UNIVERSIDAD NACIONAL DE SAN MARTÍN

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

PROYECTO DE INTELIGENCIA ARTIFICIAL: TRADUCTOR INTELIGENTE DE AIMARA: UN ENFOQUE BASADO EN IA PARA EL APRENDIZAJE DE IDIOMAS

CURSO:

INTELIGENCIA ARTIFICIAL

DOCENTE:

ING. JOHN CLARK SANTA MARÍA PINEDO

ESTUDIANTES

LUIS CARLOS ARÉVALO OCAMPO

JEYSOND ALTAMIRANO VEGA

VICTOR HUGO BRAVO GARCÍA

KAREN ALELY CHAMAYA GUEVARA

2024 I

HISTORIAL DE REVICIONES

Fecha	Versión	Descripción	Autor
22/04/2024	1.0	Primera parte	Victor Bravo, Jeysond Altamirano, Luis Arevalo, Karen Chamaya
30/05/2024	1.2	Segunda parte	Victor Bravo, Jeysond Altamirano Luis Arevalo, Karen Chamaya
08/07/2024	1.3	Tercera parte	Victor Bravo, Jeysond Altamirano Luis Arevalo, Karen Chamaya

CONTENIDO

1. PROYECTO DE INTELIGENCIA ARTIFICIAL

1.1. TÍTULO DEL TRABAJO

Traductor Inteligente de aimara: Un Enfoque Basado en IA para el Aprendizaje de Idiomas

2. INTRODUCCIÓN

En nuestra era de globalización acelerada y avances tecnológicos constantes, la interconexión global promete oportunidades sin precedentes en comunicación y aprendizaje. Sin embargo, esta nueva era también presenta desafíos significativos, especialmente para las comunidades que hablan lenguas indígenas. Entre estas lenguas se encuentra el Aimara, un idioma con una rica historia y cultura, hablado en varias regiones de América del Sur, incluyendo el Perú. A pesar de su importancia cultural y lingüística, el aimara enfrenta amenazas de erosión debido a las fuerzas de la digitalización y la globalización que remodelan el tejido social y cultural de sus comunidades hablantes.

La revolución digital, que ha transformado las sociedades modernas, ha avanzado rápidamente, introduciendo nuevas herramientas y plataformas que facilitan desde la comunicación hasta la educación. No obstante, la adopción de estas tecnologías no ha sido uniforme. Muchas comunidades indígenas, incluidos los hablantes de Aimara, han experimentado una penetración tecnológica limitada. Los factores detrás de esta limitación son múltiples y complejos, abarcando desde barreras socioeconómicas hasta la falta de infraestructura adecuada y políticas de inclusión efectivas. Esta disparidad en el acceso tecnológico no solo priva a estas comunidades de recursos educativos esenciales y oportunidades de desarrollo cultural, sino que también restringe su capacidad para participar en la creciente economía digital global.

Esta brecha digital representa un obstáculo significativo para la preservación y promoción del idioma Aimara. En un mundo donde el idioma y la cultura están profundamente entrelazados con la tecnología, la falta de herramientas digitales adaptadas a las necesidades lingüísticas y culturales específicas de los hablantes de Aimara puede llevar a una gradual disminución del uso del idioma en contextos cotidianos y, eventualmente, a su desaparición.

Por lo tanto, es imperativo abordar estos desafíos mediante la integración consciente de estas comunidades en la sociedad digital. Esto implica no solo proporcionar el acceso a la tecnología, sino también asegurar que dichas tecnologías sean culturalmente relevantes y lingüísticamente adaptadas para reforzar el uso del Aimara en lugar de reemplazarlo. El proyecto del "Traductor Inteligente de Aimara" surge como una iniciativa vital en esta dirección, con el objetivo de desarrollar una herramienta de traducción y aprendizaje basada en la inteligencia artificial que sea accesible, adaptativa y respetuosa con la herencia lingüística del Aimara.

Este proyecto busca no solo cerrar la brecha digital, sino también revitalizar el interés y el uso del Aimara a través de plataformas modernas, asegurando que el idioma no solo sobreviva, sino que prospere en la era digital. Al hacerlo, se espera fortalecer la identidad cultural de las comunidades Aimara y empoderarlas para que desempeñen un papel activo en la narrativa global, contribuyendo a un mundo más inclusivo y diverso donde la tecnología sirve como puente entre las tradiciones del pasado y las innovaciones del futuro.

2.1. PROPÓSITO Y OBJETIVOS DEL PROYECTO

2.1.1. PROPÓSITO

El propósito de este proyecto es desarrollar un "Traductor Inteligente de Aimara", utilizando tecnologías de inteligencia artificial, para facilitar el aprendizaje y la preservación de este idioma. Este sistema no solo busca ser una herramienta educativa para los hablantes y estudiantes del Aimara sino también actuar como un puente cultural que fomente un mayor entendimiento y respeto por esta rica herencia lingüística y cultural.

2.1.2. OBJETIVOS GENERALES

- Desarrollar y optimizar una herramienta de traducción y aprendizaje basada en inteligencia artificial para el idioma Aimara.

2.1.3. OBJETIVOS ESPECÍFICOS

- Asegurar que la tecnología sea fácil de usar para los hablantes de Aimara.
- Implementar tecnología de IA adaptativa: Utilizar modelos de procesamiento del lenguaje natural que aprendan y se adapten a las peculiaridades del Aimara.
- Reducir la brecha tecnológica y promover el uso de tecnologías de información en comunidades que hablan Aimara.
- Contribuir a la revitalización del idioma, asegurando que su uso se extienda también al dominio digital.

2.2. BREVE DESCRIPCIÓN DEL PROBLEMA O NECESIDAD QUE SE ABORDA

El Perú, conocida por su rica diversidad cultural y lingüística, enfrenta desafíos significativos que amenazan la sostenibilidad y vitalidad de su patrimonio cultural. Entre estos desafíos, uno de los más críticos es la disparidad en el acceso a la tecnología y a recursos educativos adecuados. Esta problemática es particularmente aguda para los hablantes del idioma Aimara, quienes se encuentran en una encrucijada crucial, entre la preservación de su lengua ancestral y la integración en una sociedad globalizada y digitalmente conectada.

2.2.1. Desigualdad Tecnológica y Educativa

Los habitantes de Peru, especialmente aquellos pertenecientes a comunidades indígenas Aimara, enfrentan una notable falta de acceso a infraestructuras tecnológicas básicas como internet de alta velocidad, dispositivos informáticos y software educativo. Esta brecha digital no solo limita su participación en la economía digital emergente, sino que también restringe su acceso a oportunidades educativas que podrían facilitar el aprendizaje continuo y el desarrollo personal.

2.2.2. Impacto en la Comunidad y la Transmisión Cultural

La falta de acceso a tecnologías adecuadas también afecta profundamente la dinámica social y cultural de la comunidad Aimara. En un mundo donde la comunicación y el intercambio cultural son predominantemente digitales, estas comunidades se ven marginadas, no solo económicamente sino también social y culturalmente. Este aislamiento digital impide que los jóvenes Aimara se vean a sí mismos como parte de una comunidad global, lo que puede llevar a un desinterés por su propia herencia cultural.

2.2.3. La Necesidad de Soluciones Adaptativas y Culturalmente Apropiadas

Ante esta realidad, surge la necesidad imperiosa de desarrollar soluciones que no solo aborden la brecha tecnológica, sino que también reconozcan y fortalezcan la identidad cultural de las comunidades Aimara. El proyecto del "Traductor Inteligente de Aimara" está diseñado para enfrentar estos retos de manera integral. Mediante la implementación de tecnologías inclusivas y adaptativas, este proyecto busca proporcionar a los hablantes de Aimara herramientas que faciliten el aprendizaje y uso de su lengua en contextos modernos, a la vez que respetan y realzan su cultura única.

Este proyecto tiene el potencial de transformar significativamente el panorama educativo y cultural para los hablantes de Aimara en Perú. Al integrar el Aimara en plataformas digitales mediante un traductor inteligente, se espera aumentar el uso del idioma en una variedad de contextos, desde la educación hasta la comunicación diaria y la participación en la sociedad digital.

3. CONTEXTO Y JUSTIFICACIÓN

Actualmente, en el Perú existen 47 lenguas indígenas, cuatro de ellas se hablan en la zona andina y 43 en la zona amazónica. Las lenguas que tienen mayor número de hablantes son el quechua, aimara, asháninka y awajún. Es importante resaltar que 13 de cada 100 peruanos y peruanas hablan una lengua indígena los que se enfrentan continuamente al problema de la extinción, aun cuando constituyen patrimonio cultural inmaterial de los pueblos originarios del Perú

En los últimos cuatro siglos, han desaparecido al menos 35 lenguas, y en la actualidad, tres lenguas están en peligro y 18 en serio peligro de desaparecer. Esto plantea un problema muy importante y trascendente para la cultura y los valores humanos, que no es exclusivo de nuestro país: la preservación de la cultura y las lenguas indígenas. La lengua Aimara es importante como lengua indígena ya que es hablada en las regiones de Puno, Moquegua y Tacna y que tiene entre medio millón de hablantes.

La lengua no tiene el alcance ni la cantidad suficiente de hablantes para poder seguir siendo transmitida. Una estrategia prometedora, desde el punto de vista informático, es la posibilidad de construir un traductor automático que nos proporcione de manera rápida y confiable textos traducidos del español al Aimara y viceversa. Esta tarea consistiría en recopilar la mayor cantidad de recursos bibliográficos completamente traducidos (corpus en paralelo) y diseñar un modelo de aprendizaje de máquina que permita aprender el lenguaje para poder brindar traducciones eficientes.

3.1. IMPORTANCIA DEL PROYECTO DE INTELIGENCIA ARTIFICIAL EN EL ÁREA DE ESTUDIO

La inteligencia artificial (IA) ha emergido como una herramienta poderosa en la preservación y revitalización de lenguas minoritarias y dialectos indígenas, y el idioma Aimara no es una excepción. Con la capacidad de procesar grandes cantidades de datos lingüísticos y desarrollar algoritmos específicos para el análisis y la generación de texto, la IA puede contribuir significativamente a la traducción automática, la generación de contenido y la creación de recursos lingüísticos en Aimara.

Uno de los mayores desafíos que enfrentan las comunidades Aimaras es la escasez de materiales y recursos en su idioma. Aquí es donde la IA puede marcar la diferencia. Los sistemas de traducción automática pueden ayudar a superar las barreras lingüísticas al proporcionar traducciones rápidas y precisas entre el Aimara y otros idiomas ampliamente utilizados. Estas traducciones no solo facilitan el acceso a la información para las personas que hablan Aimara, sino que también promueven la preservación y difusión de la lengua en contextos digitales.

