el derecho en todas las capas. Esto significa que, en lugar de predecir una palabra en función de su contexto anterior, BERT tiene en cuenta el contexto completo de una palabra, tanto a la izquierda como a la derecha, para predecir su significado. Este enfoque bidireccional permite a BERT capturar relaciones semánticas más ricas y complejas en el texto, lo que lo hace especialmente poderoso para tareas como la respuesta a preguntas y la inferencia de lenguaje. Además, BERT está diseñado para ser afinado con solo una capa de salida adicional, lo que facilita la creación de modelos de vanguardia para una amplia gama de tareas sin necesidad de modificaciones arquitectónicas sustanciales.

Los resultados obtenidos por BERT en diversas tareas de procesamiento de lenguaje natural son un claro testimonio de su capacidad para capturar y generar lenguaje humano con una fidelidad sin precedentes. En evaluaciones específicas, BERT ha logrado una mejora de rendimiento frente a sus predecesores en once tareas de procesamiento de lenguaje natural, incluyendo el aumento del puntaje GLUE a 80.5% (7.7 puntos porcentuales de mejora absoluta), la precisión MultiNLI a 86.7% (4.6 puntos porcentuales de mejora absoluta), el Test F1 de SQuAD v1.1 para responder preguntas a 93.2 (1.5 puntos porcentuales de mejora absoluta) y el Test F1 de SQuAD v2.0 a 83.1 (5.1 puntos porcentuales de mejora absoluta) [5]. Además de estas tareas, BERT también ha demostrado ser eficaz en una amplia variedad de otras tareas de lenguaje, como análisis de sentimientos, respuesta a preguntas, predicción de texto, generación de texto, resumen y resolución de polisemia [5]. Estos resultados son una clara indicación de la versatilidad y el poder de BERT para transformar el campo del procesamiento de lenguaje natural y abrir nuevas posibilidades para la interacción entre humanos y máquinas.

En seguida, aparece GPT-4, el modelo de procesamiento de lenguaje natural de última generación de OpenAI, que representa un avance significativo en la arquitectura de modelos de lenguaje de gran escala. Basado en la arquitectura Transformer, GPT-4 es un modelo multimodal de gran escala que puede aceptar entradas de texto e imágenes y producir salidas de texto, manejando entradas que consisten en texto e imágenes intercaladas arbitrariamente y generando salidas para cualquier tarea de visión o lenguaje [6]. A pesar de su gran cantidad de parámetros y el uso de una gran cantidad de cómputo para el entrenamiento, los detalles exactos no se divulgan en el documento debido a consideraciones de competitividad y seguridad. GPT-4 utiliza un proceso de alineación post-entrenamiento que resulta en una mejora del rendimiento en medidas de factualidad y adherencia al comportamiento deseado, involucrando el aprendizaje por refuerzo a partir de retroalimentación humana (RLHF) y modelos de

recompensa basados en reglas (RBRMs) que proporcionan señales de recompensa adicionales para guiar el modelo hacia un comportamiento apropiado. Su arquitectura basada en Transformers utiliza un mecanismo de atención para aprender relaciones contextuales entre palabras en un texto, y una técnica de auto-atención enmascarada que permite al modelo atender tanto al contexto izquierdo como al derecho en cada capa, similar al presentado por BERT. Además, los modelos GPT pueden ser afinados para tareas específicas, como clasificación de texto, respuesta a preguntas, resumen, etc., mediante la adición de una capa de salida específica de la tarea y la actualización de los parámetros del modelo preentrenado utilizando un objetivo de aprendizaje supervisado. Con el tiempo, los modelos GPT han aumentado en tamaño y rendimiento, y GPT-4, con sus 1 trillón de parámetros, es un testimonio de la evolución y el potencial de estos modelos para transformar el campo del procesamiento de lenguaje natural y abrir nuevas posibilidades para la interacción entre humanos y máquinas. Finalmente, entra Llama 2 como una colección de modelos de lenguaje preentrenados y afinados que varían en escala desde 7 mil millones hasta 70 mil millones de parámetros. Los modelos afinados, denominados Llama 2-Chat, están optimizados para casos de uso de diálogo. Según las evaluaciones, estos modelos superan a los demás modelos de chat de código abierto en la mayoría de los benchmarks probados, y podrían ser un sustituto adecuado para modelos de fuente cerrada basado en evaluaciones humanas para utilidad y seguridad. Esta iniciativa proporciona una descripción detallada de la metodología de afinación y las mejoras en la seguridad de Llama 2-Chat, permitiendo a la comunidad construir sobre este trabajo y contribuir al desarrollo responsable de modelos de lenguaje de gran escala [4].

El NLP contemporáneo, enriquecido por estos avances, se ha expandido hacia la inclusión de técnicas de adaptación y aprendizaje reforzado, en aras de construir modelos más resilientes y adaptativos a variadas condiciones y dominios lingüísticos. La evolución del NLP ha trazado un arco desde reglas heurísticas hasta modelos de aprendizaje profundo altamente sofisticados,

presagiando un futuro donde las máquinas podrían alcanzar una comprensión del lenguaje humano más profunda y contextualizada. En este escenario, para entender cómo funciona Llama 2, es necesario comprender la arquitectura de los modelos de lenguaje de gran escala basados en la arquitectura Transformer, que utiliza mecanismos de atención para capturar relaciones semánticas y sintácticas en el texto. Los Transformers se componen de una serie de capas de atención, cada una de las cuales procesa el texto de entrada de manera paralela, lo que permite una mayor eficiencia en comparación con las arquitecturas de red recurrentes y convolucionales. La ecuación fundamental que rige el funcionamiento de un Transformer es la siguiente: $Atención(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

Donde *Q*, *K* y *V* son las matrices de consulta, clave y valor, respectivamente, y d_k es la dimensión de las claves. Esta ecuación calcula una serie de pesos de atención que se utilizan para ponderar los valores en función de la similitud entre las consultas y las claves. Estos pesos de atención se utilizan luego para calcular una representación ponderada de los valores, que se utiliza como entrada para la siguiente capa del modelo. En el caso de Llama 2, la arquitectura del modelo seleccionado ha realizado ajustes en la forma en que se procesa el texto de entrada y se generan las respuestas. Estos ajustes se describen en detalle en la documentación de Llama 2, y son el resultado de una extensa experimentación y ajuste fino para maximizar el rendimiento del modelo en tareas de diálogo que se discutirán en la siguiente sección.

Técnicas de Adaptación de Bajo Rango (LoRAs)

Dentro del vasto dominio de las técnicas de adaptación en aprendizaje profundo, las Técnicas de Adaptación de Bajo Rango (LoRAs, por sus siglas en inglés) emergen como un conjunto prometedor para la tarea de afinar modelos pre-entrenados, especialmente en escenarios de recursos limitados. Las LoRAs operan bajo la premisa de modificar estructuras neuronales

preexistentes, manteniendo una eficiencia computacional significativa, lo cual es crucial para aplicaciones en tiempo real y en dispositivos con recursos computacionales restringidos. Las Técnicas de LoRA modifican los pesos preentrenados de un modelo de lenguaje grande (LLM) de una manera matemáticamente eficiente. En un modelo Transformer, como LlaMa 2, los pesos preentrenados se encuentran en las matrices de atención y en las capas completamente conectadas. LoRA introduce una modificación en estas matrices mediante la adición de una matriz de bajo rango W_{low} a la matriz de pesos preentrenados W. Matemáticamente, si W es una matriz de dimensión $d \times h$, donde d es la dimensión del modelo y h es el tamaño de la capa oculta, LoRA busca una matriz W_{low} de dimensión $d \times r$ y $r \times h$, donde r es el rango, que es mucho menor que d y h. La actualización de los pesos se realiza entonces como:

$$W' = W + W_{low}$$

Donde W' es la nueva matriz de pesos ajustada. Este proceso no requiere que se reentrenen todos los parámetros del modelo, lo que reduce significativamente la cantidad de cálculos y la memoria necesaria para la adaptación. Durante el entrenamiento o la afinación, solo las matrices de bajo rango W_{low} se actualizan, mientras que la matriz de pesos original W permanece fija. Esto permite que el modelo se adapte a nuevos datos o tareas con un costo computacional mucho menor, manteniendo la mayor parte del conocimiento previamente adquirido durante el preentrenamiento.

Amini et al. delinean una metodología donde las matrices de bajo rango son utilizadas para adaptar redes neuronales profundas, mostrando que esta técnica puede lograr un rendimiento comparable al de un ajuste fino completo con una fracción del costo computacional y de memoria [7]. Esta característica es esencial para nuestra tarea, donde el corpus de datos en Kichwa es limitado, y la eficiencia computacional es imperativa para lograr una adaptación exitosa.

La aplicación de LoRAs en el contexto de la adaptación de modelos para el procesamiento del Kichwa se alínea con la meta de proporcionar herramientas de procesamiento de lenguaje natural robustas y eficaces que puedan servir a comunidades lingüísticamente desatendidas, manteniendo un balance entre rendimiento, eficacia y eficiencia computacional. Con la finalidad de proporcionar un marco sólido para la adaptación del modelo LlaMa 2 al Kichwa, las LoRAs se presentan como una herramienta técnica robusta y eficiente. Esta sección ha proporcionado una visión detallada del mecanismo operativo de las LoRAs, estableciendo un fundamento teórico que será instrumental en las etapas subsecuentes de desarrollo y evaluación de nuestro modelo adaptado, URKU.

