Universidad del Valle de Guatemala
Data Science
Sec. 10

Proyecto 2

Fase Final

Javier Chavez 21016

Andres Quezada 21085

Javier Ramirez 21600

Mario Cristales 21631

Situación problemática

En la actualidad, las personas sordas o con dificultades auditivas enfrentan una gran barrera en el acceso a muchas tecnologías de reconocimiento de voz y asistentes virtuales. Mientras que tecnologías como el reconocimiento automático de voz y la traducción automática han revolucionado la manera en que las personas interactúan con dispositivos, estas soluciones no son accesibles para los más de 70 millones de personas sordas en el mundo que utilizan lenguaje de señas para comunicarse, ni para los 1.5 mil millones de personas que experimentan pérdida auditiva.

Un aspecto fundamental de la comunicación en lenguaje de señas es el "deletreo manual", que se utiliza para transmitir palabras letra por letra, como nombres, direcciones o números de teléfono. Sin embargo, la inteligencia artificial orientada al reconocimiento de lenguaje de señas se encuentra significativamente atrasada en comparación con otras tecnologías como la conversión de voz a texto o el uso de teclados virtuales.

Esto plantea un problema crítico: la falta de herramientas tecnológicas adecuadas para el reconocimiento y traducción del lenguaje de señas impide que muchas personas sordas o con dificultades auditivas se beneficien de los avances en inteligencia artificial. La ausencia de datasets robustos y modelos entrenados ha sido uno de los principales obstáculos para el desarrollo de soluciones que puedan interpretar correctamente los gestos y convertirlos en texto o en voz.

Problema científico

El problema científico que se aborda en este proyecto es el desarrollo de un modelo de inteligencia artificial capaz de reconocer y traducir de manera precisa el deletreo manual del lenguaje de señas americano a texto, a partir de datos capturados en diversas condiciones de fondo e iluminación. Este problema surge debido a la ausencia de datasets lo suficientemente amplios y modelos entrenados que permitan una interpretación eficaz de los gestos del ASL, lo que limita el acceso de las personas sordas o con dificultades auditivas a tecnologías avanzadas de interacción con dispositivos, como lo son el reconocimiento de voz y asistentes virtuales.

Objetivos

Objetivo General:

Desarrollar un modelo de inteligencia artificial que reconozca y traduzca de manera precisa el deletreo manual del lenguaje de señas americano a texto.

Objetivos específicos:

- Entrenar un modelo de aprendizaje profundo utilizando el conjunto de datos proporcionado, con una precisión mínima del 80% en la traducción de las letras del ASL a texto.

- Optimizar el tiempo de inferencia del modelo para que pueda traducir cada gesto en menos de 300 milisegundos.

- Implementar técnicas de preprocesamiento y aumentación de datos para mejorar la robustez del modelo ante variaciones en las posiciones de las manos y condiciones ambientales, midiendo su impacto en la precisión del modelo final.

Descripción de los datos

El conjunto de datos está dividido en los siguientes archivos:

- Landmarks.parquet
- train.csv
- supplemental_metadata.csv
- character_to_prediction_index.json

Ambos acompañados por archivos de coordenadas en formato Parquet que contienen la información clave de los puntos de referencia que describen las posiciones de las manos, el rostro y el cuerpo de los participantes.

También se incluye el archivo character_to_prediction_index.json, que mapea cada carácter del alfabeto a un índice numérico.

Archivos Parquet (train_landmarks/ y supplemental_landmarks/):

Columnas:

- sequence_id: Identificador único para la secuencia de landmarks.
- frame: Número de fotograma dentro de la secuencia.
- x/y/z_[type]_[landmark_index]: Coordenadas espaciales (x, y, z) para cada landmark, donde type puede ser 'face', 'left_hand', 'right_hand', o 'pose', y landmark_index es el índice del punto específico dentro de cada tipo.

Los archivos de landmarks contienen datos espaciales extraídos de videos utilizando el modelo MediaPipe Holistic. Estos landmarks representan 543 puntos de referencia en las manos, el rostro y el cuerpo de los participantes.