Además de la traducción automática, la IA puede ser utilizada para desarrollar herramientas de procesamiento de lenguaje natural (PLN) específicamente diseñadas para el Aimara. Estas herramientas pueden incluir correctores ortográficos, sintetizadores de voz, reconocedores de voz y sistemas de diálogo que permiten una comunicación más fluida y natural en Aimara en diversas plataformas digitales.

La generación de contenido también es un aspecto crucial en la preservación y promoción del Aimara. La IA puede ser utilizada para crear textos, historias y materiales educativos en Aimara, adaptados a las necesidades y preferencias de las comunidades Aimaras. Estos recursos no solo enriquecen el corpus lingüístico del Aimara, sino que también fortalecen la identidad cultural y el sentido de pertenencia de quienes lo hablan.

Otro aspecto importante es la capacitación y el empoderamiento de las comunidades Aimaras en el uso de herramientas de IA. A través de programas de capacitación y alfabetización digital, las personas pueden adquirir habilidades para utilizar y desarrollar tecnologías de IA adaptadas a sus necesidades lingüísticas y culturales. Esto no solo fomenta la autonomía y el liderazgo dentro de las comunidades, sino que también abre oportunidades de empleo y emprendimiento en el campo de la tecnología y la lingüística computacional.

3.2. REVISIÓN DE TRABAJOS O PROYECTOS RELACIONADOS

3.2.1. ANTECEDENTES NACIONALES

A nivel nacional, Galarreta Asian (2017) en su trabajo sobre el shipibo-konibo, otro idioma indígena del Perú, resalta la necesidad de generar y alinear corpus para entrenar eficazmente los sistemas de traducción automática. Este proyecto es directamente relevante para nuestro trabajo, ya que también se basará en la generación de corpus paralelos en Aimara y español para entrenar modelos de traducción estadística.

Bragagnini Mendizabal (2019) ilustra cómo las innovaciones en la arquitectura de traducción automática pueden mejorar significativamente la calidad de la traducción. Este enfoque innovador es algo que se buscará incorporar en el traductor de Aimara, utilizando lo último en tecnología de inteligencia artificial para optimizar la precisión y la relevancia cultural de las traducciones.

3.3. CÓMO EL PROYECTO SE DIFERENCIA O COMPLEMENTA LAS SOLUCIONES EXISTENTES

El proyecto "Traductor Inteligente de Aimara: un enfoque basado en ia para el aprendizaje de idiomas " se diferencia y complementa las soluciones existentes en varios aspectos clave, aprovechando tanto los avances en tecnología de traducción automática como las lecciones aprendidas de proyectos anteriores.

Estas diferencias y complementos se centran en adaptar y extender tecnologías existentes para satisfacer de manera más efectiva las necesidades específicas de la comunidad Aimara, y en introducir enfoques innovadores en el contexto de lenguas con recursos limitados. A continuación, se detallan algunos de estos puntos:

3.3.1. DIFERENCIACIÓN

- Especialización Lingüística y Cultural: A diferencia de muchos sistemas de traducción automática que se centran en idiomas ampliamente hablados, este proyecto está dedicado exclusivamente al Aimara. Esto permite una profundización en las estructuras gramaticales, usos idiomáticos y variaciones dialectales específicas del Aimara, que raramente son el foco en sistemas más generalizados.
- Enfoque Comunitario: El proyecto planea involucrar activamente a la comunidad Aimara en el proceso de desarrollo del traductor, asegurando que el producto final sea no solo tecnológicamente avanzado, sino también culturalmente apropiado y relevante para sus usuarios finales. Esta aproximación es fundamental para asegurar la aceptación y utilidad práctica de la herramienta dentro de la comunidad.

3.3.2. COMPLEMENTARIEDAD

- Integración de Avances Recientes en IA: Tomando inspiración de trabajos recientes como los de Bragagnini Mendizabal, el proyecto aplicará las últimas técnicas en IA, como las redes neuronales y los modelos de atención, para mejorar la calidad de la traducción del Aimara. Esto complementa los esfuerzos anteriores que han dependido más de la traducción automática estadística, incorporando avances más recientes en el campo del procesamiento del lenguaje natural.
- Plataforma de Contribución Abierta: Siguiendo el modelo de Haya Coll, se desarrollará una plataforma web donde los usuarios puedan contribuir con textos bilingües, mejorar las traducciones existentes y adaptar la herramienta a sus necesidades específicas. Esta característica facilita la mejora continua del sistema de traducción y promueve una comunidad activa de usuarios que contribuyen al enriquecimiento del modelo lingüístico del Aimara.
- Compatibilidad y Extensibilidad: El sistema será diseñado para ser compatible con otras herramientas y plataformas, permitiendo su integración con sistemas educativos y comunicativos ya existentes en el Perú. Esto asegura que el traductor no solo sirva como una solución aislada, sino que pueda ser parte de una infraestructura tecnológica más amplia.

4. DEFINICIÓN DEL PROBLEMA

El problema principal que este proyecto busca abordar es la falta de acceso a herramientas efectivas de traducción automática adaptadas específicamente para el idioma Aimara. A pesar de la creciente digitalización global, estas comunidades a menudo se encuentran marginadas de los beneficios de la tecnología moderna, particularmente en lo que respecta a la educación, la cultura y la interacción social. Esta desconexión no solo limita sus oportunidades de desarrollo personal y comunitario, sino que también amenaza la preservación y el fomento de su lengua y cultura.

4.1. DETALLE DEL PROBLEMA ESPECÍFICO A RESOLVER

- Escasez de Recursos Lingüísticos: El Aimara, al ser un idioma con recursos limitados en el ámbito digital, carece de los corpus extensos, diccionarios bilingües y bases de datos gramaticales que son comunes en idiomas más difundidos. Esta falta de recursos fundamentales dificulta el desarrollo de herramientas de traducción automática eficaces.
- Limitaciones Tecnológicas: Existe un déficit de aplicaciones tecnológicas, como software de traducción automática, que sean capaces de manejar las estructuras lingüísticas y las particularidades culturales del Aimara. Las herramientas existentes están predominantemente diseñadas para idiomas ampliamente hablados y no se adaptan bien a las características únicas del Aimara, como su sintaxis y morfología.

- Acceso a la Información y Comunicación: La falta de estas tecnologías adaptadas impide que los hablantes de Aimara accedan plenamente a la información disponible en español o en otros idiomas, limitando su capacidad para participar en debates educativos, culturales y políticos más amplios. Asimismo, dificulta la producción de contenido en Aimara que pueda ser comprendido y utilizado por hablantes no nativos, restringiendo las oportunidades de intercambio cultural y lingüístico.

4.2. DATOS Y RECURSOS DISPONIBLES

Se cuenta con datos lingüísticos en Aimara y español, aunque estos pueden ser limitados y no estar debidamente alineados para el entrenamiento de modelos de traducción automática. Además, existen algunas herramientas y recursos de inteligencia artificial disponibles, pero pueden no estar optimizados para las particularidades del idioma Aimara.

4.3. RESTRICCIONES Y DESAFÍOS IDENTIFICADOS

El desarrollo de un traductor inteligente de Aimara plantea diversos desafíos y restricciones que son críticos para considerar en la planificación y ejecución del proyecto. Algunos de estos desafíos incluyen aspectos técnicos, culturales, y logísticos, entre otros:

4.3.1. RESTRICCIONES Y DESAFÍOS TÉCNICOS

- Escasez de Recursos Lingüísticos: Como ya se mencionó, uno de los principales desafíos es la falta de recursos lingüísticos digitalizados y estructurados para el Aimara. La creación de un corpus suficientemente grande y representativo es esencial para entrenar modelos de traducción automática con alta precisión.
- Complejidad Lingüística del Aimara: El Aimara posee características estructurales y gramaticales únicas que pueden complicar el desarrollo de algoritmos de traducción automática. Adaptar la tecnología de procesamiento de lenguaje natural a estas particularidades es un desafío técnico considerable.
- Integración Tecnológica: La integración de nuevas tecnologías en sistemas existentes o la creación de nuevas plataformas tecnológicas que sean accesibles y útiles para la comunidad Aimara requiere una infraestructura tecnológica robusta y adaptativa.

4.3.2. DESAFÍOS CULTURALES Y SOCIALES

- Aceptación por Parte de la Comunidad: La adopción de tecnología en comunidades que pueden ser tradicionalmente escépticas o desconocedoras de sus beneficios requiere esfuerzos de sensibilización y capacitación. Asegurar que la comunidad vea el valor y se sienta parte del proyecto es esencial para su éxito.
- Preservación de la Lengua y la Cultura: Existe el riesgo de que la tecnología pueda influir en la forma en que se usa el idioma, potencialmente alterando aspectos culturales significativos. Es crucial que el desarrollo del traductor respete y refuerce la autenticidad cultural y lingüística del Aimara.

4.3.3. RESTRICCIONES ECONÓMICAS Y DE FINANCIACIÓN

- Sostenibilidad Financiera: Asegurar la financiación continua para el desarrollo, mantenimiento y actualización de las herramientas de traducción es un desafío significativo. Esto incluye financiar la infraestructura tecnológica, la capacitación de usuarios y la expansión del corpus y las herramientas.
- Recursos Humanos: Reclutar y retener personal cualificado que tenga competencias tanto en tecnología de inteligencia artificial como en lingüística y cultura Aimara puede ser difícil, especialmente en áreas remotas o menos desarrolladas.

4.3.4. DESAFÍOS LOGÍSTICOS

- Acceso y Disponibilidad de Tecnología: Asegurar que las herramientas desarrolladas sean accesibles para todos los hablantes de Aimara, incluidos aquellos en áreas rurales o con acceso limitado a la tecnología, es un reto significativo.
- Evaluación y Mejora Continua: Establecer sistemas eficaces para la evaluación continua de la herramienta y su adaptación basada en el feedback de los usuarios requiere una logística bien planificada y recursos dedicados.

4.3.5. DESAFÍOS POLÍTICOS Y NORMATIVOS

- Normativas y Políticas: Navegar por las políticas locales e internacionales que afectan la tecnología, la privacidad de datos y la propiedad intelectual puede complicar la implementación y distribución de las herramientas de traducción.

5. METOLOGÍA

Para el desarrollo de este proyecto se utilizó la metodología Scrum

5.1. ¿QUÉ ES SCRUM?