Aprendizaje Reforzado en NLP

El Aprendizaje Reforzado (AR) se ha establecido como una piedra angular en la evolución de los sistemas autónomos y agentes inteligentes, proporcionando un marco donde los modelos aprenden óptimas estrategias de decisión mediante la interacción con su entorno. En el contexto del NLP, el AR introduce un paradigma que permite a los modelos aprender de manera iterativa a partir de las señales de retroalimentación recibidas sobre su desempeño en tareas lingüísticas específicas. La esencia del AR en NLP se centra en la formulación de tareas de procesamiento del lenguaje como problemas de decisión secuencial, donde un agente, en este caso el modelo de lenguaje, interactúa con un entorno, que podría ser un corpus lingüístico o un sistema de diálogo, para generar respuestas o traducciones. Este proceso se articula a través de episodios donde el modelo recibe una recompensa (o penalización) basada en la calidad de sus decisiones, es decir, la precisión y coherencia de las respuestas o traducciones proporcionadas. En el corazón de este enfoque se encuentra la función de recompensa, que guía al modelo hacia la generación de respuestas lingüísticamente adecuadas y contextualmente relevantes. La función de recompensa es crucial para el AR en NLP, ya que cuantifica la calidad de las

soluciones proporcionadas por el modelo, y por ende, su eficacia en la tarea en cuestión. Uno de los marcos más prominentes en AR para NLP es el Aprendizaje por Refurezo Inverso (IRL, por sus siglas en inglés), que permite estimar las funciones de recompensa óptimas a partir de ejemplos de comportamiento deseado, proporcionando así un camino para entrenar modelos en tareas como la generación de texto y la traducción automática. Este enfoque ha sido particularmente útil en escenarios donde las recompensas explícitas son difíciles de obtener. Así también, encontramos un marco de implementación alternativo que atiende mejor a nuestras necesidades de entrenamiento: el Aprendizaje por Refuerzo Basado en Retroalimentación Humana (RLHF, por sus siglas en inglés). El RLHF es un enfoque de aprendizaje por refuerzo que integra la retroalimentación humana directa en el ciclo de entrenamiento, permitiendo que el modelo se alinee más estrechamente con las expectativas y normas lingüísticas de los hablantes nativos. Este enfoque se sustenta en el trabajo seminal de Sutton y Barto, que establece las bases del AR, y se articula a través de una interacción continua con hablantes nativos, cuyas evaluaciones sirven como señales de recompensa para afinar el modelo en cada iteración [8].

Para futuros avances en el desarrollo de URKU a partir de nuestra propuesta metodológica, recomendamos una estrategia que incorpore RLHF, aprovechando la experiencia y conocimientos de hablantes nativos de Kichwa. Esta extensión implica que URKU, ya adaptado con LoRA para Kichwa, entre en un ciclo iterativo de RLHF. En él, los hablantes nativos desempeñarán un papel clave evaluando y proporcionando retroalimentación directa sobre las respuestas generadas por el modelo. Durante la fase de ajuste del modelo con RLHF, empleamos un marco matemático para actualizar los parámetros del modelo θ en función de la retroalimentación humana. La función de recompensa R se calcula utilizando la retroalimentación de los hablantes nativos y se integra en el proceso de aprendizaje del modelo. La actualización de los parámetros se realiza mediante el algoritmo de optimización de

políticas, donde la política $\pi(a|s;\theta)$ representa la probabilidad de que el modelo elija la acción a dado el estado s y los parámetros θ. La actualización de los parámetros se puede describir mediante la siguiente fórmula de gradiente de política:

$$\Delta\theta = \alpha \cdot \nabla_\theta \log \pi(a|s;\theta) \cdot R(s,a)$$

Donde:

- $\Delta\theta$ es el cambio aplicado a los parámetros del modelo.
- α es la tasa de aprendizaje.
- ∇_θ denota el gradiente con respecto a los parámetros θ.
- $\log \pi(a|s;\theta)$ es el logaritmo de la probabilidad de la política.
- $R(s,a)$ es la recompensa asignada por la acción a en el estado s.

Para la implementación de RLHF, la recompensa $R(s,a)$ se deriva de la retroalimentación humana. Por ejemplo, si un hablante nativo evalúa una traducción generada por el modelo como precisa y culturalmente adecuada, la recompensa sería positiva. En cambio, si la traducción es inexacta o inapropiada, la recompensa sería negativa o incluso podría incluir una penalización. Además, para asegurar que el modelo no solo se ajuste a ejemplos individuales sino que generalice a través de toda la distribución de tareas, se puede emplear una función de valor $V(s;\theta)$ para estimar el valor a largo plazo de los estados, y así actualizar la política en dirección a acciones con mejores recompensas futuras:

$$\Delta\theta = \alpha \cdot \nabla_\theta \log \pi(a|s;\theta) \cdot (R(s,a) + \gamma \cdot V(s';\theta) - V(s;\theta))$$

Donde:

- s' es el estado siguiente después de tomar la acción a.
- γ es el factor de descuento que equilibra la importancia de las recompensas inmediatas versus futuras.

Este enfoque matemático permitirá que el modelo URKU se ajuste de manera efectiva y eficiente, asegurando que la adaptación esté fundamentada en la retroalimentación cualitativa de los usuarios y en la optimización cuantitativa de los parámetros del modelo.

Iniciativas Globales: "No Language Left Behind", Meta; "Aya", Cohere

En el panorama global, emergen dos iniciativas notables que delinean la frontera entre la inclusión lingüística y la avanzada tecnológica del NLP; nos referimos a "No Language Left Behind" (NLLB) de Meta y "Aya" de Cohere. Estas propuestas representan sendos esfuerzos para cerrar la brecha lingüística, especialmente en el ámbito de las lenguas de pocos registros, y ofrecen un marco referencial invaluable para nuestro proyecto en curso.

La iniciativa "No Language Left Behind" de Meta emerge como un proyecto pionero en la esfera de la Inteligencia Artificial (IA), con el propósito de presionar el desarrollo de modelos de código abierto, públicamente disponibles, que faciliten traducciones evaluadas de alta calidad entre 200 idiomas, abarcando lenguas con recursos limitados como el Asturiano, Luganda y Urdu. La meta principal radica en posibilitar que las personas accedan y compartan contenido web en su idioma nativo, así como interactuar con cualquier individuo alrededor del globo, sin importar las preferencias lingüísticas [9]. Este emprendimiento se alínea con nuestra aspiración de fortalecer la interacción en Kichwa, mediante la adaptación y afinación de LlaMa 2, una acción que resuena con el espíritu inclusivo y global de NLLB.

Por otro lado, el proyecto "Aya" de Cohere se erige como una iniciativa de ciencia abierta, con la ambición de construir un modelo generativo multilingüe de vanguardia, bajo el auspicio de la sabiduría colectiva y contribuciones globales. Cohere For AI, el laboratorio de investigación comprometido con la resolución de desafíos intrincados en machine learning, desvela a Aya como un esfuerzo colaborativo en desarrollo, que invita a la construcción de un modelo de lenguaje multilingüe mediante la sintonización de instrucciones. Este proyecto de ciencia

abierta de un año congrega a expertos en IA de diversos sectores como la academia, la industria, organizaciones sin ánimo de lucro e investigadores independientes, con el propósito de edificar un modelo multilingüe de última generación y fomentar la colaboración abierta. Aya aspira a mejorar los modelos generativos multilingües existentes y acelerar el progreso en lenguas de diversas latitudes. Además, busca democratizar el acceso a la tecnología del lenguaje, permitiendo a cualquier entusiasta en el campo del NLP contribuir al proyecto. Todos los modelos, datos de entrenamiento y herramientas de recolección de datos serán 'open-source', resonando con la naturaleza inclusiva y colaborativa de este proyecto [10].

La alineación de estas iniciativas globales con nuestro proyecto se revela ineludible. Ambas resaltan un compromiso palpable hacia la inclusión lingüística y la preservación cultural, mediado por la utilización de tecnologías avanzadas de NLP. Al contextualizar los esfuerzos de modelado del lenguaje en una diversidad lingüística global, estas iniciativas no solo propulsan el estado del arte del NLP, sino que también se avocan a abordar temas de equidad y acceso en la era digital. Este interludio global refuerza la pertinencia y la necesidad de nuestro proyecto, situando la adaptación de LlaMa 2 para el Kichwa Ecuatoriano como un eslabón promisorio en esta cadena global de esfuerzos para la preservación lingüística y la inclusión digital.

Un Marco de Referencia: GPT Builder, por OpenAI.

GPT Builder, una nueva característica de OpenAI, permite la creación de modelos GPT personalizados para tareas específicas, como el aprendizaje de idiomas, sin requerir conocimientos de programación. Funciona combinando instrucciones, conocimientos y capacidades proporcionadas por el usuario para crear un modelo adecuado para un propósito específico [11]. Esta herramienta es particularmente útil para proyectos de idiomas, ya que ofrece un enfoque fácil de usar para la personalización de modelos, lo que la hace un marco de

comparación ideal para el proyecto de afinamiento del idioma Kichwa utilizando Adaptación de Rango Bajo en LLaMA 2. La capacidad del GPT Builder para integrar datos y funciones específicos del idioma podría proporcionar información valiosa cuando se compara con métodos tradicionales de afinamiento. Por lo mismo, se ha decidido incluirla en la evaluación final de los modelos entrenados debido a su sencilla implementación y poderosos resultados.

DESARROLLO DE LA PROPUESTA

Elección del Modelo de Lenguaje LlaMA 2

La selección de LlaMA 2 como modelo de lenguaje para el procesamiento y enseñanza del kichwa se justifica por su sofisticado tokenizador y su arquitectura Transformer optimizada. El tokenizador de LlaMA 2 utiliza un algoritmo de Codificación de Pares de Bytes (BPE), que descompone las palabras en subunidades denominadas tokens de manera eficiente. Esta técnica es especialmente beneficiosa para idiomas que presentan palabras no registradas previamente en el corpus de entrenamiento, como ocurre con el kichwa. La ventaja distintiva de este método es su capacidad para reconstruir significados a partir de los tokens más pequeños, obtenidos tras dividir las palabras en sus caracteres individuales (o bytes).

El funcionamiento del tokenizador se basa en la unión iterativa de los pares de bytes que aparecen con mayor frecuencia en el conjunto de datos. Este proceso comienza con la creación de un vocabulario inicial a partir del texto de entrenamiento y el cálculo de la frecuencia de todos los pares de bytes adyacentes. El par de bytes más común se identifica y se sustituye por un token inexistente en el texto, que luego se incorpora al vocabulario. Posteriormente, se actualiza el texto reemplazando todas las instancias del par de bytes más frecuente con el nuevo token. Este procedimiento se repite hasta que se agotan los pares frecuentes o se alcanza el tamaño de vocabulario deseado [4]. El resultado es un conjunto de vocabulario y reglas de

fusión que se emplearán para tokenizar cualquier texto nuevo. Parece importante resaltar que este tokenizador es especialmente útil en el presente caso de uso dado que el Kichwa es un idioma escrito en base al alfabeto romano. De otra manera, haría falta hacer una implementación de bajo nivel para posibilitar la tokenización del corpus. Este es el caso de idiomas como el chino, japonés, árabe, ruso, etc.