Archivos CSV (train.csv y supplemental_metadata.csv):

Columnas:

- path: Contiene la ruta al archivo Parquet que almacena los datos de los landmarks (posiciones de las manos y otras partes del cuerpo) para una secuencia de movimientos de manos.
- file_id: Identificador único para cada archivo de datos.
- participant_id: Identificador único para cada participante, representando la persona que realizó los gestos de ASL.
- sequence_id: Identificador único para cada secuencia de gestos dentro de un archivo. Un archivo puede contener múltiples secuencias.

- phrase: Frase representada por la secuencia de signos. Estas frases incluyen direcciones, números de teléfono, URLs, entre otros, generados de manera aleatoria.

Estos archivos proporcionan el contexto necesario para entender qué frase se está representando en cada secuencia de datos, permitiendo correlacionar los movimientos de los landmarks con la frase correspondiente.

Archivo character_to_prediction_index.json:

Este archivo asocia cada carácter del alfabeto, junto con algunos símbolos, a un índice numérico único. Es crucial para convertir las secuencias de gestos de manos en predicciones numéricas que luego se pueden mapear a texto.

Operaciones de Limpieza de datos:

Para mejorar la calidad de los datos y prepararlos para el análisis, se llevaron a cabo las siguientes operaciones de limpieza:

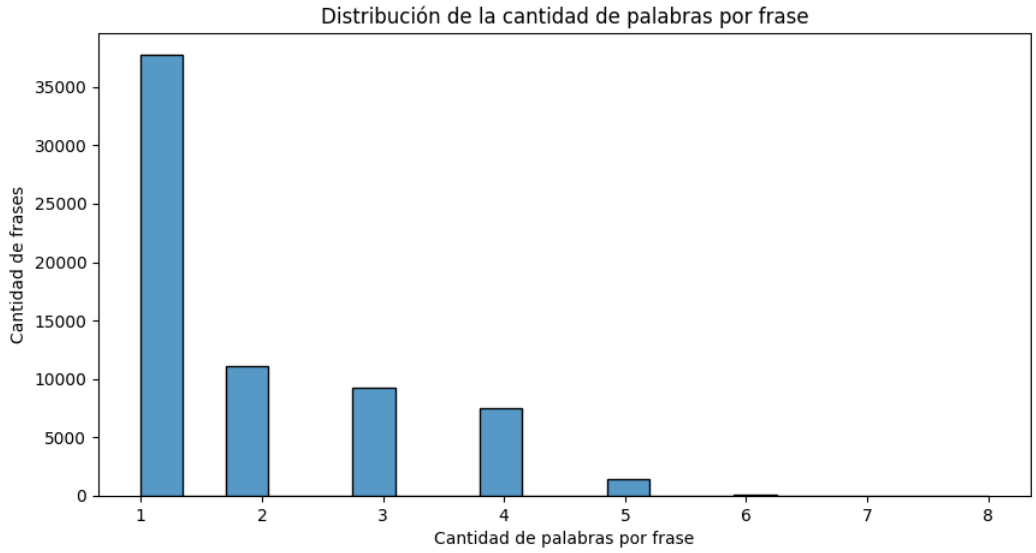
- Detección de valores faltantes (NaN): Se identificaron columnas con valores faltantes, particularmente en las coordenadas asociadas a las manos y el rostro.
- Eliminación de columnas con más del 50% de valores faltantes: Se eliminan las columnas con más de la mitad de sus valores faltantes para evitar información ruidosa o irrelevante.
- Interpolación: Para las columnas restantes con valores faltantes, se aplicó una interpolación lineal para llenar los huecos en los datos. Además, se usó el método bfill (backward fill) y ffill (forward fill) para imputar cualquier valor faltante restante, utilizando el valor anterior o siguiente más cercano.
- Preservación de columnas clave: Se aseguraron de que las columnas relacionadas con la mano derecha no se eliminaran, dado que son críticas para el análisis de gestos manuales.