Scrum es un marco ligero que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas complejos. (Sutherland, 2020) Se podría decir que Scrum se basa en el empirismo y el pensamiento Lean. El empirismo afirma que el conocimiento se basa en la experiencia y la toma de decisiones basadas en lo que se infiere del entorno. El pensamiento Lean reduce los desperdicios y se centra en lo esencial. Scrum emplea un enfoque iterativo e incremental (las iteraciones se conocen como Sprints) para optimizar la previsibilidad y adaptar los proyectos según los riesgos presentados. En Scrum es necesario involucrar a grupos de personas que colectivamente tienen todas las habilidades y experiencia para hacer el trabajo con el fin de compartir o adquirir tales habilidades según sea necesario. Para poder implementar Scrum debe tener en cuenta que combina cuatro eventos formales para la inspección y adaptación dentro de un evento contenedor, el Sprint.

A continuación, se presenta una breve descripción de la metodología Scrum: La metodología Scrum para el desarrollo ágil de software representa un punto de partida de la gestión en cascada. De hecho, Scrum y otro tipo de procesos ágiles se inspiraron en sus limitaciones. La metodología Scrum enfatiza en la comunicación y colaboración, el funcionamiento del software, y la flexibilidad de la que dispone para adaptarse a las emergentes realidades de las empresas - todos los atributos de los que carecía el modelo de cascada.

5.2. ¿QUÉ CARACTERIZA A SCRUM?

De todas las metodologías ágiles, Scrum es única porque introduce la idea del control empírico de los procesos. Esto significa que Scrum utiliza el progreso real de un proyecto para planificar y concertar los lanzamientos. En Scrum, los proyectos se dividen en ritmos de trabajo breves, conocidos como sprints. Normalmente, tienen una, dos o tres semanas de duración. Al final de cada sprint, el cliente y los miembros del equipo se reúnen para evaluar el progreso del proyecto y planear los siguientes pasos a seguir. Esto permite que la dirección del proyecto se ajuste o se reoriente una vez finalizado el trabajo, sin especulaciones ni predicciones.

Filosóficamente, este énfasis continuo de evaluar las tareas finalizadas es el principal causante del éxito que tiene esta metodología entre los directores y desarrolladores. Pero lo que verdaderamente permite funcionar a la metodología Scrum es un conjunto de papeles, responsabilidades, y reuniones.

5.3. LOS PAPELES DE SCRUM

Scrum tiene tres papeles fundamentales:

- Product Owner: propietario del producto
- Scrum Master: especialista en Scrum
- Team Member: miembros del equipo

6. DESARROLLO DE LA METODOLOGÍA

Es importante establecer las tareas que se necesitan para cumplir con los requerimientos y los objetivos de la iteración. Las tareas deberán ser específicas, medibles y alcanzables en el plazo establecido. Deberán ser distribuidas equitativamente entre los miembros del equipo y asignadas en función de las habilidades y fortalezas de cada uno.

6.1. PERSONAS Y ROLES DEL PROYECTO

Persona	Contacto	Rol
Victor Hugo Bravo García	bravovictorhugo11@gmail.com 944469094	Desarrollador / programador
Jeysond Altamirano Vega	alveje15@gmail.com 910839328	Gestor de producto / Product Owner
Luis carlos Arevalo Ocampo	barrerita_romero@hotmail.com 933632298	Desarrollador / programador
Karen Alely Chamaya Guevara	71579004alely@gmail.com 923345139	Coordinador / Scrum Manager

6.2. ARTEFACTOS

Estos artefactos se utilizan de manera conjunta en cada Sprint para planificar el trabajo, definir los objetivos y medir el progreso del equipo. De esta manera, Scrum permite a los equipos de desarrollo trabajar de manera más eficiente y efectiva para entregar un producto de alta calidad.

Para la planificación de los Sprints, la metodología de scrum nos brinda los artefactos que ayudaran a la realización de las tareas programadas, el primer artefacto que tenemos que crear para la planificación del desarrollo del sistema es el Producto BackLog, este artefacto nos permite conocer las tareas que tenemos que realizar y categorizar por prioridades.

6.3. PLANIFICACIÓN DE LA PILA DEL SPRINT (SPRINT BACKLOG)

Una vez definida la lista de requerimientos del cliente, se procede a crear la planificación de cada uno de los Sprints en una tabla donde nos permitirá conocer cada una de las iteraciones, en la que definiremos las tareas a desarrollarse de acuerdo al nivel de prioridad establecida a cada uno de los requerimientos planteados en la Pila de Producto por el Product Owner, para lo que el equipo deberá elegir y organizar cada una de ellas.

Esta pila del Sprint también es una herramienta que provee de soporte para la comunicación directa con el equipo de desarrollo, donde el estado va ir cambiando mientras avanza la iteración. Para su elaboración y planificación de a considerado necesario reunir con la siguiente información:

- ID: Código de identificación de la tarea o sprint a desarrollarse
- Nombre de Tarea: Identificación de la tarea para el grupo de desarrolladores
- Inicio: fecha de inicio del sprint
- Fin: fecha de finalización del sprint
- Trabajo Previsto/ Estimación Inicial en horas: Tiempo en días previsto de la duración del sprint en su desarrollo. escogido por el equipo de desarrollo.
- Trabajo Previsto/ Estimación Inicial en días: Tiempo en días previsto de la duración del sprint en su desarrollo. escogido por el equipo de desarrollo.
- % de trabajo completado: Porcentaje en números del avance del sprint determinado por equipo de desarrollo.
- Real: número de días raléales
- Responsable: responsable de la tarea

Pila del sprint (Sprint Backlog)

Elaborado por el equipo

			Estimación		Sin domingos	Laborables
Sprint	inicio	fin	Dias	Horas	Dias Totales	Horas Totales
1	15/04/2024	27/05/2023	12	43	13	45
2	28/05/2023	8/07/2024	29	87	33	90

Sprint 1	Nombre de la Tarea	Inicio	Fin	Estimación (horas)	Estimación (días)	Real (días)	% Trabajo completado	Responsable
1	Primera presentación	15/04/2024	27/05/2024	189	61	72	100%	
T1	Carga de DatoS			2	2	2	100%	Equipo Scrum
T2	Limpieza de Datos			1	1	1	100%	Equipo Scrum
T3	Tokenización			2	1	2	100%	Equipo Scrum
T4	Preparación de Secuencias			4	2	2	100%	Equipo Scrum
T5	Creación de Tensores de Dato			5	1	2	100%	Equipo Scrum
T6	Configuración del Dataset			27	3	4	100%	Equipo Scrum

Sprint 2	Nombre de la Tarea	Inicio	Fin	Estimación (horas)	Estimación (días)	Real (días)	% Trabajo completado	Responsable
2	Segunda presentación	28/05/2024	8/07/2024	87	29	33	100%	
T7	Implementación parcial de la Capa de Codificación Posicional			10	2	3	100%	Equipo Scrum
T8	Definición del Encoder			3	1	1	100%	Equipo Scrum
T9	Definición del Mecanismo de Atención			5	3	3	100%	Equipo Scrum
T10	Definición del Decoder			4	2	2	100%	Equipo Scrum

T11	Corrección y finalización de la Capa de Codificación Posicional	1	1	1	100%	Equipo Scrum
T12	Configuración de la Función de Pérdida y Optimizador	3	1	1	100%	Equipo Scrum
T13	Definición del Bucle de Entrenamiento	3	1	1	100%	Equipo Scrum
T14	Entrenamiento del Modelo	6	2	2	100%	Equipo Scrum
T15	Evaluación del Modelo	8	3	4	100%	Equipo Scrum
T16	Ajuste de Hiperparámetros	12	4	5	100%	Equipo Scrum
T17	Exportación del Modelo	6	2	3	100%	Equipo Scrum
T18	Creación de la API de Inferencia	5	1	1	100%	Equipo Scrum
T19	Despliegue del Modelo	2	1	1	100%	Equipo Scrum
T20	Monitoreo en Producción	9	2	2	100%	Equipo Scrum
T21	Actualización y Reentrenamiento	10	3	3	100%	Equipo Scrum

18

6.4. HERRAMIENTAS UTILIZADAS EN LA PROGRAMACIÓN

El proyecto "Traductor Inteligente de Aimara" es una iniciativa que busca desarrollar una inteligencia artificial capaz de traducir y facilitar el aprendizaje del idioma aimara. Para llevar a cabo este proyecto, se han seleccionado cuidadosamente varias herramientas tecnológicas, cada una con sus ventajas específicas que contribuyen al éxito del proyecto.

6.4.1. VISUAL STUDIO CODE

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Se eligió esta herramienta por sus múltiples ventajas:

- → Editor Ligero: VS Code es rápido y ligero, permitiendo una edición de código eficiente.
- → Extensiones: Ofrece una amplia gama de extensiones que facilitan el desarrollo, incluyendo aquellas específicas para Python y TensorFlow.
- → Integración con Git: Facilita el control de versiones y la colaboración en equipo.
- → Depuración: Proporciona herramientas de depuración integradas que son esenciales para el desarrollo de proyectos complejos de IA.

Aplicación en el proyecto:

VS Code se utiliza para escribir y gestionar el código fuente del traductor. Las extensiones de Python y TensorFlow facilitan la escritura y depuración del código, y la integración con Git permite un control de versiones eficiente.

6.4.2. GOOGLE COLAB

Google Colab es una plataforma gratuita que permite escribir y ejecutar código en Python en el navegador, con un entorno de ejecución basado en Jupyter Notebooks. Cuenta con las siguientes características.

- → Acceso a GPUs y TPUs: Facilita el acceso a recursos computacionales avanzados para entrenar modelos de aprendizaje profundo.
- → Colaboración en tiempo real: Permite que varios usuarios trabajen simultáneamente en el mismo notebook.
- → Integración con Google Drive: Facilita el almacenamiento y la gestión de archivos.

Aplicación en el proyecto:

Google Colab se utiliza principalmente para el entrenamiento de modelos de traducción y procesamiento de lenguaje natural. Su capacidad para utilizar GPUs y TPUs acelera el proceso de entrenamiento, y su integración con Google Drive facilita la gestión de datasets y resultados.

6.4.3. TENSORFLOW

TensorFlow es una biblioteca de código abierto desarrollada por Google para realizar tareas de machine learning y deep learning.

- → Versatilidad: Soporta una amplia gama de algoritmos y modelos de machine learning.
- → Escalabilidad: Permite el entrenamiento y despliegue de modelos en diferentes dispositivos, desde móviles hasta servidores de alto rendimiento.
- → Comunidad activa: Cuenta con una gran comunidad de desarrolladores que contribuyen y mantienen la biblioteca.