Con objetivos demostrativos, se presenta el pseudocódigo del algoritmo BPE implementado por LlaMA 2:

```
function BytePairEncoding(text, vocab_size)
    vocab = get_initial_vocab(text)
    token_freqs = count_token_freqs(text, vocab)
    while size(vocab) < vocab_size
        most_freq_pair = find_most_freq_pair(token_freqs)
        if most_freq_pair is None
            break
        new_token = merge_pair(most_freq_pair)
        vocab.add(new_token)
        text = replace_pair_in_text(text, most_freq_pair, new_token)
        update_token_freqs(token_freqs, most_freq_pair, new_token)
    return text, vocab
```

Este proceso iterativo de BPE permite que LlaMA 2 maneje eficientemente el vocabulario del kichwa, incluso cuando se encuentran palabras fuera del conjunto inicial de entrenamiento. La implementación práctica de este tokenizador se realiza mediante la biblioteca SentencePiece, como se muestra en el siguiente fragmento de código en Python:

```
import sentencepiece as spm

# Inicialización del tokenizador BPE con SentencePiece
spm.SentencePieceTrainer.train(input='kichwaTexts.txt',
  model_prefix='kichwa_bpe', #Exemplification
  vocab_size=32000, model_type='bpe')
```

La arquitectura Transformer de LlaMA 2 es altamente eficiente en la captura de dependencias de largo alcance y procesamiento paralelo de secuencias de texto, lo que es crucial para aprender la estructura gramatical y sintáctica del kichwa. La adaptabilidad inherente de LlaMA

2 lo hace ideal para idiomas menos representados, permitiendo una rápida adaptación a nuevos dominios y vocabularios.

La Utilización de Técnicas de LoRA

Haciendo eco de una explicación previa, la implementación de LoRA se puede describir matemáticamente como la inserción de matrices de bajo rango en las capas de atención y/o capas completamente conectadas del modelo Transformer.

La fórmula general para la adaptación de LoRA es:

$W' = W + BA$; donde BA es el equivalente a la matriz de bajo rango W_{low}

Ahora que se ha entendido el funcionamiento de BPE como tokenizador, es lógico notar que la generación de estas matrices de menor rango permite agregar únicamente el vocabulario nuevo resultante del corpus de entrenamiento sin tener que pasar antes por la totalidad dimensional del modelo pre entrenado base. Estas características aseguran que LlaMA 2 sea una opción prometedora para el proyecto de tesis, capaz de manejar las particularidades lingüísticas y los desafíos de representación que presenta el kichwa en el ámbito digital.

Lidiando con la Lingüística del Kichwa Ecuatoriano

Cuando el modelo se encuentra con una palabra en kichwa que no ha visto antes, como "champayana" [v. tener malestar], el tokenizador de BPE descompone la palabra en subunidades conocidas o tokens. El resultado, para el ejemplo, del proceso de tokenización, sería un vocabulario con los caracteres individuales ‘c’, ‘h’, ‘a’, ‘m’, ‘p’, ‘y’, ‘n’; BPE busca pares de caracteres adyacentes que aparecen con más frecuencia en el corpus de entrenamiento, si esta fuera palabra nueva no tendría pares pre existentes en el vocabulario; después, cuando el modelo procese más texto, empezará a combinar caracteres adyacentes según su aparición, lo que podría convertirse en ‘ch’, ‘pa’, ‘ya’, ‘na’, etc. En el caso de que ‘champayana’ sea una palabra que aparece con suficiente frecuencia, llegaría a convertirse en un token en sí misma,

aunque inicialmente sería representada por una secuencia de tokens más pequeños ‘ch’, ‘a’, ‘m’, ‘pa’, ‘ya’, ‘na’. Estos tokens son luego procesados por la arquitectura Transformer de LlaMA 2, donde las técnicas de LoRA entran en juego. La adaptación mediante LoRA se realiza en las capas de atención y capas completamente conectadas del modelo, donde las matrices de bajo rango B y A se utilizan para ajustar los pesos preexistentes W. El alcance de este reajuste depende de los parámetros de proyección del entrenamiento que se pasan al objeto ‘lora_config’ en que se almacenan los parámetros de entrenamiento. El siguiente ejemplo muestra todas las opciones disponibles para proyección a distintas capas de la arquitectura de LlaMA 2 que, sin embargo, para ser entrenados en su totalidad requerirían recursos computacionales significativamente más altos a los de la GPU A100 de 80GB disponible para la tarea presente:

```
from peft import LoraConfig

lora_config = LoraConfig(
    target_modules=["q_proj", "k_proj", "v_proj",
    "o_proj", "up_proj", "down_proj", "gate_proj"],
    # rank, kbit, alpha, beta, gamma, delta, etc...)
```

En la materialidad, solo se realizó un entrenamiento sobre los módulos objetivo q y k. Como resultado, obtenemos W' como la nueva matriz de pesos adaptada para comprender y procesar la nueva palabra. Por ejemplo, consideremos la entrada "Wamintsi tullpu waraka chinkaripashkami". El tokenizador de BPE descompone la frase en tokens, algunos de los cuales pueden ser nuevos para el modelo. A medida que estos tokens pasan a través de las capas de atención, las matrices de LoRA ajustan los pesos para adaptarse a la estructura gramatical y el significado contextual de la frase en kichwa. Esto permite que LlaMA 2 genere la salida correcta en español: "Se me ha perdido el pantalón de color rosado."

Al final, el trabajo conjunto del tokenizador de BPE y las técnicas de LoRA sobre LlaMA 2 proporciona una metodología robusta y eficiente para el procesamiento del idioma objetivo,

asegurando que el modelo pueda manejar eficazmente palabras y frases no vistas, y adaptarse continuamente a medida que se expone a más datos en este idioma.

DESARROLLO DEL PROTOTIPO

Proceso de Recolección de Datos

La construcción de un modelo de lenguaje robusto y preciso para el kichwa, una lengua con limitada representación digital, requiere una selección meticulosa de fuentes de datos confiables y representativas. En este contexto, se identificaron dos recursos primordiales que forman la columna vertebral de nuestra base de datos: el "Diccionario de la lengua kichwa del Ecuador" de J. C. Moya y el "Diccionario kichwa-castellano" publicado por el Ministerio de Educación del Ecuador [12][13]. Estos documentos, disponibles públicamente en línea, fueron escogidos por su autenticidad, exhaustividad y relevancia académica.

1. **J. C. Moya, "Diccionario de la lengua kichwa del Ecuador"**
    - **Ubicación y Acceso**: Este diccionario fue publicado por FLACSO en Quito, Ecuador, en 2012 y está disponible en formato digital en la biblioteca de FLACSO Andes. La accesibilidad en línea del documento facilitó su incorporación en nuestro corpus de entrenamiento.
    - **Contenido y Utilidad**: El diccionario de Moya es una obra bilingüe (español-kichwa y kichwa-español) que proporciona una amplia gama de vocabulario, incluyendo términos contemporáneos y, sobretodo, tradicionales. Su estructura y claridad en las definiciones, así como sus ejemplos de uso en oraciones acompañadas de su traducción, lo hacen un recurso invaluable para la comprensión y traducción precisa de términos, lo cual es esencial para la enseñanza y procesamiento automatizado del kichwa.

2. **Ministerio de Educación del Ecuador, "Diccionario kichwa-castellano"**
    - **Ubicación y Acceso**: Publicado en 2013 por el Ministerio de Educación del Ecuador, este diccionario está disponible en línea y es una fuente oficial para el estudio del kichwa unificado.
    - **Contenido y Utilidad**: Además de ser un diccionario bilingüe, este documento incluye un estudio introductorio sobre el kichwa, que ofrece una visión profunda de la estructura gramatical, variaciones dialectales y aspectos culturales del idioma. Esta sección se separó para ser utilizada en el entrenamiento de texto sin procesar, proporcionando al modelo una base sólida en la sintaxis y semántica del kichwa.

La utilización de estos diccionarios se justifica por ser algunos de los pocos compendios exhaustivos y accesibles digitalmente del kichwa. La búsqueda de fuentes se realizó mediante consultas en bases de datos académicas, recomendaciones de expertos en lingüística andina como Kuymi Tambaco, profesora de cosmovisión andina y Kichwa en la USFQ. La decisión de emplear estos recursos se basa en su autoridad lingüística y su capacidad para proporcionar un aprendizaje integral del idioma, desde su estructura básica hasta sus matices culturales.

Ahora bien, la tarea que nos llama no solo requiere una base de datos textual robusta, sino también un corpus que refleje el uso vivo y dinámico del idioma. En este sentido, el trabajo de campo antropológico realizado por Simeon Floyd, Ph.D. en Filosofía con especialización en Antropología de la Universidad de Texas en Austin, y su equipo, se convierte en un componente esencial de nuestra investigación. Para la realización de estas entrevistas se siguió un proceso metódico que partió del Proyecto Prometeo y su integración es descrita en la sección correspondiente al procesamiento de los datos.

El proyecto Prometeo, descrito por la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador, es una iniciativa que busca fortalecer la economía social del conocimiento en el país [14]. La participación del Dr. Floyd en este proyecto subraya la relevancia y el calibre académico de las entrevistas recopiladas, las cuales fueron realizadas bajo los auspicios de un programa que valora la investigación y la transferencia de conocimiento. El resultado del proyecto fue un corpus de audio, video y texto disponible para acceso público en el repositorio de lenguaje ELAN [15]. El Dr. Floyd, hablante fluido de kichwa y vinculado por lazos de trabajo social a la comunidad, utilizó su experiencia y conocimiento para facilitar conversaciones naturales y significativas. Las entrevistas abarcaron temas como la historia personal, la forma de vida y las experiencias de los entrevistados, proporcionando así un corpus rico y diverso. Este enfoque etnográfico asegura que el modelo de lenguaje no solo comprenda el kichwa desde una perspectiva lingüística, sino también desde su contexto cultural y social. El Dr. Floyd, junto con sus estudiantes, emprendió un meticuloso proceso de inmersión en las comunidades de Imbabura entre 2015 y 2016. Este esfuerzo resultó en 34 entrevistas que proporcionan una visión auténtica del uso cotidiano del kichwa. La recopilación de este material no fue una tarea trivial; implicó un proceso de acercamiento respetuoso y sensible hacia las comunidades indígenas. Se solicitaron permisos a las autoridades locales y se estableció un diálogo con los ancianos para garantizar una colaboración basada en el respeto mutuo y el entendimiento intercultural.

Las anotaciones de las entrevistas se realizaron en múltiples niveles, desde la transcripción fonética hasta la traducción detallada al español, incluyendo matices lingüísticos. Este proceso meticuloso de anotación, que tomó siete años, asegura que el modelo de lenguaje pueda capturar las complejidades del kichwa, incluyendo su naturaleza aglutinante y las variaciones dialectales. La inclusión de este corpus antropológico enriquece significativamente nuestro modelo de lenguaje. No solo proporciona ejemplos auténticos del uso del kichwa, sino que

también refleja las prácticas sociales y culturales que son inseparables del lenguaje. Este enfoque holístico es crucial para desarrollar tecnologías de procesamiento de lenguaje natural que sean verdaderamente inclusivas y representativas de la diversidad lingüística y cultural.