Análisis Exploratorio

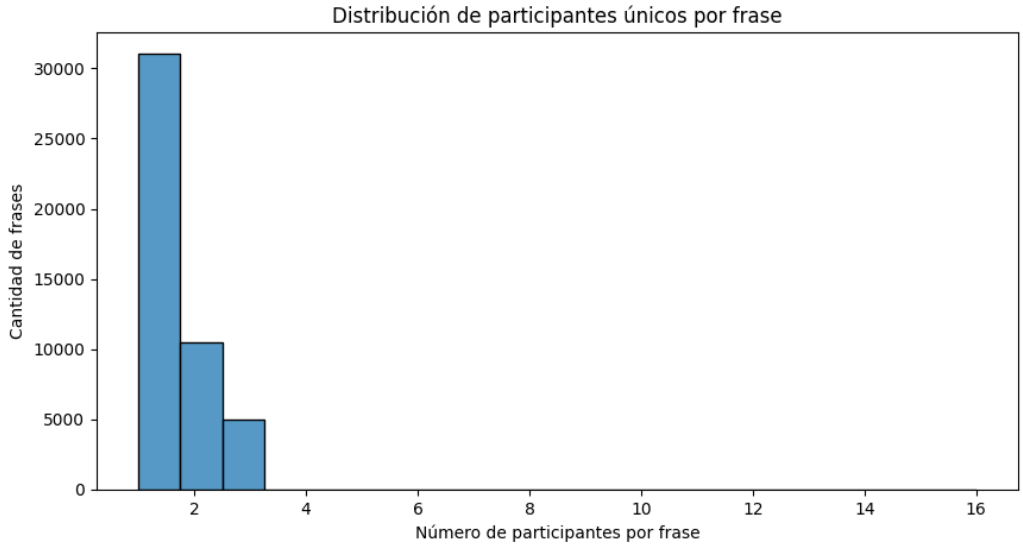
El conjunto de datos adjunto contiene las siguientes características:

- Cantidad total de participantes: 94

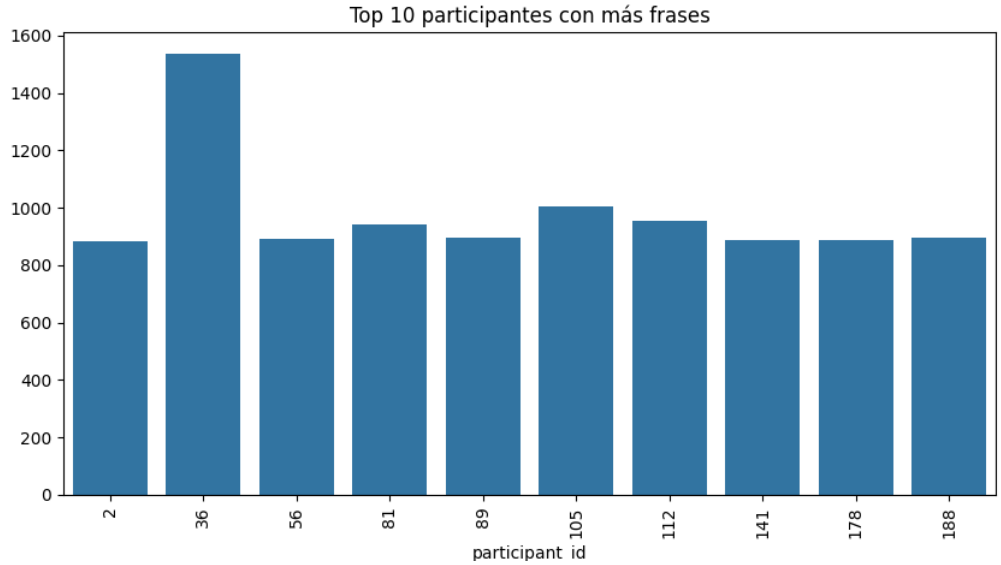
El promedio de palabras por frase es 1.87, la gran mayoría de las frases son de 1 palabra, pero llegan a hasta 8 palabras por frase, conforme a la siguiente distribución:



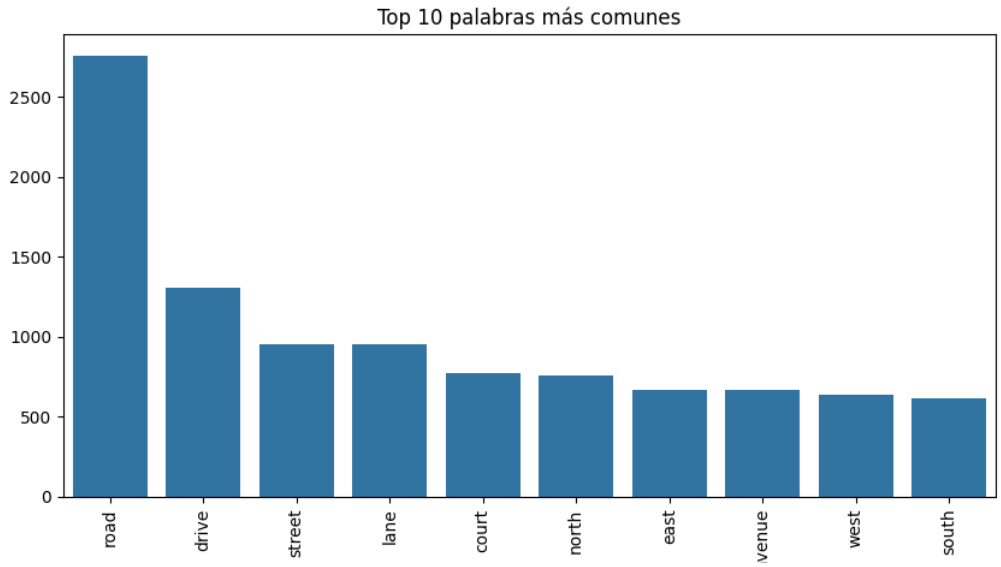
La mayoría de las frases es elaborada solamente por 1 participante, pero existen otras dónde 2 o 3 participantes repiten la misma frase.



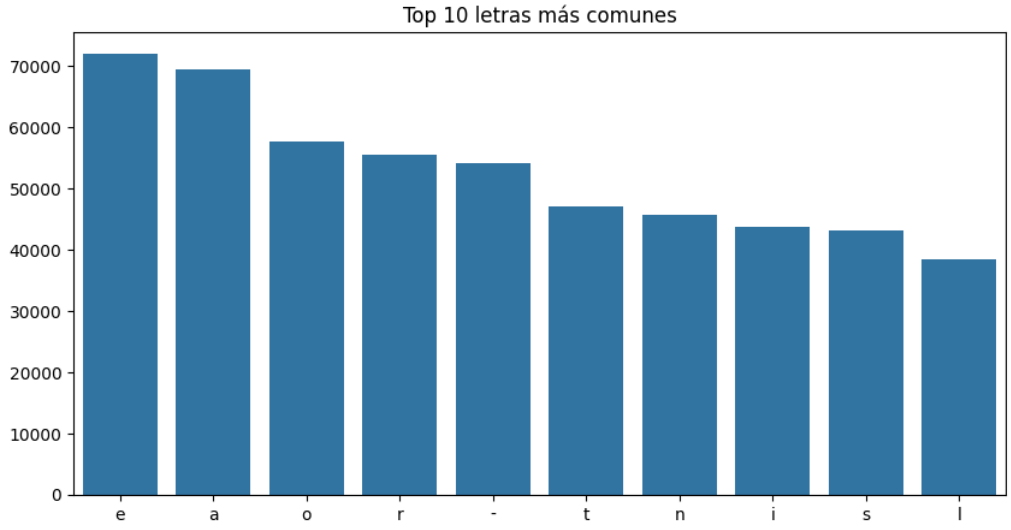
Cada participante realizó en promedio 714 frases, el participante con más frases es el ID 36 con 1535 frases. Los 10 participantes con más frases aparecen en la siguiente gráfica.