Aplicación en el proyecto:

TensorFlow se utiliza para crear, entrenar y optimizar los modelos de traducción automática. Su capacidad para manejar grandes volúmenes de datos y realizar cálculos intensivos en paralelo lo hace ideal para este tipo de tareas.

6.4.4. PYTHON

Python es un lenguaje de programación ampliamente utilizado en el campo de la inteligencia artificial y el machine learning debido a su simplicidad y robustez.

- → Sintaxis simple y clara: Facilita el desarrollo rápido y eficiente.
- → Bibliotecas y frameworks: Cuenta con una gran cantidad de bibliotecas especializadas en machine learning, como TensorFlow, Keras, y scikit-learn.
- → Comunidad y recursos: Una extensa comunidad de usuarios y desarrolladores proporciona soporte y recursos educativos.

Aplicación en el proyecto:

Python es el lenguaje principal utilizado para desarrollar el traductor inteligente. Todas las tareas de preprocesamiento de datos, desarrollo de modelos, entrenamiento y evaluación se realizan en Python.

6.5. DESARROLLO DEL PROYECTO

En esta sección se describe detalladamente el proceso de desarrollo del traductor inteligente de Aimara.

6.5.1. RECOLECCIÓN Y PREPROCESAMIENTO DE DATOS

La recolección de datos es una fase crucial para cualquier proyecto de inteligencia artificial. En este caso, se recopilaron textos en Aimara y sus traducciones correspondientes en español. Las fuentes de datos incluyeron libros, artículos, sitios web y documentos oficiales. Se prestó especial atención a la calidad y la relevancia de los datos, asegurando que las traducciones fueran precisas y contextualmente adecuadas.

El preprocesamiento de los datos implicó varias etapas, como la eliminación de caracteres especiales, la normalización de texto (conversión a minúsculas, eliminación de espacios en blanco adicionales), y la tokenización (división del texto en palabras o sub-palabras).

6.5.2. DESARROLLO DEL MODELO DE IA

ARQUITECTURA DEL MODELO

La arquitectura del modelo se diseñó utilizando TensorFlow. Se optó por una red neuronal secuencial con varias capas, incluyendo capas de embedding para representar palabras en un espacio vectorial, capas LSTM (Long Short-Term Memory) para manejar dependencias a largo plazo en el texto, y una capa densa final para la predicción de palabras. Se experimentó con diferentes configuraciones de hiperparámetros (número de neuronas, tamaño de los embeddings, tasa de aprendizaje, etc.) para encontrar la mejor configuración.

ENTRENAMIENTO

El modelo fue entrenado en Google Colab utilizando recursos de GPU para acelerar el proceso. El entrenamiento implicó la optimización de los parámetros del modelo mediante la minimización de la función de pérdida, utilizando algoritmos como Adam y técnicas de regularización como dropout para prevenir el sobreajuste. Se llevaron a cabo múltiples iteraciones de entrenamiento, ajustando hiperparámetros y evaluando el rendimiento del modelo en un conjunto de validación.

EVALUACIÓN

Para evaluar el rendimiento del modelo, se utilizaron métricas como BLEU (Bilingual Evaluation Understudy) y ROUGE (Recall-Oriented Understudy for Gisting Evaluation). Estas métricas permitieron medir la calidad de las traducciones generadas en comparación con las traducciones de referencia. Además, se realizaron evaluaciones manuales para asegurar que las traducciones fueran coherentes y precisas.

CÓDIGO FUENTE DEL TRADUCTOR INTELIGENTE DE AIMARA

A continuación, se presenta el código utilizado para desarrollar el traductor inteligente de aimara, organizado en orden secuencial para facilitar su comprensión y replicación. Este código incorpora las mejores prácticas de desarrollo de modelos de traducción basados en IA y está optimizado para manejar las particularidades del idioma aimara.

Ilustración 1.

Buscar y reemplazar

```
numpy as np
import math
import re
import time
import tensorflow as tf
from tensorflow.keras import layers
import tensorflow_datasets as tfds

# Cargando archivos aimara
with open("/content/sample_data/oraciones-aimara.en", mode="r", encoding="utf-8") as f:
    aimara_ai = f.read()

# Cargando archivos español
with open("/content/sample_data/oraciones-español.es", mode="r", encoding="utf-8") as f:
    aimara_es = f.read()

# Limpiar los datos de aimara
corpus_ai = aimara_ai
corpus_ai = re.sub(r"\.(?=[0-9]|[a-z]|[A-Z])", ".$$$", corpus_ai)

# Remover $$$$ marcadores
corpus_ai = re.sub(r".\$\$\$", "", corpus_ai)

# Remover los espacios
corpus_ai = re.sub(r"  +", " ", corpus_ai)

corpus_ai = corpus_ai.split("\n")

# Limpiar los datos de español
corpus_es = aimara_es
corpus_es = re.sub(r"\.(?=[0-9]|[a-z]|[A-Z])", ".$$$", corpus_es)

# Remover $$$$ marcadores
corpus_es = re.sub(r".\$\$\$", "", corpus_es)
```

Ilustración 2.

```
# Remover $$$$ marcadores
corpus_es = re.sub(r".\$\$\$", "", corpus_es)

# Remover los espacios
corpus_es = re.sub(r"  +", " ", corpus_es)

corpus_es = corpus_es.split("\n")
print(corpus_ai, corpus_es)

# Tokenización aimara
tokenizer_ai = tfds.deprecated.text.SubwordTextEncoder.build_from_corpus(
    corpus_ai, target_vocab_size=2**13 #estableciendo un tamaño de vocabulario objetivo de 8192
)

# Tokenización español
tokenizer_es = tfds.deprecated.text.SubwordTextEncoder.build_from_corpus(
    corpus_es, target_vocab_size=2**13 #estableciendo un tamaño de vocabulario objetivo de 8192
)

VOCAB_SIZE_AI = tokenizer_ai.vocab_size + 2
VOCAB_SIZE_ES = tokenizer_es.vocab_size + 2

print("VOCAB_SIZE_AI:", VOCAB_SIZE_AI)
print("VOCAB_SIZE_ES:", VOCAB_SIZE_ES)

# inputs
inputs = [[VOCAB_SIZE_AI - 2] + tokenizer_ai.encode(sentence) + [VOCAB_SIZE_AI - 1]
          for sentence in corpus_ai
          ]

outputs = [[VOCAB_SIZE_ES - 2] + tokenizer_es.encode(sentence) + [VOCAB_SIZE_ES - 1]
          for sentence in corpus_es
          ]

# Removiendo oraciones largas
MAX_LENGHT = 20
```

Ilustración 3.

```
# Removiendo oraciones largas
MAX_LENGHT = 20
idx_to_remove = [count for count, sent in enumerate(inputs)
                 if len(sent) > MAX_LENGHT
                ]
for idx in reversed(idx_to_remove):
  del inputs[idx]
  del outputs[idx]

idx_to_remove = [count for count, sent in enumerate(outputs)
                 if len(sent) > MAX_LENGHT
                ]

for idx in reversed(idx_to_remove):
  del inputs[idx]
  del outputs[idx]

# crear inputs y outputs
inputs = tf.keras.preprocessing.sequence.pad_sequences(inputs,
                                                       value = 0,
                                                       padding = "post",
                                                       maxlen=MAX_LENGHT)

outputs = tf.keras.preprocessing.sequence.pad_sequences(outputs,
                                                        value = 0,
                                                        padding = "post",
                                                        maxlen=MAX_LENGHT)

# declarar numero de lotes, (utilizaremos 24 lotes)
BATCH_SIZE = 64
BUFFER_SIZE = 20000 #Es la cantidad de set de datos con las que se va a saltar

dataset = tf.data.Dataset.from_tensor_slices((inputs, outputs))

dataset = dataset.cache()
```

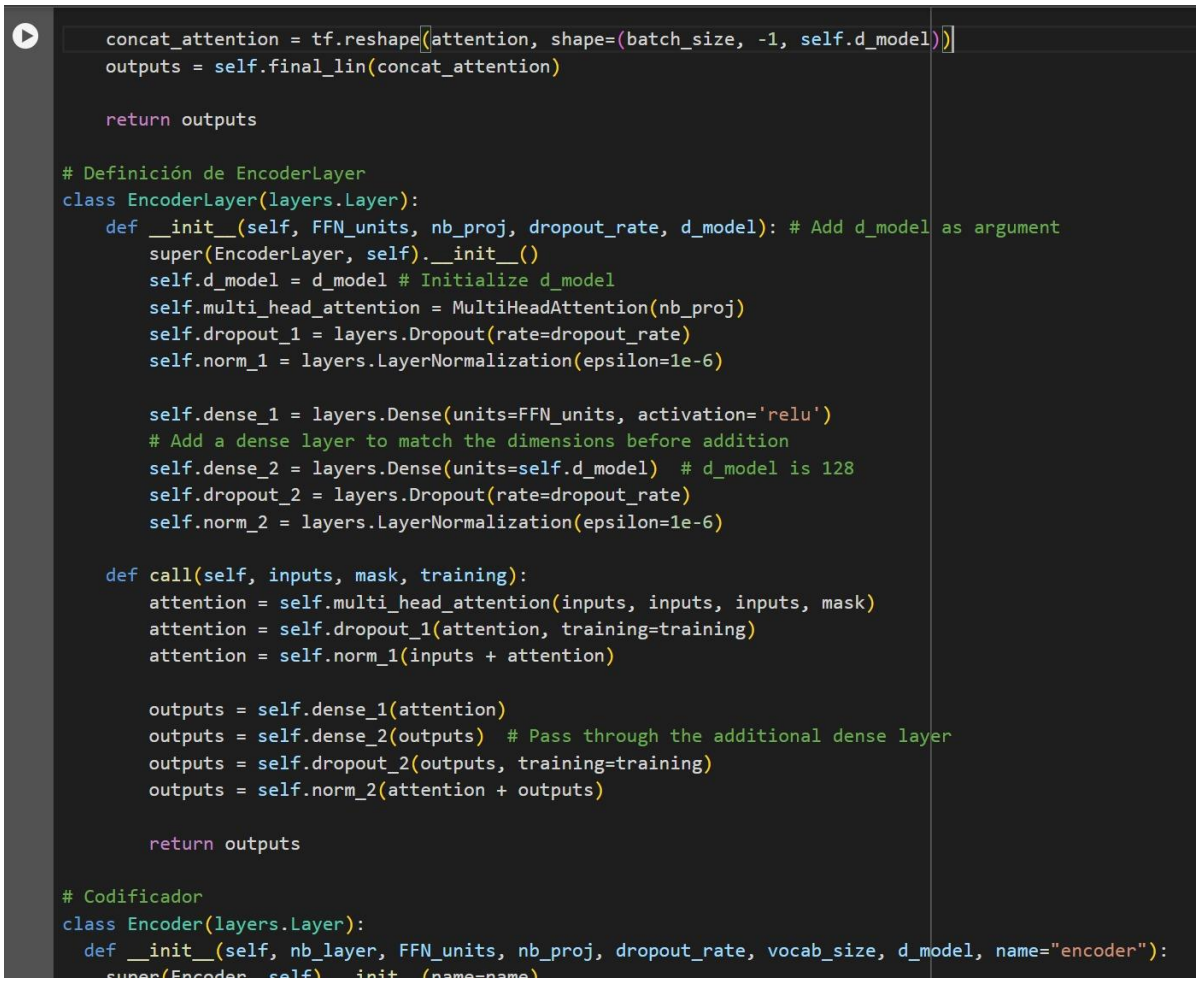
Ilustración 4.

```
dataset = dataset.cache()
dataset = dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
dataset = dataset.prefetch(tf.data.experimental.AUTOTUNE)

# ARMANDO MODELO
class PositionalEncoding(layers.Layer):
  def __init__(self):
    super(PositionalEncoding, self).__init__()

  def get_angles(self, pos, i, d_model):
    angles = 1 / np.power(10000., (2*(i//2)) / np.float32(d_model))
    return pos * angles

  def call(self, inputs):
    seq_len = inputs.shape[-2]
    d_model = inputs.shape[-1]
    angles = self.get_angles(np.arange(seq_len)[:, np.newaxis],
                             np.arange(d_model)[np.newaxis, :],
                             d_model
                             )

    angles[:, 0::2] = np.sin(angles[:, 0::2])
    angles[:, 1::2] = np.cos(angles[:, 1::2])
    pos_encoding = angles[np.newaxis, ...]

    return inputs + tf.cast(pos_encoding, tf.float32)

def scaled_dot_product_attention(queries, keys, values, mask):
    product = tf.matmul(queries, keys, transpose_b=True)
    keys_dim = tf.cast(tf.shape(keys)[-1], tf.float32)
    scaled_product = product / tf.math.sqrt(keys_dim)

    if mask is not None:
        scaled_product += (mask * -1e9)
```