Procesamiento de Datos

En el ámbito de la lingüística computacional, la transmutación de datos brutos en un formato susceptible al escrutinio algorítmico constituye un paso crucial que sustenta la eficacia de los sistemas de procesamiento de lenguaje. Esta tesis delinea el meticuloso proceso de convertir los recursos existentes del idioma Kichwa—hasta ahora encapsulados dentro de los confines estáticos de documentos PDF—a un formato de diccionario JSON estructurado y dinámico. Esta transformación no es simplemente una exigencia técnica, sino una elección metodológica deliberada, diseñada para facilitar la ingestión sin obstáculos de datos por el modelo de lenguaje LlaMA 2. La discusión subsiguiente explica la lógica y el enfoque sistemático empleado para extraer, limpiar y reconstituir los datos textuales, asegurando su alineación con los requisitos arquitectónicos del modelo y optimizándolos para empeños computacionales subsecuentes.

En la fase inicial de transformación de datos, se utilizó la biblioteca **pdfminer** para extraer meticulosamente el contenido textual de los documentos PDF. Esta herramienta facilitó el análisis del texto, discerniendo la distribución espacial de los bloques de texto dentro del diseño del documento. Un aspecto crucial de esta extracción fue la bifurcación del texto en columnas izquierda y derecha, una tarea realizada evaluando las coordenadas x de los bloques de texto. Se determinó juiciosamente un valor umbral que sirvió como punto de demarcación entre las columnas, asegurando la fidelidad de la disposición espacial del texto en la transcripción digital. Posteriormente, los segmentos de texto de las columnas correspondientes se concatenaron para formar un flujo coherente de datos para cada página. Este enfoque metódico

no solo preservó la estructura original del documento, sino que también optimizó el texto para las etapas de procesamiento subsiguientes. Su implementación se puede entender de la siguiente manera:

```
Inicializar listas para columnas izquierda y derecha y texto de páginas
left_column = []
right_column = []
pages_text = []

Definir función extract_text_from_pdf(pdf_path)
    Para cada página en el documento PDF en pdf_path
        Inicializar listas temporales para texto de página actual
        left_page = []
        right_page = []

        Para cada elemento en el diseño de la página
            Si el elemento es un contenedor de texto
                Obtener texto del elemento

                Si la coordenada x del elemento < valor umbral
                    Añadir texto a left_page
                De lo contrario
                    Añadir texto a right_page

        Combinar texto de left_page y right_page
        Añadir texto combinado a pages_text
```

Una vez los PDFs han sido transformados en archivos de texto empieza la segmentación y limpieza del texto utilizando expresiones regulares (regex), una herramienta robusta para el reconocimiento de patrones dentro de cadenas de texto, para analizar meticulosamente el texto extraído. Este paso fue fundamental para identificar las entradas del diccionario, que a menudo comenzaban con una palabra seguida de sus variaciones fonéticas encerradas entre corchetes. Se diseñaron patrones regex para detectar estos marcadores de entrada, permitiendo el aislamiento de entradas individuales. Posterior a esta segmentación, abordamos el problema de la división de palabras con guiones, un artefacto común en textos procesados por OCR donde las palabras al final de una línea están partidas. Mediante regex, las palabras divididas por guiones se unieron sin problemas, asegurando la integridad léxica. Además, rectificamos las irregularidades en el espaciado, reemplazando secuencias de múltiples espacios por un único espacio para lograr uniformidad. Este proceso se refinó iterativamente, asegurando que cada

línea representara de manera precisa una entrada única y limpia. El resultado fue un texto optimizado, desprovisto de caracteres superfluos y anomalías de espaciado, sentando así una base prístina para la conversión subsiguiente al formato de diccionario JSON. Varios ejemplos de las expresiones regulares que se utilizaron son:

1. **r"\s*\w+\s*\[.*?\]" :** Para identificar las entradas del diccionario. Busca líneas que comiencen con una o más palabras seguidas de variaciones fonéticas entre corchetes. El **\s*** permite espacios en blanco antes y después de la palabra, mientras que **\w+** coincide con una o más letras o dígitos que constituyen la palabra, y **\[.*?\]** captura la variación fonética entre corchetes de manera no codiciosa.
2. **r" +" :** Reemplaza múltiples espacios consecutivos con un solo espacio. Esto es útil para normalizar el espaciado entre palabras y asegurar que no haya irregularidades en el texto debido a errores de OCR o de formato.
3. **r'[-]\s*$' :** Encuentra y elimina guiones al final de las líneas, lo que suele indicar que una palabra ha sido dividida entre dos líneas. Esto ayuda a mantener la integridad de las palabras que fueron incorrectamente separadas durante el proceso de digitalización.
4. **r'(\w+ \[\w+\])' :** Encuentra palabras seguidas de una variación fonética entre corchetes sin espacios iniciales, lo que indica el comienzo de una nueva entrada en el diccionario.
5. **r'\b([A-Z]+|\d+)\b' :** Busca secuencias de letras mayúsculas o números que están aislados como palabras completas, lo que puede ser útil para identificar encabezados o números de página que deben ser excluidos del texto del diccionario.
6. **r'([A-Z\s]+[.]-)' :** Detecta palabras en mayúsculas seguidas de un punto y un guión, lo que podría indicar la presencia de abreviaturas o términos especiales que necesitan ser tratados de manera diferente durante la segmentación del texto.

7. **r'=' :** Identifica líneas que contienen el símbolo igual (=). Al encontrar líneas con '=', el script puede separarlas en un archivo diferente, facilitando así la distinción entre las entradas del diccionario y otro tipo de contenido que requiere un tratamiento especial.

La conversión de datos a un formato adecuado es un paso crítico en NLP. Entre los formatos comunes como JSON, SQuAD y CSV, cada uno tiene sus ventajas dependiendo de la tarea específica a realizar. JSON, con su estructura de pares clave-valor y objetos anidados, es ideal para tareas como la clasificación de texto y el análisis de sentimientos, donde la estructura jerárquica y la legibilidad humana son beneficiosas. Por otro lado, SQuAD es preferible para el entrenamiento en tareas de respuesta a preguntas, ya que su formato está diseñado para asociar preguntas con pasajes de texto y respuestas localizadas. En contraste, CSV, con su formato tabular simple, es útil para tareas como la generación de texto y la traducción, donde las parejas de entrada y salida se alinean de manera clara y concisa [16].

En el contexto de nuestro proyecto, el formato JSON se destaca como la opción más adecuada debido a varias razones. Primero, la estructura de JSON facilita la representación de diccionarios lingüísticos de manera estructurada, esencial para la manipulación y acceso eficiente durante el entrenamiento. Segundo, la capacidad de JSON para manejar objetos anidados permite una representación más rica y detallada de la información lingüística, lo que es crucial para capturar las complejidades del Kichwa. Además, la naturaleza flexible de JSON permite una fácil expansión o modificación del esquema de datos, lo que puede ser necesario a medida que el modelo aprende y se adapta a las particularidades del idioma.

Comparativamente, aunque SQuAD podría ser útil para entrenar modelos en tareas de comprensión lectora y respuesta a preguntas, su estructura específica no se alinea con las necesidades de nuestro proyecto, que requiere una representación más general de los datos lingüísticos. Del mismo modo, mientras que CSV es excelente para alinear grandes volúmenes de texto paralelo, carece de la capacidad de representar la estructura jerárquica y las relaciones

complejas entre los datos, que son fundamentales para el entrenamiento. En conclusión, el formato JSON se alinea mejor con los objetivos presentes, proporcionando la combinación óptima de estructura, flexibilidad y accesibilidad, lo que resulta en un proceso de entrenamiento más eficiente y efectivo para capturar las sutilezas del Kichwa.

En la búsqueda de precisión y fiabilidad dentro de nuestro conjunto de datos, los ajustes manuales fueron indispensables, sirviendo como complemento a nuestros procedimientos automatizados de procesamiento de texto. Esta fase meticulosa estuvo regida por una estrategia de doble propósito: rectificar anomalías no detectadas por los scripts basados en expresiones regulares y mejorar la coherencia semántica de las entradas. Cada línea del texto extraído fue examinada minuciosamente, con especial atención a los patrones lingüísticos inherentes al Kichwa, que las herramientas automatizadas podrían pasar por alto o interpretar erróneamente, así como a los datos cuya presencia podía afectar la calidad del entrenamiento, sean estos números de página pasados por alto o el nombre de secciones presentes en los datos iniciales. Este nivel de aseguramiento de la calidad, detallado y granular, fue primordial, no solo para mantener la integridad de los datos sino también para asegurar su fidelidad al idioma. El resultado de este proceso fue un conjunto de datos pulido hasta alcanzar un estándar que satisface las demandas de la presente investigación, preparado para facilitar la comprensión y generación matizada del Kichwa por modelos de lenguaje.

Entrenamiento del Modelo

Las técnicas de LoRA se pueden implementar usando la librería **peft**, siglas en inglés para Ajuste Fino Eficiente en Parámetros. PEFT engloba una serie de metodologías destinadas a adaptar modelos de aprendizaje automático preentrenados a tareas o dominios específicos, minimizando la cantidad de parámetros entrenables. Al ajustar únicamente un pequeño subconjunto de parámetros del modelo, PEFT conserva recursos computacionales, acelera los

procesos de entrenamiento y facilita la implementación rápida de modelos. Este enfoque no solo reduce el riesgo de sobreajuste, dado el espacio reducido de parámetros, sino que también representa una práctica sostenible al disminuir la huella de carbono asociada con el entrenamiento extensivo de modelos. Las técnicas de PEFT se pueden categorizar ampliamente en métodos que adaptan pesos, inyectan módulos adaptables o utilizan enfoques centrados en datos. Entre estos, la Adaptación de Rango Bajo (LoRA) destaca debido a su enfoque único de descomponer matrices de peso en componentes de bajo rango, permitiendo un ajuste fino del modelo mínimo pero efectivo. Las ventajas de LoRA son múltiples: mantiene la arquitectura original del modelo, preservando así sus representaciones aprendidas; requiere la sintonización de menos parámetros, lo que agiliza el proceso de entrenamiento; y apoya la adaptación rápida a nuevos idiomas o tareas, lo que es crítico para aplicaciones de NLP. Otras técnicas de PEFT incluyen adaptadores, donde se insertan pequeños módulos entrenables entre las capas del modelo, y la sintonización de prompts, que implica aprender un conjunto de prompts suaves para guiar los mecanismos de atención. Cada técnica ofrece beneficios distintivos, pero LoRA es particularmente adecuada para tareas donde el equilibrio entre rendimiento y agilidad es crítico.