Dentro de todo el conjunto de datos la palabra más común es “road”, las 10 palabras más frecuentes se pueden observar en el siguiente gráfico.



El carácter más común es la letra “e”, los 10 caracteres más frecuentes se pueden observar en el siguiente gráfico.



La variación en la cantidad de palabras por frase es un dato importante. Por lo que no hay outliers graves, las frases con más de 5 palabras son escasas y podrían considerarse atípicas en este conjunto de datos predominantemente corto.

Los datos de los .parquet ya son output de un modelo de análisis de videos por lo que ya están normalizados y estructurados completamente.

Hallazgos

El conjunto de datos contiene un total de 46,478 frases únicas, lo que proporciona una gran diversidad de muestras para análisis y entrenamiento.

El promedio de palabras por frase es de 1.87, lo que indica que la mayoría de las frases son bastante cortas, lo cual es consistente con la naturaleza de las frases usadas en la comunicación por deletreo manual en ASL.

Frases más comunes:

- Las palabras más repetidas están relacionadas con nombres de calles y direcciones (por ejemplo, "road", "drive", "street"), lo que refuerza la naturaleza de los datos como una representación de frases prácticas.

Participación:

- Hay un total de 254 participantes diferentes en el conjunto de datos, siendo el participante con más frases el identificado con el ID 36, quien contribuyó con 1,535 frases.

- El promedio de frases por participante es de 714.98, lo que refleja una distribución relativamente uniforme de la cantidad de frases entre los diferentes participantes.

Letras y palabras más comunes:

- La letra más común en las frases es la "e", seguida de "a", "o", y "r". Esto refleja una distribución de letras similar a la del idioma inglés, lo cual es un hallazgo importante para optimizar el reconocimiento de gestos en el modelo.

- La palabra más repetida es "road", con 2,754 apariciones, lo que sugiere que las direcciones y ubicaciones son elementos clave en las frases utilizadas en el conjunto de datos.

Distribución de la cantidad de palabras por frase:

- La mayoría de las frases tienen una sola palabra, y el número de frases disminuye a medida que el número de palabras aumenta. Esto sugiere que el modelo debe estar optimizado para reconocer frases cortas con una alta frecuencia de aparición.

Distribución de participantes por frase:

- La mayoría de las frases fueron realizadas por uno o dos participantes. Pocas frases fueron repetidas por más de dos personas, lo que sugiere que el conjunto de datos contiene una gran diversidad de frases únicas realizadas por diferentes personas.

Conclusiones

- Optimización del modelo para frases cortas: Dado que la mayoría de las frases contienen una o dos palabras, los modelos de reconocimiento de gestos deben ser optimizados para procesar frases cortas de manera eficiente. Esto implica que el entrenamiento debe enfocarse en identificar rápidamente los gestos que corresponden a palabras individuales y nombres propios.

- Foco en letras comunes: La alta frecuencia de letras como "e", "a", "o", y "r" sugiere que el reconocimiento de estas letras debe ser especialmente preciso. El sistema debe ser capaz de detectar con alta precisión estas letras debido a su alta prevalencia en el conjunto de datos.

- Variabilidad entre participantes: Aunque algunos participantes tienen más frases que otros, el número de participantes únicos por frase es bajo, lo que significa que las frases suelen ser realizadas por un solo individuo. Esto sugiere que es importante garantizar que el modelo sea robusto frente a la variabilidad entre los diferentes participantes.

Selección de Algoritmos

#1 Transformer

- Descripción

  Los modelos de transformadores son arquitecturas de aprendizaje profundo diseñadas para manejar datos secuenciales, como texto o secuencias de gestos, mediante el uso de mecanismos de atención que destacan relaciones contextuales en la secuencia. Son especialmente eficaces en tareas que requieren captar dependencias a largo plazo en los datos.

- Ajuste a nuestros datos y objetivo

  Dado que los gestos en lenguaje de señas representan secuencias temporales complejas, el modelo transformer puede capturar patrones y relaciones temporales entre frames de video o landmarks, lo cual se ajusta bien al objetivo de reconocimiento de señas, donde el contexto y la secuencia son cruciales para identificar correctamente cada gesto.