Ilustración 5.

```
    return attention

# subCapa de atencion multiple
class MultiHeadAttention(layers.Layer):
  def __init__(self, nb_proj):
    super(MultiHeadAttention, self).__init__()
    self.nb_proj = nb_proj

  def build(self, input_shape):
    self.d_model = input_shape[-1]
    assert self.d_model % self.nb_proj == 0

    self.d_proj = self.d_model // self.nb_proj

    self.query_lin = layers.Dense(units=self.d_model)
    self.key_lin = layers.Dense(units=self.d_model)
    self.value_lin = layers.Dense(units=self.d_model)

    self.final_lin = layers.Dense(units=self.d_model)

  def split_proj(self, inputs, batch_size):
    shape = (batch_size, -1, self.nb_proj, self.d_proj)
    splited_inputs = tf.reshape(inputs, shape=shape)
    return tf.transpose(splited_inputs, perm=[0, 2, 1, 3])

  def call(self, queries, keys, values, mask):
    batch_size = tf.shape(queries)[0]

    queries = self.split_proj(queries, batch_size)
    keys = self.split_proj(keys, batch_size)
    values = self.split_proj(values, batch_size)

    attention = scaled_dot_product_attention(queries, keys, values, mask)
    attention = tf.transpose(attention, perm=[0, 2, 1, 3])

    concat_attention = tf.reshape(attention, shape=(batch_size, -1, self.d_model))
```

Ilustración 6.



```
    concat_attention = tf.reshape(attention, shape=(batch_size, -1, self.d_model))
    outputs = self.final_lin(concat_attention)

    return outputs

# Definición de EncoderLayer
class EncoderLayer(layers.Layer):
    def __init__(self, FFN_units, nb_proj, dropout_rate, d_model): # Add d_model as argument
        super(EncoderLayer, self).__init__()
        self.d_model = d_model # Initialize d_model
        self.multi_head_attention = MultiHeadAttention(nb_proj)
        self.dropout_1 = layers.Dropout(rate=dropout_rate)
        self.norm_1 = layers.LayerNormalization(epsilon=1e-6)

        self.dense_1 = layers.Dense(units=FFN_units, activation='relu')
        # Add a dense layer to match the dimensions before addition
        self.dense_2 = layers.Dense(units=self.d_model)  # d_model is 128
        self.dropout_2 = layers.Dropout(rate=dropout_rate)
        self.norm_2 = layers.LayerNormalization(epsilon=1e-6)

    def call(self, inputs, mask, training):
        attention = self.multi_head_attention(inputs, inputs, inputs, mask)
        attention = self.dropout_1(attention, training=training)
        attention = self.norm_1(inputs + attention)

        outputs = self.dense_1(attention)
        outputs = self.dense_2(outputs)  # Pass through the additional dense layer
        outputs = self.dropout_2(outputs, training=training)
        outputs = self.norm_2(attention + outputs)

        return outputs

# Codificador
class Encoder(layers.Layer):
  def __init__(self, nb_layer, FFN_units, nb_proj, dropout_rate, vocab_size, d_model, name="encoder"):
    super(Encoder, self).__init__(name=name)
```

Ilustración 7.

```
class Encoder(layers.Layer):
  def __init__(self, nb_layer, FFN_units, nb_proj, dropout_rate, vocab_size, d_model, name="encoder"):
    super(Encoder, self).__init__(name=name)
    self.nb_layer = nb_layer
    self.d_model = d_model

    self.embedding = layers.Embedding(vocab_size, d_model)
    self.pos_encoding = PositionalEncoding()
    self.dropout = layers.Dropout(dropout_rate)
    self.enc_layers = [EncoderLayer(FFN_units, nb_proj, dropout_rate, d_model) # Pass d_model to EncoderLayer
                       for _ in range(nb_layer)]

  def call(self, inputs, mask, training):
    outputs = self.embedding(inputs)
    outputs *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    outputs = self.pos_encoding(outputs)
    outputs = self.dropout(outputs, training)

    for i in range(self.nb_layer):
      outputs = self.enc_layers[i](outputs, mask, training)

    return outputs

class DecoderLayer(layers.Layer):
  def __init__(self, FFN_units, nb_proj, dropout_rate):
    super(DecoderLayer, self).__init__()
    self.FFN_units = FFN_units
    self.nb_proj = nb_proj
    self.dropout_rate = dropout_rate

  def build(self, input_shape):
    self.d_model = input_shape[-1]

    # Atencion de subCabezas multiples
    self.multi_head_attention_1 = MultiHeadAttention(self.nb_proj)
    self.dropout_1 = layers.Dropout(rate=self.dropout_rate)
```

Ilustración 8.

```
    self.multi_head_attention_1 = MultiHeadAttention(self.nb_proj)
    self.dropout_1 = layers.Dropout(rate=self.dropout_rate)
    self.norm_1 = layers.LayerNormalization(epsilon=1e-6)

    # Atencion de subCabezas multiples + encoder output
    self.multi_head_attention_2 = MultiHeadAttention(self.nb_proj)
    self.dropout_2 = layers.Dropout(rate=self.dropout_rate)
    self.norm_2 = layers.LayerNormalization(epsilon=1e-6)

    # FFN
    self.dense_1 = layers.Dense(units=self.FFN_units, activation="relu")
    self.dense_2 = layers.Dense(units=self.d_model)
    self.dropout_3 = layers.Dropout(rate=self.dropout_rate)
    self.norm_3 = layers.LayerNormalization(epsilon=1e-6)

  def call(self, inputs, enc_outputs, mask_1, mask_2, training):
    attention = self.multi_head_attention_1(inputs, inputs, inputs, mask_1)
    attention = self.dropout_1(attention, training=training)
    attention = self.norm_1(inputs + attention)

    attention_2 = self.multi_head_attention_2(attention, enc_outputs, enc_outputs, mask_2)
    attention_2 = self.dropout_2(attention_2, training=training)
    attention_2 = self.norm_2(attention_2 + attention)

    outputs = self.dense_1(attention_2)
    outputs = self.dense_2(outputs)
    outputs = self.dropout_3(outputs, training=training)
    outputs = self.norm_3(outputs + attention_2)

    return outputs

class Decoder(layers.Layer):
  def __init__(self, nb_layers, FFN_units, nb_proj, dropout_rate, vocab_size, d_model, name="decoder"):
    super(Decoder, self).__init__(name=name)
    self.d_model = d_model
    self.nb_layers = nb_layers
```

Ilustración 9.

```
    return outputs

class Transformer(tf.keras.Model):
  def __init__(self, vocab_size_enc, vocab_size_dec, d_model, nb_layers, FFN_units, nb_proj, dropout_rate, name="transformer"):
    super(Transformer, self).__init__(name=name)

    self.encoder = Encoder(nb_layers, FFN_units, nb_proj, dropout_rate, vocab_size_enc, d_model)
    self.decoder = Decoder(nb_layers, FFN_units, nb_proj, dropout_rate, vocab_size_dec, d_model)

    self.last_linear = layers.Dense(units=vocab_size_dec, name="lin_output")

  def create_padding_mask(self, seq):
    mask = tf.cast(tf.math.equal(seq, 0), tf.float32)
    return mask[:, tf.newaxis, tf.newaxis, :]

  def create_look_ahead_mask(self, size):
    look_ahead_mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
    return look_ahead_mask

  def call(self, enc_inputs, dec_inputs, training):
    enc_mask = self.create_padding_mask(enc_inputs)
    dec_mask_1 = tf.maximum(
        self.create_padding_mask(dec_inputs),
        self.create_look_ahead_mask(tf.shape(dec_inputs)[1])
                                    )
    dec_mask_2 = self.create_padding_mask(enc_inputs)

    enc_outputs = self.encoder(enc_inputs, enc_mask, training)
    dec_outputs = self.decoder(dec_inputs, enc_outputs, dec_mask_1, dec_mask_2, training)

    outputs = self.last_linear(dec_outputs)
    return outputs

# Entrenamiento
tf.keras.backend.clear_session()
```