Sobresimplificando, con LoRA trabajando sobre las capas de atención de arquitecturas de transformadores, y LlaMA 2 como un modelo con capas de atención, es lógico concluir la posibilidad de usar la técnica descrita sobre el modelo seleccionado. Para optimizar el proceso fue útil basar el entrenamiento en el proyecto de ***text-generation-webui*** que ha desarrollado una interfaz de web intuitiva y funcional en los procesos de prompting y entrenamiento [17]. Aún así, es preciso entender el proceso de entrenamiento para asegurar que la implementación es coherente y cumple con los estándares de afinamiento. El entrenamiento del modelo para LoRA es un procedimiento sofisticado que involucra varias etapas. Comienza con el cálculo de parámetros entrenables, donde la función ***calc_trainable_parameters*** determina el número

total de parámetros en el modelo que están sujetos a optimización durante el entrenamiento, así como el número total de parámetros presentes. Esta distinción es esencial para comprender la capacidad del modelo y el alcance de su adaptabilidad a través del entrenamiento. La codificación del texto se maneja mediante la función ***encode***, que asegura que el texto esté correctamente tokenizado dentro de una restricción de longitud definida y que el token de inicio de oración (BOS) se maneje correctamente. La tokenización es un paso crítico ya que convierte el texto en bruto en un formato que el modelo puede entender. Se introduce una mayor sofisticación con la función ***tokenize***, la cual prepara la entrada para el entrenamiento. Aplica selectivamente la tokenización a diferentes segmentos de la entrada basándose en condiciones específicas de entrenamiento, diferenciando entre los tokens que serán entrenados y aquellos que serán ignorados por el modelo durante la fase de entrenamiento. También maneja la colocación de tokens de fin de oración (EOS), asegurando que las secuencias terminen correctamente y gestiona el relleno para mantener longitudes de secuencia consistentes.

En la preparación del conjunto de datos, el proceso de entrenamiento implica cargar y procesar archivos de texto en bruto y archivos JSON, aplicando un enfoque diferenciado para cada tipo de formato. Mientras que para los archivos de texto en bruto se aplica el truncamiento y la segmentación apropiados para manejar eficientemente conjuntos de datos grandes, los archivos JSON requieren un procesamiento que atienda a su estructura inherente—en función de un formato pre definido—, extrayendo y convirtiendo la información relevante en un formato compatible con el modelo. Se presta especial atención a la gestión de memoria eliminando tanto el texto en bruto como los datos JSON de la memoria después de que sus versiones tokenizadas están preparadas. Este procedimiento dual asegura que se mantenga la integridad de los datos estructurados y no estructurados durante el entrenamiento. El formateo de los datos implica crear indicaciones que el modelo utilizará durante el entrenamiento, lo cual es un paso crítico para moldear la comprensión y las capacidades de generación del modelo. Las funciones

generate_prompt y ***generate_and_tokenize_prompt*** ilustran el mecanismo intrincado de convertir puntos de datos estructurados en indicaciones de lenguaje natural de las que el modelo puede aprender.

Una vez que el conjunto de datos está preparado y las indicaciones están listas, el entorno de entrenamiento se configura con un objeto ***Trainer*** de la biblioteca transformers. Este objeto orquesta el proceso de entrenamiento, gestionando tamaños de lotes, acumulación de gradientes, tasas de aprendizaje y métricas de evaluación. Cabe destacar que el proceso se mejora con callbacks personalizados, los cuales proporcionan enganches en el bucle de entrenamiento, permitiendo un monitoreo detallado y control sobre la progresión del entrenamiento. El ***lora_model*** es una encarnación del modelo mejorado con LoRA, configurado con adaptaciones específicas a la arquitectura del modelo que están dirigidas a aprender de manera eficiente a partir de una cantidad limitada de datos de entrenamiento. Esta configuración se ajusta a través del objeto ***LoraConfig***, que especifica el rango y otros hiperparámetros relevantes para la técnica LoRA. A lo largo del entrenamiento, los parámetros del modelo se cuantifican para gestionar la memoria y el cálculo de manera más eficiente, especialmente relevante para hardware con recursos limitados o para lograr tiempos de entrenamiento más rápidos. En resumen, el proceso de entrenamiento encapsula la conversión de texto en bruto en un formato estructurado que el modelo puede interpretar, un eficiente canal de preparación de conjuntos de datos, una configuración detallada del procedimiento de entrenamiento utilizando LoRA y un meticuloso manejo de memoria y procesos. El enfoque refleja un equilibrio entre la profundidad del entrenamiento del modelo y las limitaciones prácticas de los recursos computacionales.

Descripción del Modelo Final 'URKU'

El modelo 'URKU' emerge como un resultado interesante en el panorama de los modelos de lenguaje, configurado meticulosamente para capturar la esencia del Kichwa. Su formación se ha basado en un tríptico de conjuntos de datos —'AllData', proporcionando una comprensión integral del idioma; 'MoreDetail', que refina esta visión con entradas de alta calidad que realzan la precisión contextual; y 'Grammar_guidelines', que instila al modelo las estructuras gramaticales fundamentales del Kichwa. Cada fuente de datos ha sido instrumental para dotar a URKU de una perspectiva única, desde la comprensión teórica hasta el reconocimiento de matices complejos y uso práctico del idioma.

Modelo	Training Runtime (minutes)	Dataset	Entries in the dataset	Alpha	Rank	Projections	Learning Rate	Steps	Loss	SISA Benchmark*
Llama 2 7B	100	MoreDetail	4337	4096	2048	q, v, k, o	1,60E-05	3051	2,90E+14	3,3%
Llama 2 7B	130	MoreDetail	4337	4096	2048	q, v, k, o	1,60E-05	5043	2,90E+12	17,8%
Llama 2 7B	240	AllData	14175	2048	1024	q, v	2,00E-04	7017	6,50E+00	0,6%
Llama 2 70B	90	Grammar_guidelines	2112	512	256	q, v,k	9,10E-05	191	2,90E+00	11,7%
Llama 2 70B	90	Grammar_guidelines	2112	2048	1024	q, v	1,80E-04	431	2,40E+00	0,0%
Llama 2 70B	612	MoreDetail	4337	2048	1024	q, v	1,70E-04	1863	4,70E-01	6,1%
Llama 2 70B	**1031**	**MoreDetail**	**4337**	**2048**	**1024**	**q, v**	**1,10E-04**	**2503**	**3,90E-01**	**86,7%**
Llama 2 70B	1560	MoreDetail	4337	2048	1024	q, v	1,00E-05	3727	3,60E-01	66,7%
Llama 2 70B	1920	AllData	14175	1024	512	q, v	1,70E-06	19235	1,80E-01	77,8%
Llama 2 70B	2102	AllData	14175	2048	1024	q, v	3,10E-05	19235	8,60E+00	0,0%
GPT Builder	**6**	**AllData**	**14175**	**n/a**	**n/a**	**n/a**	**n/a**	**n/a**	**n/a**	**95,0%**

Tabla 1. Comparativa de Configuraciones y Rendimiento de Modelos LLaMA 2 y GPT Builder entrenados sobre una GPU A100/80GB.

El modelo 'URKU', especializado en Kichwa, se ha forjado a través de un proceso de optimización que considera meticulosamente la interacción de los hiperparámetros 'Alpha' y 'Rank'. La evaluación de diversas configuraciones ha revelado que una afinación más conservadora de estos parámetros suele resultar en un rendimiento superior en el benchmark

SISA, sugiriendo que la efectividad no radica necesariamente en la maximización de los parámetros sino en encontrar un equilibrio que se adecue a la especificidad de las tareas lingüísticas del Kichwa.

El benchmark SISA, cuyo nombre significa 'flor' en Kichwa, es un modelo de evaluación integral diseñado en colaboración con Kuymi Tambaco, lingüista experta en kichwa ecuatoriano. Este benchmark ha sido meticulosamente desarrollado para medir la competencia del modelo en áreas críticas como la conjugación de verbos, la relación de conceptos, la identificación de morfemas, así como la generación de texto y la capacidad de responder preguntas de manera coherente. SISA no solo examina la precisión lingüística de 'URKU', sino que también pone a prueba su habilidad para utilizar el Kichwa de manera práctica y completa, reflejando las capacidades que se esperarían de un hablante competente del idioma. El modelo de evaluación consta de 180 preguntas sobre las que se calcula el porcentaje de éxito según la cantidad de preguntas que han sido correctamente respondidas a criterio del evaluador.

El análisis detallado del entrenamiento y la adaptación de URKU, utilizando conjuntos de datos variados, ha arrojado luz sobre la importancia de la calidad del contenido sobre la cantidad. Configuraciones que utilizan 'MoreDetail' superaron aquellas con 'AllData' en términos de rendimiento en SISA, destacando que un corpus curado puede ser más beneficioso que uno más amplio pero potencialmente menos focalizado. Esto refleja la posibilidad de que un conjunto más grande de datos pueda introducir ruido o propiciar sobreajuste, en detrimento de la capacidad de generalización del modelo.

La tasa de aprendizaje y el número de pasos también han demostrado ser factores críticos. Tasas de aprendizaje más bajas están asociadas con menores pérdidas, indicando un entrenamiento más estable. Además, la cantidad de pasos de entrenamiento no es un indicador directo del éxito; es la sinergia entre la duración del entrenamiento y la configuración de los hiperparámetros lo que define la calidad del modelo resultante.

Contrastando con GPT Builder, que no se adaptó mediante LoRA pero alcanzó un 95% en SISA, URKU ilustra que las estrategias de entrenamiento específicas al contexto del idioma pueden llegar a resultados importantes en una infraestructura local e independiente. Mientras GPT Builder muestra robustez general, este depende del acceso permitido por OpenAI para su creación y uso. De modo que existe una relación de poder y falta de acceso cuando se trata de entender qué es lo que realmente está ocurriendo cuando un GPT de dominio específico es generado desde la interfaz de ChatGPT. En este contexto de dominio digital frente a alternativas de uso libre sobre hardware de usuario, URKU resalta la efectividad de su enfoque como un posible método hacia proyectos más grandes y flexibles.

En resumen, 'URKU' encarna un modelo altamente competente en el procesamiento del Kichwa, avanzando significativamente en la inclusión lingüística y la preservación cultural. La adaptación precisa de URKU, con hiperparámetros calibrados y datos seleccionados cuidadosamente, lo posiciona como un modelo pionero para el tratamiento de idiomas menos representados, asegurando que la riqueza de cada lengua tenga su espacio en la era digital.

CONCLUSIONES

El modelo 'URKU' representa un esfuerzo significativo en el ámbito del Procesamiento del Lenguaje Natural para atender a las necesidades lingüísticas de las comunidades que hablan Kichwa en Ecuador. Este trabajo no solo responde a la declinación en el número de hablantes nativos de Kichwa, reflejando así un compromiso con la preservación cultural y lingüística, sino que también aborda la exclusión digital de lenguajes minoritarios a nivel global. La colaboración con la lingüista Kuymi Tambaco para el desarrollo del benchmark SISA ha sido fundamental, permitiendo una evaluación práctica del modelo que va más allá de la teoría, probando su efectividad en la conjugación de verbos, la relación de conceptos y la generación de texto. Estas pruebas, esenciales para medir la competencia lingüística, han demostrado que 'URKU' puede funcionar como un hablante nativo competente del Kichwa.

El proceso de adaptación del modelo mediante técnicas de LoRA ha revelado la importancia de la calidad del corpus sobre la cantidad de datos, y la necesidad de un equilibrio entre los hiperparámetros para el rendimiento óptimo en tareas específicas. A pesar de los desafíos encontrados, como la inicial escasez de datos en Kichwa, 'URKU' ha mostrado ser una herramienta valiosa, no solo académicamente sino en aplicaciones prácticas que pueden influir positivamente en la inclusión social y la educación bilingüe.

La comparación con GPT Builder ha reafirmado la necesidad de modelos especializados de uso local para idiomas menos representados, con 'URKU' estableciendo un alto estándar de éxito en el benchmark SISA. Esto es, alternativas que puedan correr en hardware de usuario suponen un avance en la democratización de los recursos digitales modernos que brindan la esperanza —o vaga ilusión— de acceso independiente a las capacidades de los modelos de lenguaje de vanguardia. Así, permitiendo aplicaciones de alto impacto social que no estén sujetas a la disponibilidad de los grandes jugadores en la hegemonía de la inteligencia artificial.

Este estudio sienta un precedente para futuras investigaciones, alentando el desarrollo de corpus más extensos y diversificados para mejorar el rendimiento de este tipo de modelos. Además, insta a la reflexión sobre cómo la tecnología de inteligencia artificial debe ser modelada para reflejar y respetar la rica diversidad cultural y lingüística del mundo, asegurando que la herencia lingüística de Ecuador y de otras comunidades no solo sobreviva sino que prospere en la era digital.

REFERENCIAS BIBLIOGRÁFICAS

[1] Datos obtenidos del Censo Nacional. [En línea]. Disponible: https://www.censoecuador.gob.ec/data-y-resultados/. [Accedido: Oct. 05, 2023].

[2] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in Proceedings of the Eighteenth International Conference on Machine Learning, 2001.

[3] A. Vaswani et al., "Attention Is All You Need," in Advances in Neural Information Processing Systems 30, I. Guyon et al., Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [En línea]. Available: https://www.researchgate.net/figure/An-example-of-a-transformer-architecture-Vaswani-et-al-2017_fig4_355046085 [Accedido: Oct. 05, 2023].

[4] H. Touvron et al., "Llama 2: Open foundation and Fine-Tuned chat models," *arXiv (Cornell University)*, Jul. 2023, doi: 10.48550/arxiv.2307.09288. [En línea]. Disponible: https://arxiv.org/abs/2307.09288 [Accedido: Oct. 10, 2023].

[5] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018. [En línea]. Disponible: https://arxiv.org/abs/1810.04805 [Accedido: Oct. 15, 2023].

[6] OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023.

[7] A. R. F. Amini, M. Saberian, and H. Lakkaraju, "Low-Rank Adaptation of Deep Networks," in Proceedings of the 2021 AAAI Conference on Artificial Intelligence, 2021.

[8] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," MIT Press, 2018.

[9] A. Fan, A. Baevski, Y. Dauphin, and M. Auli, "No Language Left Behind: Scaling Human-Centered Machine Translation," Facebook AI Research, 2023. [En línea]. Disponible: No Language Left Behind: Scaling Human-Centered Machine Translation - Meta Research (facebook.com). [Accedido: Oct. 05, 2023].

[10] Aya, “About Aya,” 2023. [En línea]. Disponible: https://sites.google.com/cohere.com/aya-en/about-aya [Accedido: Oct. 05, 2023].

[11] OpenAI, "Introducing GPTs," 2023. [En línea]. Disponible: https://openai.com/blog/introducing-gpts [Accedido: Nov. 31, 2023].

[12] J. C. Moya, “Diccionario de la lengua kichwa del Ecuador,” Quito, Ecuador: FLACSO, 2012. [En línea]. Disponible: https://biblio.flacsoandes.edu.ec/libros/digital/55476.pdf

[13] Ministerio de Educación del Ecuador, “Diccionario kichwa-castellano,” Quito, Ecuador: Ministerio de Educación del Ecuador, 2013. [En línea]. Disponible: https://educacion.gob.ec/wp-content/uploads/downloads/2013/03/RK_diccionario_kichwa_castellano.pdf

[14] Secretaría de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador, “What is the Prometeo Project?”, en Prometeo, s.f. [En línea]. Disponible: prometeo.educacionsuperior.gob.ec. [Accedido: 1 nov. 2023].

[15] S. Floyd, "Imbabura Quechua," in Floyd Collection, The Language Archive, 2011-2013. [En línea]. Disponible: https://hdl.handle.net/1839/00-0000-0000-0022-42BA-3. [Accedido: Nov. 4, 2023].

[16] Lakera AI, "The Ultimate Guide to LLM Fine Tuning: Best Practices & Tools," Lakera AI Blog, [En línea]. Disponible: https://www.lakera.ai/blog/llm-fine-tuning-guide. [Accedido: 6-Nov-2023].

[17] Oobabooga, "text-generation-webui," GitHub repository, [En línea]. Disponible: https://github.com/oobabooga/text-generation-webui/tree/main. [Accedido: Nov. 8, 2023].

[18] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," in Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

[19] M. F. Porter, "An Algorithm for Suffix Stripping," Program, vol. 14, no. 3, pp. 130–137, 1980.

[20] C. D. Manning and H. Schütze, "Foundations of Statistical Natural Language Processing," MIT Press, 1999.

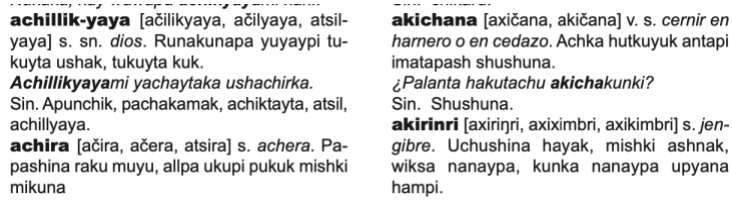
[21] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[22] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," OpenAI, 2018.

ANEXO A: DATAGENERATOR.IPYNB

Es importante entender que la naturaleza de los datos con los que se formó el corpus es caótica y compleja. Los datos se originan de fuentes oficiales en formato PDF, por lo que es necesario, en primer lugar, crear un archivo de texto manteniendo la mayor fidelidad posible frente a los datos originales. Esto solo es un problema debido a que la mayoría de los documentos estaban originalmente formateados en una distribución de dos columnas o, en su defecto, contaban con tablas imposibles de traducir a texto plano mediante métodos iterativos de programación. A pesar de esto, se creó un jupyter notebook llamado 'DataGenerator.ipynb' que me asistió en todo el proceso de automatizar cuando sea posible automatizar. Este anexo busca demostrar algunas de las funciones más importantes que se consiguieron que, además, fueron esenciales para alcanzar el corpus de datos final.

En el caso de los layouts de dos columnas, se optó por una solución arbitraria que depende de la posición de cada columna en un documento. Esto funcionó gracias a que todas las páginas de un documento en específico funcionaban sobre la misma distribución de espacio:



achillik-yaya [ačilikyaya, ačilyaya, atsilyaya] s. sn. *dios*. Runakunapa yuyaypi tukuyta ushak, tukuyta kuk.
***Achillikyaya**mi yachaytaka ushachirka.*
Sin. Apunchik, pachakamak, achiktayta, atsil, achillyaya.
achira [ačira, ačera, atsira] s. *achera*. Papashina raku muyu, allpa ukupi pukuk mishki mikuna

akichana [axičana, akičana] v. s. *cernir en harnero o en cedazo*. Achka hutkuyuk antapi imatapash shushuna.
*¿Palanta hakutachu **akicha**kunki?*
Sin. Shushuna.
akirinri [axiriŋri, axiximbri, axikimbri] s. *jengibre*. Uchushina hayak, mishki ashnak, wiksa nanaypa, kunka nanaypa upyana hampi.