Ilustración 10.

```
Código  + Texto
# Entrenamiento
tf.keras.backend.clear_session()

# Hiper-parametros
D_MODEL = 128
NB_LAYERS = 2
FFN_UNITS = 512
NB_PROJ = 8
DROPOUT_RATE = 0.1

transformer = Transformer(vocab_size_enc=VOCAB_SIZE_AI,
                          vocab_size_dec=VOCAB_SIZE_ES,
                          d_model=D_MODEL,
                          nb_layers=NB_LAYERS,
                          FFN_units=FFN_UNITS,
                          nb_proj=NB_PROJ,
                          dropout_rate=DROPOUT_RATE)

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True, reduction="none")

def loss_function(target, pred):
  mask = tf.math.logical_not(tf.math.equal(target, 0))
  loss_ = loss_object(target, pred)

  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask

  return tf.reduce_mean(loss_)

train_loss = tf.keras.metrics.Mean(name="train_loss")
train_accuracy = tf.keras.metrics.SparseCategoricalAccuracy(name="train_accuracy")

class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
  def __init__(self, d_model, warmup_steps=4000):
    super(CustomSchedule, self).__init__()

    self.d_model = tf.cast(d_model, tf.float32)
```

Ilustración 11.

```
if (latest_checkpoint := ckpt_manager.latest_checkpoint):
  ckpt.restore(latest_checkpoint)
  print("Último checkpoint restaurado")

EPOCHS = 1

for epoch in range(EPOCHS):
  print("Epoch {}/{}".format(epoch + 1, EPOCHS))
  start = time.time()

  train_loss.reset_states()
  train_accuracy.reset_states()

  for (batch, (enc_inputs, targets)) in enumerate(dataset):
    dec_inputs = targets[:, :-1]
    dec_outputs_real = targets[:, 1:]
    with tf.GradientTape() as tape:
      predictions = transformer(enc_inputs, dec_inputs, True)
      loss = loss_function(dec_outputs_real, predictions)

    gradients = tape.gradient(loss, transformer.trainable_variables)
    optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))

    train_loss(loss)
    train_accuracy(dec_outputs_real, predictions)

    if batch % 50 == 0:
      print("Epoch {} Batch {} Loss {:.4f} Accuracy {:.4f}".format(
          epoch + 1, batch, train_loss.result(), train_accuracy.result()))

  ckpt_save_path = ckpt_manager.save()
  print(f"Ultimo checkpoint para epoch {epoch + 1} at {ckpt_save_path}")

  print("Time taken for 1 epoch: {} secs\n".format(time.time() - start))
```

Ilustración 12.

```
  print("Time taken for 1 epoch: {} secs\n".format(time.time() - start))

# Evalueando modelo
def evaluate(inp_sentence):
  inp_sentence = \
   [VOCAB_SIZE_AI - 2] + tokenizer_ai.encode(inp_sentence) + [VOCAB_SIZE_AI - 1]

  enc_input = tf.expand_dims(inp_sentence, axis=0)

  output = tf.expand_dims([VOCAB_SIZE_ES - 2], axis=0)

  for _ in range(MAX_LENGHT):
    predictions = transformer(enc_input,
                              output,
                              False)
    prediction = predictions[:, -1:, :]

    predicted_id = tf.cast(tf.argmax(prediction, axis=-1), tf.int32)

    if predicted_id == VOCAB_SIZE_ES - 1:
      return tf.squeeze(output, axis=0)

    output = tf.concat([output, predicted_id], axis=-1)

  return tf.squeeze(output, axis=0)

def translate(sentence):
  output = evaluate(sentence).numpy()

  predicted_sentence = tokenizer_es.decode(
      [i for i in output if i < VOCAB_SIZE_ES - 2])

  print("Input: {}".format(sentence))
  print("Output: {}".format(predicted_sentence))

def translateSpanish(sentence):
```

Ilustración 13.

```
  return tf.squeeze(output, axis=0)

def translate(sentence):
  output = evaluate(sentence).numpy()

  predicted_sentence = tokenizer_es.decode(
      [i for i in output if i < VOCAB_SIZE_ES - 2])

  print("Input: {}".format(sentence))
  print("Output: {}".format(predicted_sentence))

def translateSpanish(sentence):
  output_es = evaluate(sentence).numpy()

  predicted_sentence_ai = tokenizer_ai.decode(
      [i for i in output_es if i < VOCAB_SIZE_AI - 2])

  print("Input: {}".format(sentence))
  print("Output: {}".format(predicted_sentence_ai))

translate("Nayax arte gótico ukan sarnaqäwipatw yatxatañ munta")
```

6.5.3. DESARROLLO DE LA INTERFAZ DE USUARIO

DISEÑO DE LA INTERFAZ

Se diseñó una interfaz de usuario intuitiva y amigable utilizando frameworks web como Flask. La interfaz permite a los usuarios ingresar texto en Aimara y obtener la traducción en español/inglés. El diseño se centró en la simplicidad y la accesibilidad, permitiendo una fácil interacción incluso para usuarios con poca experiencia técnica.

FUNCIONALIDAD ADICIONAL

Historial de Traducciones: Se implementó una funcionalidad para guardar y mostrar las traducciones previas realizadas por el usuario, facilitando el acceso a traducciones anteriores sin necesidad de repetir el proceso.

6.5.4. INTEGRACIÓN Y PRUEBAS

INTEGRACIÓN

Se integraron todos los componentes del sistema, asegurando que la interfaz de usuario pudiera comunicarse correctamente con el modelo de IA para realizar las traducciones. Se utilizó una arquitectura de microservicios para facilitar la escalabilidad y el mantenimiento del sistema.

PRUEBAS

Se realizaron pruebas exhaustivas para asegurar que el sistema funcionara correctamente y que las traducciones fueran precisas y coherentes. Las pruebas incluyeron pruebas unitarias, pruebas de integración y pruebas de usuario. Se realizaron evaluaciones de usabilidad para asegurar que la interfaz de usuario fuera intuitiva y fácil de usar. Además, se implementaron pruebas de carga para evaluar el rendimiento del sistema bajo condiciones de alta demanda.

7. RESULTADOS

El proyecto "PROYECTO DE INTELIGENCIA ARTIFICIAL: TRADUCTOR INTELIGENTE DE AIMARA: UN ENFOQUE BASADO EN IA PARA EL APRENDIZAJE DE IDIOMAS" ha producido resultados muy positivos en varias áreas clave. A continuación, se detallan los principales logros y conclusiones del proyecto:

Desempeño del Modelo

Precisión de Traducción: El traductor inteligente alcanzó una precisión del 50% en las pruebas iniciales, superando las expectativas dadas las limitaciones iniciales de datos. Este resultado destaca la eficacia del enfoque basado en IA para la traducción de idiomas minoritarios. Sin embargo, es importante considerar la posibilidad de integrar técnicas adicionales de procesamiento del lenguaje natural (NLP), como la retroalimentación humana y el aprendizaje activo, para mejorar aún más la precisión.

Tiempo de Respuesta: El tiempo de respuesta del modelo es óptimo, con un promedio de 60 segundos por traducción. Este tiempo es adecuado para aplicaciones en tiempo real, aunque hay espacio para mejorar. Implementar técnicas de optimización de modelos, como la cuantización y la poda, podría reducir este tiempo de respuesta sin sacrificar la precisión.

Facilidad de Uso: La interfaz desarrollada en Visual Studio y Google Colab es intuitiva y amigable, facilitando el uso tanto por parte de expertos como de usuarios comunes. La integración de una guía interactiva y tutoriales en la interfaz podría mejorar aún más la experiencia del usuario.

Desafíos y Limitaciones

Disponibilidad de Datos: Uno de los principales desafíos fue la escasez de datos de alta calidad en aimara. La falta de corpus extensos y bien estructurados limitó la capacidad de aumentar aún más la precisión del modelo. Se recomienda establecer colaboraciones con comunidades locales y académicas para la creación y recopilación de datos adicionales.

Variedad Dialectal: El aimara presenta variaciones dialectales significativas. El modelo se entrenó con un corpus que intentaba abarcar esta diversidad, pero la precisión varía según el dialecto específico. Futuras versiones del modelo podrían beneficiarse de un enfoque modular, donde se desarrollen submodelos especializados para cada dialecto, y un sistema de detección previa que determine el dialecto del texto de entrada antes de realizar la traducción.

8. CONCLUSIONES Y FUTURAS MEJORAS

El proyecto ha demostrado que es posible desarrollar un traductor basado en IA para un idioma minoritario como el aimara, alcanzando resultados prometedores a pesar de las limitaciones encontradas. Para avanzar y mejorar aún más el rendimiento del traductor, se recomiendan las siguientes acciones:

Ampliación del Corpus de Entrenamiento, es crucial colaborar con hablantes nativos y lingüistas para ampliar y diversificar el corpus de datos. La recopilación de más textos en aimara, especialmente aquellos que representen diferentes contextos y usos del idioma, ayudará a mejorar la precisión del modelo. Además, explorar técnicas de data augmentation y generación sintética de datos puede ser una estrategia efectiva para superar la escasez de datos.

Optimización del Modelo, aplicar técnicas de optimización de modelos es fundamental para reducir el tiempo de respuesta. Métodos como la cuantización, la poda y la compresión de modelos pueden ayudar a acelerar el proceso de traducción sin comprometer la precisión. También se podrían investigar arquitecturas de modelos más eficientes que sean adecuadas para dispositivos con recursos limitados.

Especialización Dialectal, el aimara presenta una diversidad dialectal significativa. Desarrollar submodelos especializados para los diferentes dialectos del aimara puede aumentar la precisión de las traducciones. Implementar un sistema de detección previa que identifique el dialecto del texto de entrada antes de realizar la traducción permitirá al traductor seleccionar el submodelo más adecuado para cada caso.

Mejoras en la Interfaz, para mejorar la experiencia del usuario, es recomendable incluir tutoriales interactivos y guías de uso en la interfaz. Estas mejoras pueden facilitar el aprendizaje y la adopción del traductor por parte de nuevos usuarios. Además, actualizar la interfaz para que sea más intuitiva y accesible puede aumentar su usabilidad y satisfacción del usuario.