Esta observación hizo posible codificar una función que lea las columnas una por una para luego unirlas cuando todo el texto de una misma página ha sido leído. Este acercamiento funcionó en su mayoría exceptuando ocasiones extraordinarias en las que algún factor inesperado cambiara el orden en el que el texto era leído. El problema con esto es que hacía falta revisar entrada por entrada que no existan irregularidades lo bastante graves como para afectar el entrenamiento. Parte de este proceso fue lidiar con expresiones regulares que aprovecharan el formato de cada documento y así poder procesarlo con algo de ayuda.

```python
from pdfminer.high_level import extract_pages
from pdfminer.layout import LTTextContainer, LTChar

# Initialize variables to hold text
left_column = []
right_column = []
pages_text = []  # Holds the combined text for each page

# Loop through each page
for page_layout in extract_pages("Data/Tests/sample.pdf"):
    # Temporary lists to hold the text for the current page's
    # left and right columns
    left_page = []
    right_page = []

    for element in page_layout:
        # Check if the element is a text container
        if isinstance(element, LTTextContainer):
            text = element.get_text()

            # Check the x-coordinate to determine the column
            # (You may need to adjust the threshold value)
            if element.bbox[0] < 300:
                left_page.append(text)
            else:
                right_page.append(text)

    # Combine text blocks for the current page and append to pages_text
    combined_page_text = ' '.join(left_page) + ' ' + ' '.join(right_page)
    pages_text.append(combined_page_text)

# Print or save the texts
for i, text in enumerate(pages_text):
    print(f"Page {i+1} Text:")
    print(text)
    print("="*40)

first_output_path = "Data/sample.pdf"

# Save to file
with open(first_output_path, "w") as text_file:
    for page_text in pages_text:
        text_file.write(page_text + '\n\n')
```

```
Page 1 Text:
A
 achachay  [ačačay, ačačaw, atsatsay] in-
terj. expre sión de frío. Yapa chiri kakpi rimay.
Wawakunaka achachay nishpa chayamun.
Sin. Chiri chiri.
achachaw  [ačačaw,  ačučuy]  interj.  amz.
expresión de calor. Yapa rupay tiyakpi rimay.
achachaw, mikunaka rupakmi kashka.
Sin. Araray, rupakuk.
achka  [ačka,  ačika,  aška]  adv.  bastante,
harto, mucho. Imatapash tawkata, tawka  ti-
yakta, mana ashallata rikuchik.
Chakramantaka achka saratami pallarkani.
Sin. Ashtaka,  hatunta, pachan, tawka, llas-
hak.
achik [ačix, ačig, ači] s. luz, claridad, claro.
Killamanta, intimanta, kuyllur manta llukshik
llipyarik; imatapash rikunkapak kak.
Tamya punchapika achikka mana rikurinchu.
achiklla [ačixlya] adj. lúcido, claro, nítido.
Killamanta, intimanta, kuyllur manta llukshik
ancha llipyarik; imatapash rikunkapaklla kak.
Tamya punchaka  mana achikllachu kan.
Sin. chuya.
...
Mushukta imatapash yachachina.
Mushuk man kata arikuni.
```

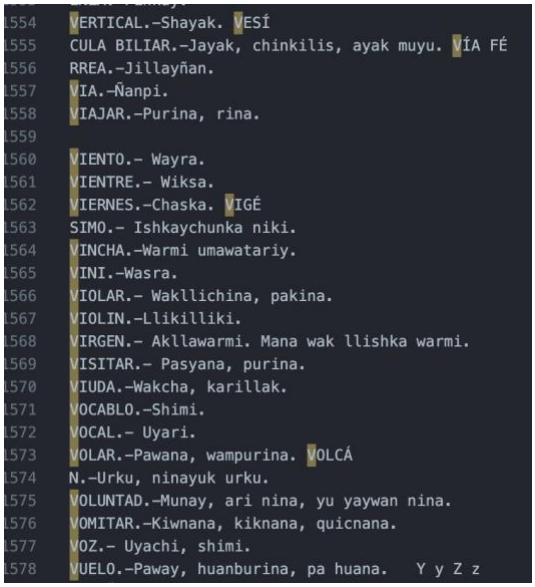
A partir de aquí –con el contenido en formato de texto–, para este caso específico buscamos los corchetes ‘[]’ para separar la palabra de sus pronunciaciones, traducciones y ejemplos de uso en oraciones.

```python
# Regular expression to match the desired format: "word [variation]"
pattern = re.compile(r'(\w+ \[[^\]]+\])')

for line in lines:
    # Find all matches of the pattern in the line
    matches = [(m.start(), m.group()) for m in pattern.finditer(line)]

    # If more than one match is found, split the line
    if len(matches) > 1:
        start_index = 0
        for start, match in matches:
            # Append the segment from the start of the line or the end of the last match, up to the start of this match
            processed_lines.append(line[start_index:start].strip() + '\n')
            start_index = start

        # Append the remaining part of the line
        processed_lines.append(line[start_index:].strip() + '\n')
    else:
        # If only one match is found, keep the line as is
        processed_lines.append(line)
```

A pesar de este paso, siempre existían secciones que escapaban al buscador de patrones. De esta manera, llegó a ser necesario manejar funciones extra para hacer una doble revisión:

```
def find_multiple_brackets_in_line(file_path):
    # Regular expression pattern to match '[...words.]' structures
    pattern = r'\b([A-Z]+|\d+)\b'

    # Initialize a list to store line numbers where multiple '[...words.]' occur
    line_numbers = []
```

La función de arriba se utilizó para limpiar las líneas del archivo de texto en donde había una cantidad irregular de corchetes. Así mismo, con otros textos, cada uno con su respectiva función auxiliar de formateo, persisten problemas similares. Por ejemplo, para un diccionario español – kichwa, varias palabras se cortaban e interrumpían el flujo correcto de formateo cuando se esperaba convertir el archivo de texto a un diccionario json. En estos casos, la búsqueda manual siempre terminaba siendo la mejor alternativa a los errores.



Aún así, cada archivo fue formateado de forma tal que exista un identificador en cada línea capaz de determinar el tipo de separación que se realizaría a los datos para poder separarlos como valores de una llave específica en el formato alpaca. Eso es, elegir el valor que correspondería a la “Entrada”, “Salida” y, de forma distinta y arbitraria, la “Instrucción”. En el caso del ejemplo, el identificador es el guión que está presente entre la palabra en español y la

palabra en Kichwa. Así, cuando se llama al método Split, esto se hace desde un ‘try’. De tal forma que si una línea no cumple el formato esperado sea posible identificarla para ir a solucionar el error.

```
            except Exception as e:
                print(f"Error on line {line_number}: {line}")
                print(f"Exception: {e}")

    return dataset

# Example usage:
file_path = 'Outputs/Clean txt/Final/cas_kich_F.txt'
dataset = parse_to_json_spanish_to_kichwa(file_path)

# Save the dataset as a JSON file
output_file_path = 'Outputs/JSON/Base_JSONs/cas_kich_BASE.json'
with open(output_file_path, 'w', encoding='utf-8') as f:
    json.dump(dataset, f, ensure_ascii=False, indent=4)
```

✓ 0.0s

```
Error on line 273: armazón (palos pe queños colocados dentro de  la olla para cocinar) s. panshi.
Exception: not enough values to unpack (expected 2, got 1)
Error on line 442: bajo de las piedras: kantina.
Exception: not enough values to unpack (expected 2, got 1)
Error on line 495: churu; caracol grande: chinanku.
Exception: not enough values to unpack (expected 2, got 1)
Error on line 696: casa: kanchapunku.
Exception: not enough values to unpack (expected 2, got 1)
Error on line 744: chusku patsak.
Exception: not enough values to unpack (expected 2, got 1)
Error on line 887: dañarse las cosas: putasyay.
```

Al final, cuando el documento no tiene errores, el objetivo es conseguir un archivo JSON que se vea así:


```
[
    {
        "Instrucción": "¿Qué palabra se asemeja a a cambio de en el idioma Kichwa?",
        "Entrada": "a cambio de",
        "Salida": "adv. ranti, rantimpa."
    },
    {
        "Instrucción": "¿Cuál es el sinónimo en Kichwa de a continuación?",
        "Entrada": "a continuación",
        "Salida": "adv. kipa; chaymanta."
    },
    {
        "Instrucción": "¿Cómo se traduciría a diario al Kichwa?",
        "Entrada": "a diario",
        "Salida": "adv. punchanta punchanta, punchantin."
    },
    {
        "Instrucción": "¿Cómo se enunciaría a gusto en el idioma Kichwa?",
        "Entrada": "a gusto",
        "Salida": "adv. ninantak."
    },
```

Donde la instrucción es un valor seleccionado de forma pseudo-aleatoria a partir de una lista de posibles opciones generada con ayuda de inteligencia artificial (ChatGPT) para expandir el horizonte de posibilidades brindando mayor representatividad a los datos e idealmente aumentando las posibilidades de inferencia funcional.

Este proceso de expansión de datos para las instrucciones pudo haber sido logrado haciendo uso del propio LlaMA 2 caso de estudio para este trabajo, mas se decidió optar por un modelo más robusto en su rendimiento sobre métricas de lenguaje e inferencia para asegurar que el entrenamiento sea resultado de una calidad de datos externa a sí mismo. Así, logramos evitar conversaciones —relevantes, aunque no indispensables— sobre el posible efecto que tendría para un modelo alimentarlo en base a su propio, determinado, comportamiento. Sino, conseguimos resultados que se ven beneficiados de las fortalezas de otros modelos de lenguaje. Esto es, si un modelo que se busca entrenar en datos que no ha visto antes junto a datos que sí estaban presentes originalmente, es importante minimizar el factor de predecibilidad. Al pedir a LlaMA 2 que genere las instrucciones con las que sería entrenado para la tarea de afinamiento, es probable que se cumplan determinados patrones que generan conflicto en el entrenamiento del LoRA al ser datos poco relevantes dada su presencia en el corpus original, sea esta presencia cuestión de estilo, estructura o forma. Por otro lado, al unir comportamientos, como sería usar los patrones presentes en ChatGPT con el entrenamiento del LlaMA 2, forzamos la representatividad de manera significativa.

Extiende la siguiente lista de variaciones con tantas alternativas útiles como puedas imaginar:

```
instruction_variations = [
  "¿Cómo se traduce la palabra española {word} al Kichwa?",
  "Dame la traducción en Kichwa de {word}.",
  "¿Qué significa {word} en Kichwa?",
  "Traduce {word} al Kichwa.",
  "¿Cómo traducirías el español {word} a Kichwa?"
]
```

Briefing

Certainly, expanding the list of instruction variations can help make your dataset richer and more versatile. Here's an extended list that incorporates a variety of phrasing and query types.

Extended List of Instruction Variations

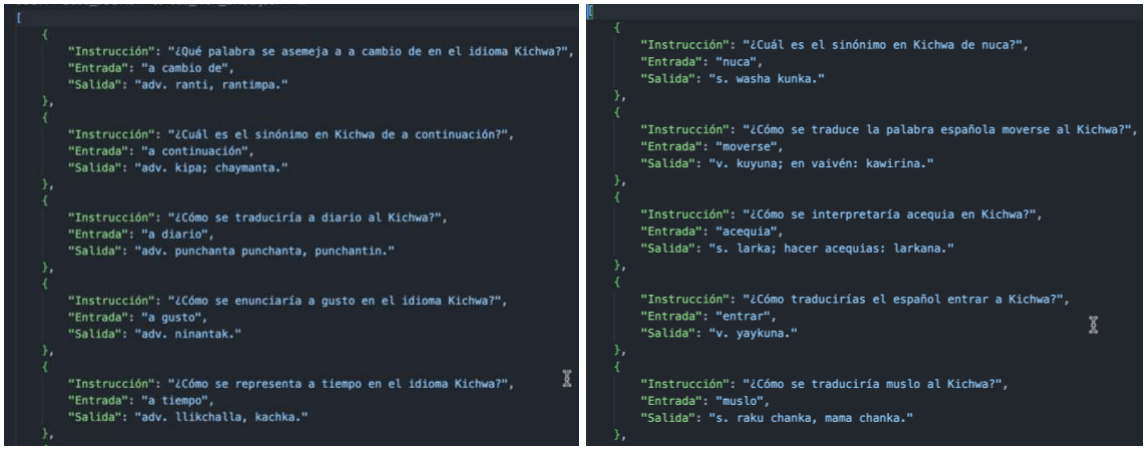
python Copy code

```
instruction_variations = [
    "¿Cómo se traduce la palabra española {word} al Kichwa?",
    "Dame la traducción en Kichwa de {word}.",
    "¿Qué significa {word} en Kichwa?",
    "Traduce {word} al Kichwa.",
    "¿Cómo traducirías el español {word} a Kichwa?",
    "¿Cuál es la palabra en Kichwa que corresponde a {word} en español?",
    "¿Cómo se dice {word} en Kichwa?",
    "Ofrece el equivalente en Kichwa de la palabra española {word}.",
    "¿Qué palabra en Kichwa se usa para {word}?",
```

Finalmente, a manera de optimizar el random state inicial de entrenamiento del modelo, los archivos JSON finales se guardan como archivos de tipo ‘BASE’ para posteriormente leerlos, aplicarles el método shuffle de la librería random y así evitar posibles problemas en el entrenamiento al tener los datos en un orden alfabético altamente determinístico:

```python
input_file = 'Outputs/JSON/KichwaTexts.json'
output_file = 'Outputs/JSON/KichwaTexts_SHUFFLED.json'

# Load the JSON data from the file
with open(input_file, 'r', encoding='utf-8') as f:
    dataset = json.load(f)

# Shuffle the dataset
random.shuffle(dataset)

# Save the shuffled dataset back to a JSON file
with open(output_file, 'w', encoding='utf-8') as f:
    json.dump(dataset, f, ensure_ascii=False, indent=4)
```

El cambio en el resultado es evidente:



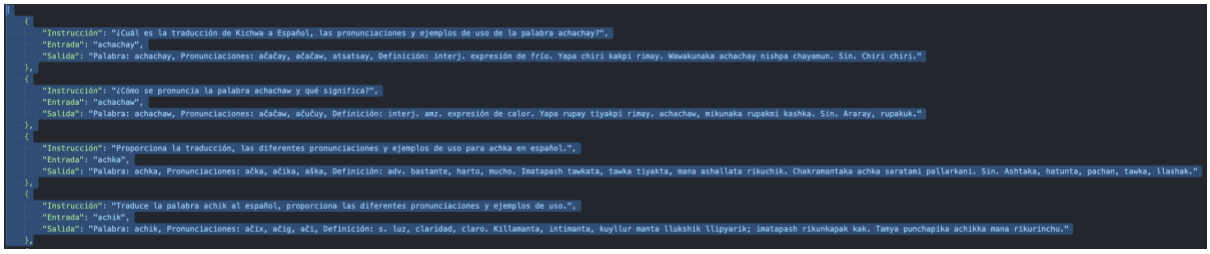
ANEXO B: SNIPPET DEL CORPUS DE DATOS

El siguiente es un reporte del tamaño total de los datos (a la fecha) con los que se está trabajando y su identificador correspondiente:

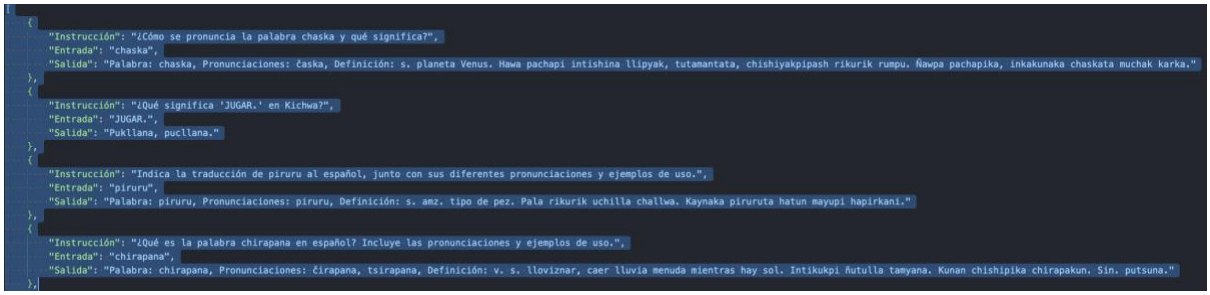
```
The first dictionary has 2066 entries.
The dictionary entries have 8838 entries.
The kichwa texts dataset has 2271 entries.
The Spanish hf dataset has 51942 entries.
```

Donde ‘first dictionary’ es el primer dataset que se llegó a formatear existosamente; ‘dictionary entries’ es el documento de palabras con sus traducciones (Español – Kichwa y Kichwa – Español); ‘kichwa texts’ es el set de datos de oraciones traducidas del kichwa al español; y Spanish HF es el set de datos descargado de Hugging Face para potenciar las habilidades del modelo en las tareas en español.

First Dictionary

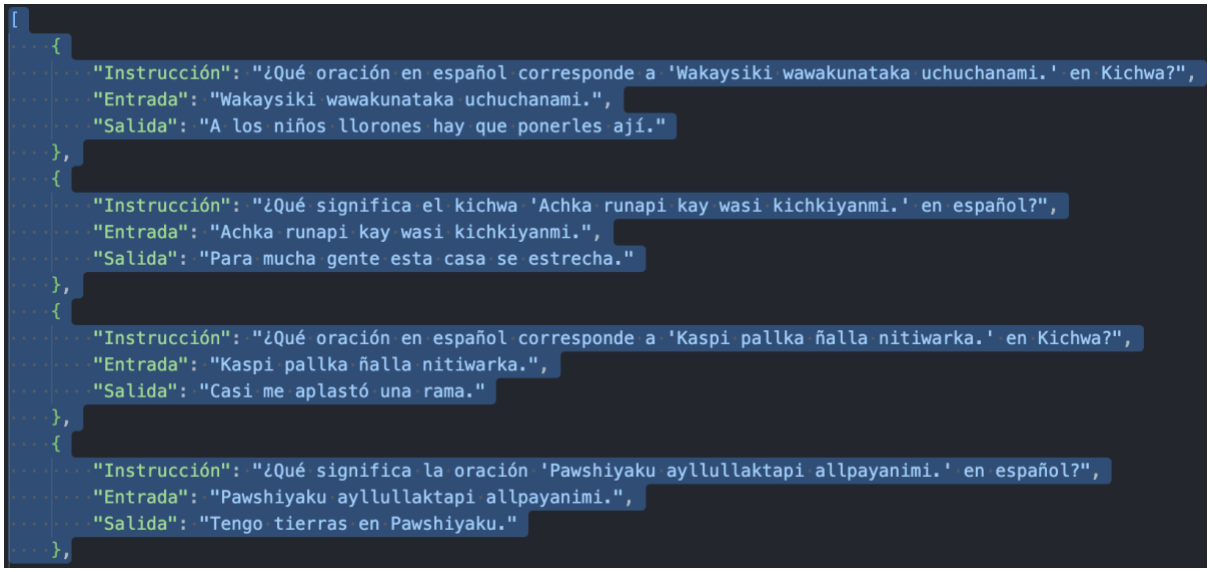


Dictionary Entries



```
[
  {
    "Instrucción": "¿Cómo se pronuncia la palabra chaska y qué significa?",
    "Entrada": "chaska",
    "Salida": "Palabra: chaska, Pronunciaciones: ʧaska, Definición: s. planeta Venus. Hawa pachapi intishina llipyak, tutamantata, chishiyakpipash rikurik rumpu. Ñawpa pachapika, inkakunaka chaskata muchak karka."
  },
  {
    "Instrucción": "¿Qué significa 'JUGAR.' en Kichwa?",
    "Entrada": "JUGAR.",
    "Salida": "Puklllana, puellana."
  },
  {
    "Instrucción": "Indica la traducción de piruru al español, junto con sus diferentes pronunciaciones y ejemplos de uso.",
    "Entrada": "piruru",
    "Salida": "Palabra: piruru, Pronunciaciones: piruru, Definición: s. amz. tipo de pez. Pala rikurik uchilla challwa. Kaynaka piruruta hatun mayupi hapirkani."
  },
  {
    "Instrucción": "¿Qué es la palabra chirapana en español? Incluye las pronunciaciones y ejemplos de uso.",
    "Entrada": "chirapana",
    "Salida": "Palabra: chirapana, Pronunciaciones: ʧirapana, tsirapana, Definición: v. s. lloviznar, caer lluvia menuda mientras hay sol. Intikukpi ñutulla tamyana. Kunan chishipika chirapakun. Sin. putsuna."
  },
```


Kichwa Texts



```
[
    {
        "Instrucción": "¿Qué oración en español corresponde a 'Wakaysiki wawakunataka uchuchanami.' en Kichwa?",
        "Entrada": "Wakaysiki wawakunataka uchuchanami.",
        "Salida": "A los niños llorones hay que ponerles ají."
    },
    {
        "Instrucción": "¿Qué significa el kichwa 'Achka runapi kay wasi kichkiyanmi.' en español?",
        "Entrada": "Achka runapi kay wasi kichkiyanmi.",
        "Salida": "Para mucha gente esta casa se estrecha."
    },
    {
        "Instrucción": "¿Qué oración en español corresponde a 'Kaspi pallka ñalla nitiwarka.' en Kichwa?",
        "Entrada": "Kaspi pallka ñalla nitiwarka.",
        "Salida": "Casi me aplastó una rama."
    },
    {
        "Instrucción": "¿Qué significa la oración 'Pawshiyaku ayllullaktapi allpayanimi.' en español?",
        "Entrada": "Pawshiyaku ayllullaktapi allpayanimi.",
        "Salida": "Tengo tierras en Pawshiyaku."
    },
```


Spanish HF

```
[
    {
        "Instrucción": "¿Qué significa DNA?",
        "Entrada": "",
        "Salida": "DNA significa ácido desoxirribonucleico."
    },
    {
        "Instrucción": "¿Cuál es la capital de Francia?",
        "Entrada": "",
        "Salida": "La capital de Francia es París."
    },
    {
        "Instrucción": "Identifica el que no encaja.",
        "Entrada": "Twitter, Instagram, Telegram",
        "Salida": "Telegram"
    },
    {
        "Instrucción": "Traduisez la phrase suivante en français.",
        "Entrada": "I miss you",
        "Salida": "Je te manque."
    },
```