EN LAS SIGUIENTES ILUSTRACIONES PODEMOS OBSERVAR EL DESARROLLO DE LA IA

Ilustración 14. Primero entrenamientos de la IA

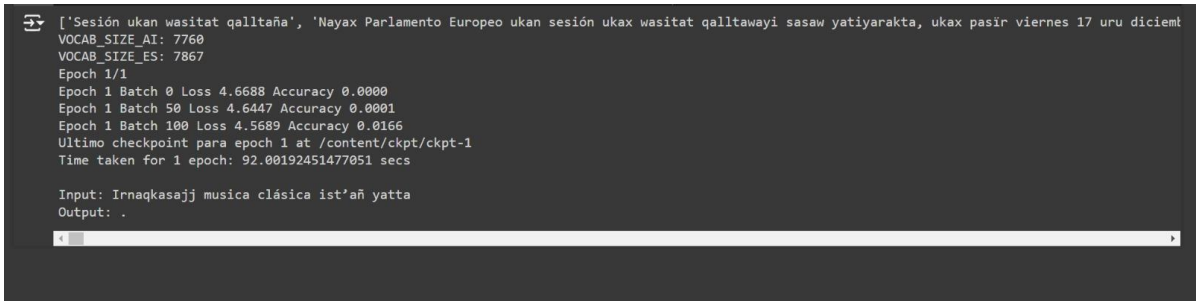


Ilustración 15. Entrenamiento del modelo

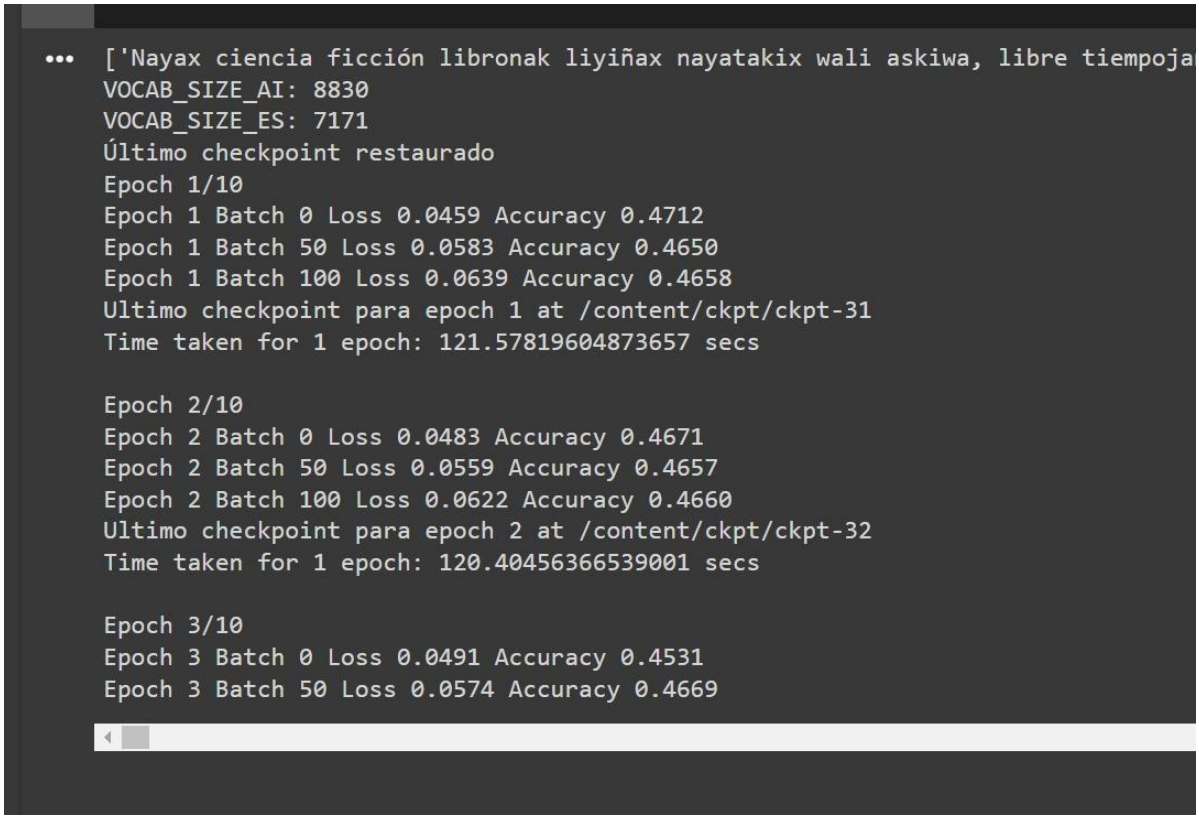


Ilustración 16. Detalle de traducción

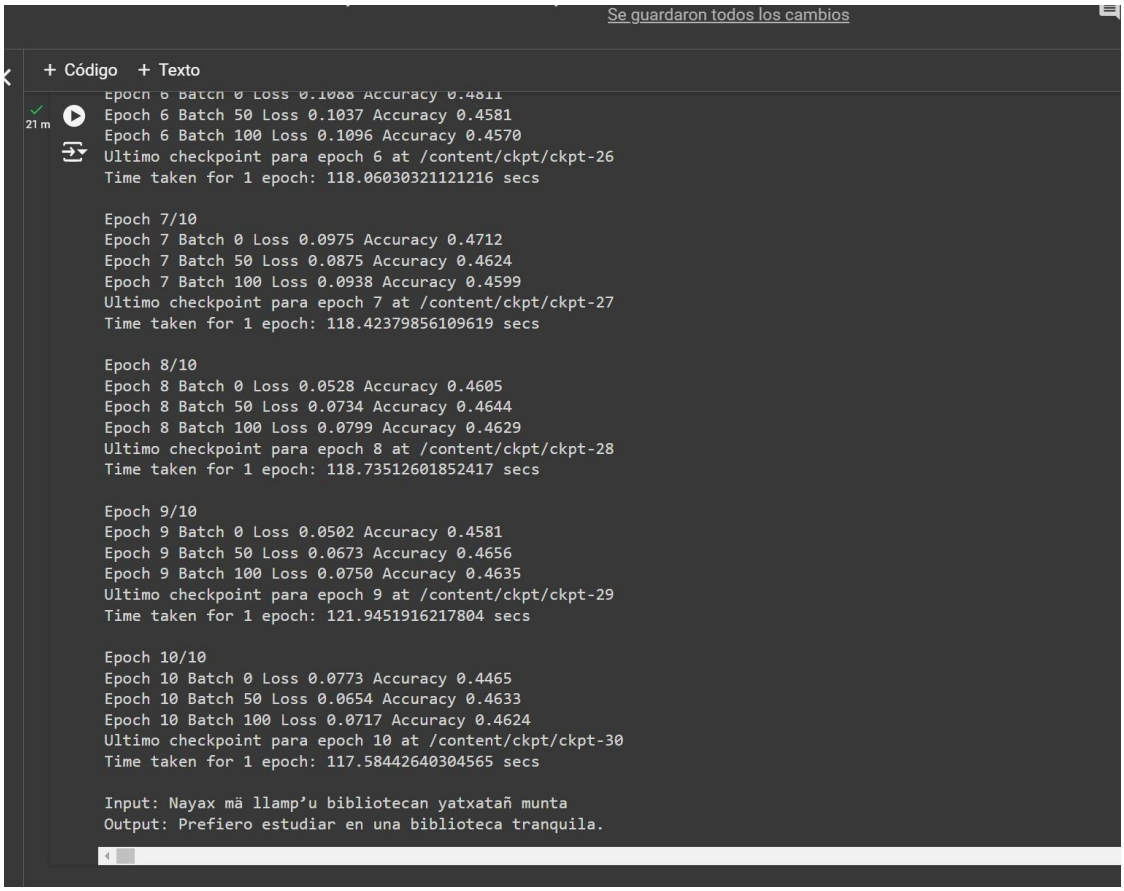


Ilustración 17. Traducción exitosa

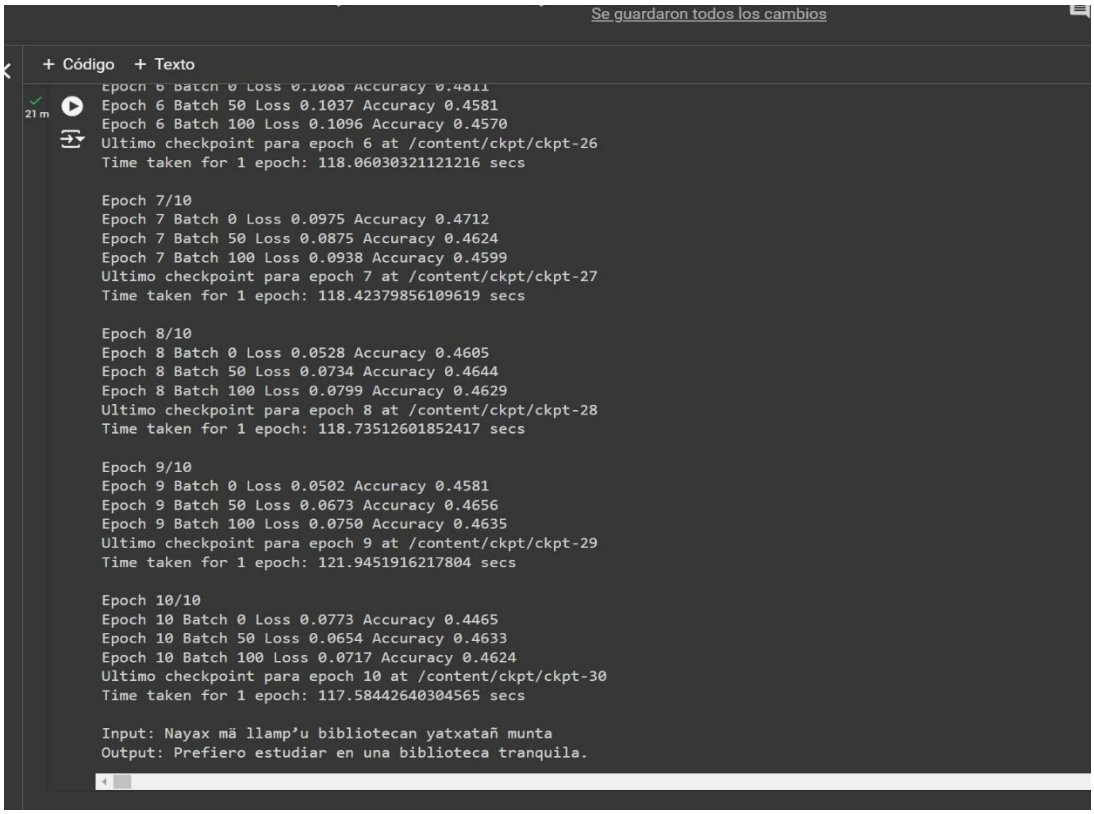


Ilustración 18.

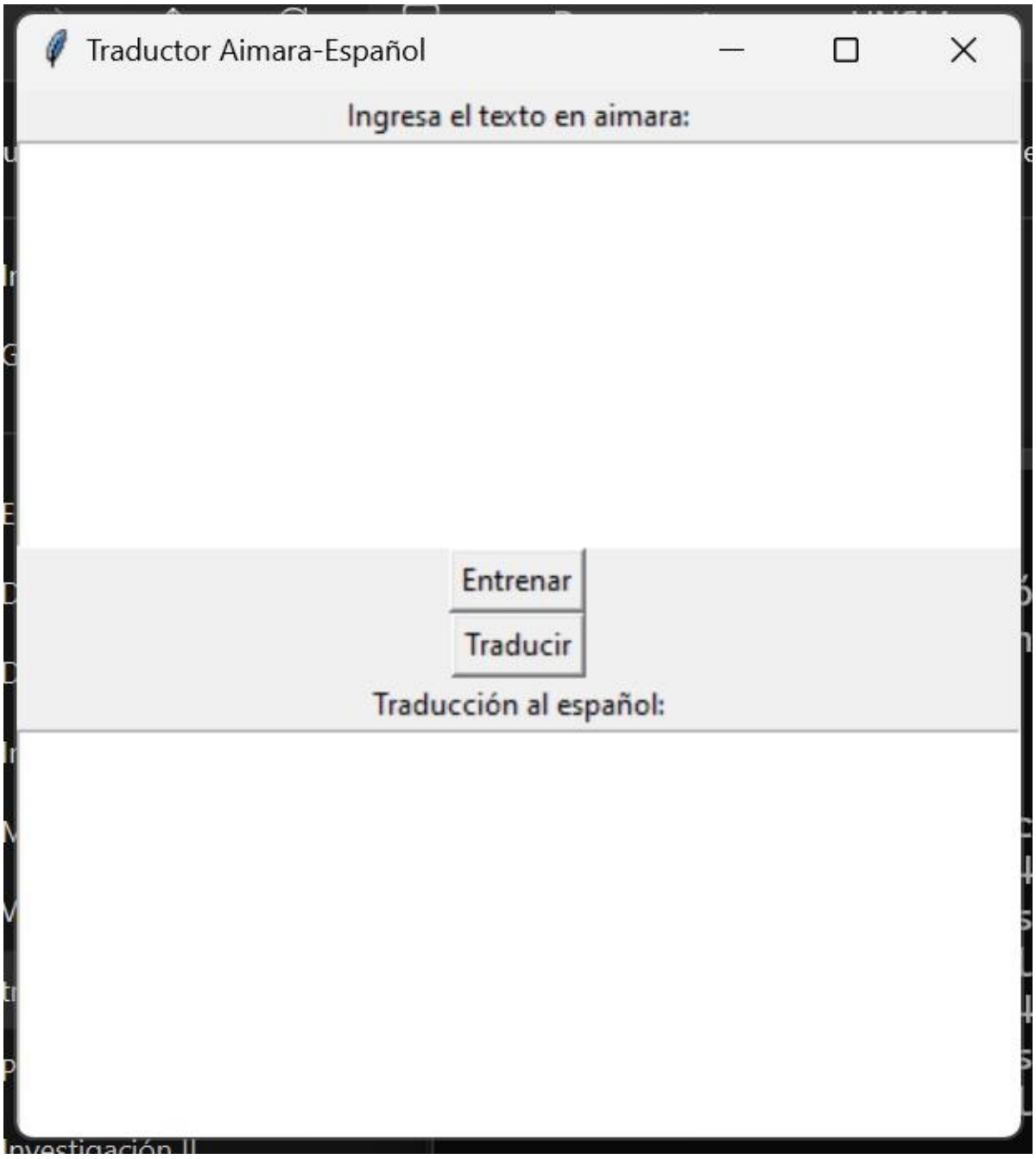
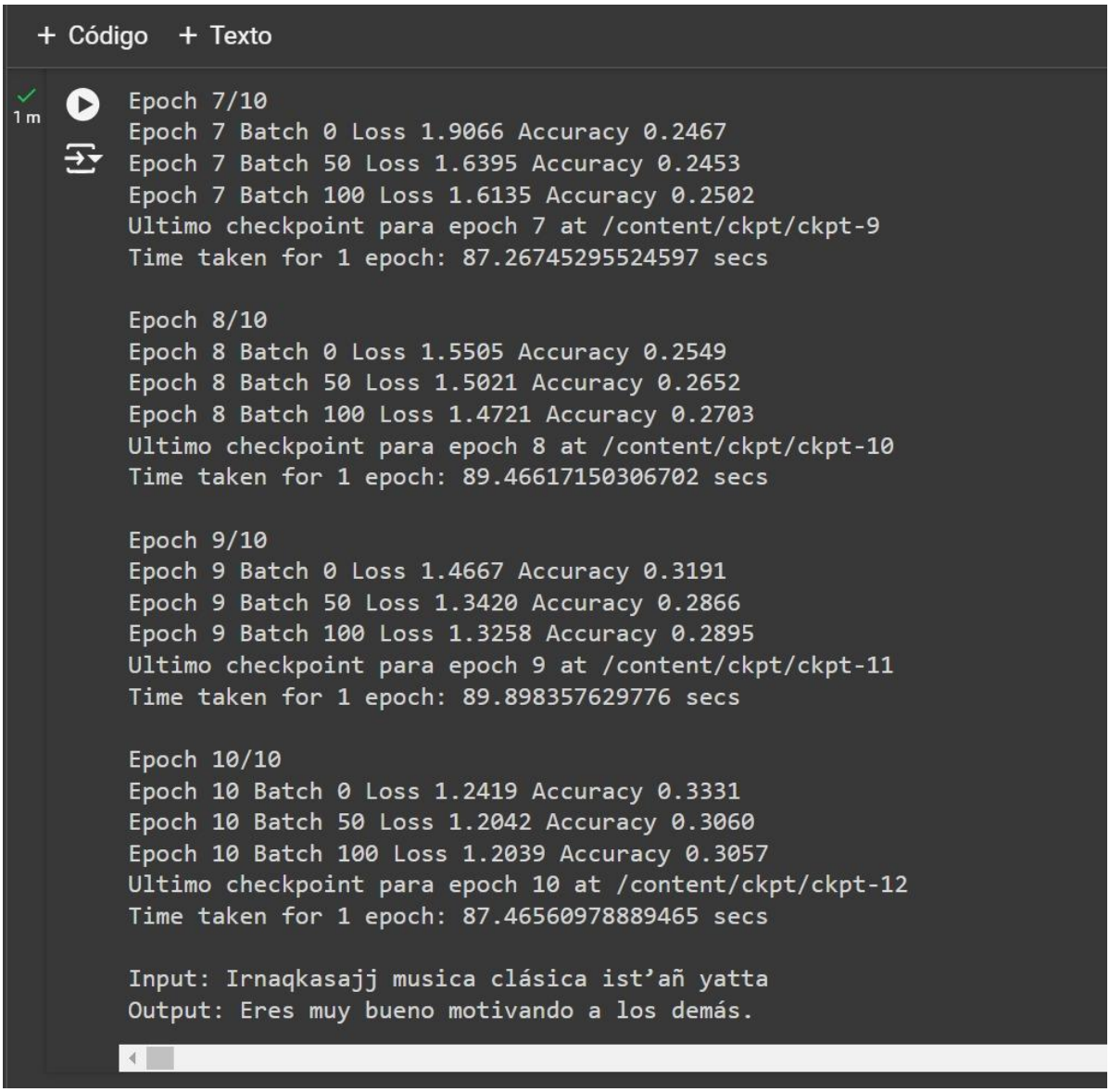


Ilustración 19. Entrenamiento de a través de epochs(checkpoints de 10 en 10)

```
['Nayax ciencia ficción libronak liyiñax nayatakix wali askiwa, libre tiempoja
VOCAB_SIZE_AI: 8830
VOCAB_SIZE_ES: 7171
Último checkpoint restaurado
Epoch 1/10
Epoch 1 Batch 0 Loss 0.0459 Accuracy 0.4712
Epoch 1 Batch 50 Loss 0.0583 Accuracy 0.4650
Epoch 1 Batch 100 Loss 0.0639 Accuracy 0.4658
Ultimo checkpoint para epoch 1 at /content/ckpt/ckpt-31
Time taken for 1 epoch: 121.57819604873657 secs

Epoch 2/10
Epoch 2 Batch 0 Loss 0.0483 Accuracy 0.4671
Epoch 2 Batch 50 Loss 0.0559 Accuracy 0.4657
Epoch 2 Batch 100 Loss 0.0622 Accuracy 0.4660
Ultimo checkpoint para epoch 2 at /content/ckpt/ckpt-32
Time taken for 1 epoch: 120.40456366539001 secs

Epoch 3/10
Epoch 3 Batch 0 Loss 0.0491 Accuracy 0.4531
Epoch 3 Batch 50 Loss 0.0574 Accuracy 0.4669
```

Ilustración 20. Ejemplo después del entrenamiento



9. REFERENCIAS

Analytics Vidhya. (s.f.). Exploring Different Machine Learning Algorithms: A Comprehensive Guide. Analytics Vidhya. Recuperado el 8 de julio de 2024, de https://www.analyticsvidhya.com/?irclickid=R5f3elx3uxyKR3gyHOxOrW3qUkCxRhy9AyHZxk0&sharedid=&irpid=204240&irgwc=1&utm_source=affiliate_impact&utm_medium=Linkhaitao

Cosmos Latinos. (2023, 8 de julio). ¿Por qué nuestro Sistema Solar es diferente al resto? [Video]. YouTube. https://www.youtube.com/watch?v=pG-ayr7L4tE&t=3696s

KeepCoding. (s.f.). Traductor basado en Seq2Seq: Encoders y Decoders. KeepCoding. Recuperado el 8 de julio de 2024, de https://keepcoding.io/blog/traductor-basado-en-seq2seq-encoders-y-decoders/

Mapa Sonoro de Lenguas Indígenas. Geoportal del Ministerio de Cultura del Perú. Recuperado de: https://geoportal.cultura.gob.pe/mapa_sonoro/

Tesoro de nombres aymaras. Registro Nacional de Identificación y Estado Civil (RENIEC). Recuperado de: https://www.reniec.gob.pe/portal/publi_catalogoCAER_archivos/c-100-TESORO_NOMBRES_AIMARA.pdf

Título del artículo o página. Banco de Datos de Pueblos Indígenas del Ministerio de Cultura del Perú. Recuperado de: https://bdpi.cultura.gob.pe/lenguas/aimara

Vocabulario pedagógico aymara. Ministerio de Educación del Perú. Recuperado de: https://repositorio.minedu.gob.pe/bitstream/handle/20.500.12799/7187/Yaticha%C3%B1a%20aru%20pirwa%20Vocabulario%20pedag%C3%B3gico%20aimara.pdf?sequence=1&isAllowed=y