#2 Seq2Seq

- Descripción

  El modelo Seq2Seq (Sequence-to-Sequence) es una arquitectura de redes neuronales diseñada para transformar una secuencia de entrada en una secuencia de salida, comúnmente utilizada en tareas de procesamiento de lenguaje natural (PLN) como traducción automática, resumen de texto, generación de respuestas, entre otros.

- Ajuste a nuestros datos y objetivo

  Para el reconocimiento de señas, el modelo Seq2Seq es adecuado ya que permite interpretar secuencias de movimientos o posiciones de manos (entrada) y generar la secuencia correspondiente en lenguaje de señas (salida), capturando dependencias temporales en los datos y ajustándose al contexto específico de cada gesto en la secuencia.

#3 CTC-based RNN (Recurrent Neural Networks with Connectionist Temporal Classification)

- Descripción

  CTC (Connectionist Temporal Classification) es una técnica comúnmente utilizada en tareas de reconocimiento de secuencias donde la alineación entre la secuencia de entrada y la salida no está predeterminada. Usualmente, los modelos CTC se combinan con redes neuronales recurrentes (RNNs), como LSTMs o GRUs, que son capaces de procesar secuencias de datos temporales. CTC ayuda a manejar la variabilidad en la longitud de las secuencias y permite que el modelo prediga secuencias con longitudes variables sin la necesidad de una alineación precisa entre la entrada y la salida.

- Ajuste a nuestros datos y objetivos

  El modelo basado en CTC es ideal para nuestro problema de reconocimiento de señas, donde las secuencias de movimientos de manos pueden variar en longitud, y no siempre existe una alineación clara entre la entrada (gestos o landmarks) y la salida (frase o palabra correspondiente). Al utilizar CTC con una red neuronal recurrente, podemos procesar los datos de gestos como una secuencia temporal y predecir la secuencia correspondiente en lenguaje de señas sin requerir una alineación previa exacta. Además, CTC es excelente para tareas donde las entradas y salidas tienen longitudes diferentes, como en nuestro caso.

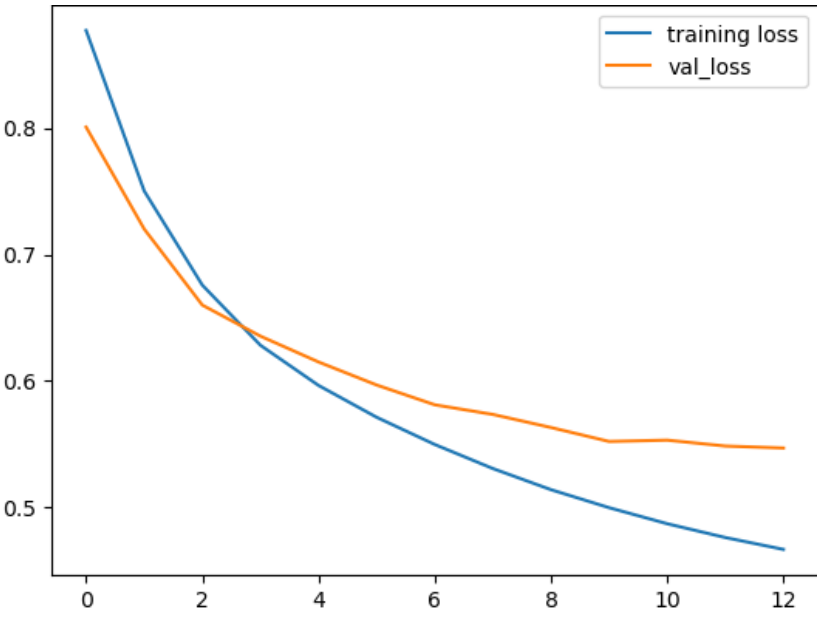
Implementación

https://github.com/JaniMariQuesiRami/DS-PR2

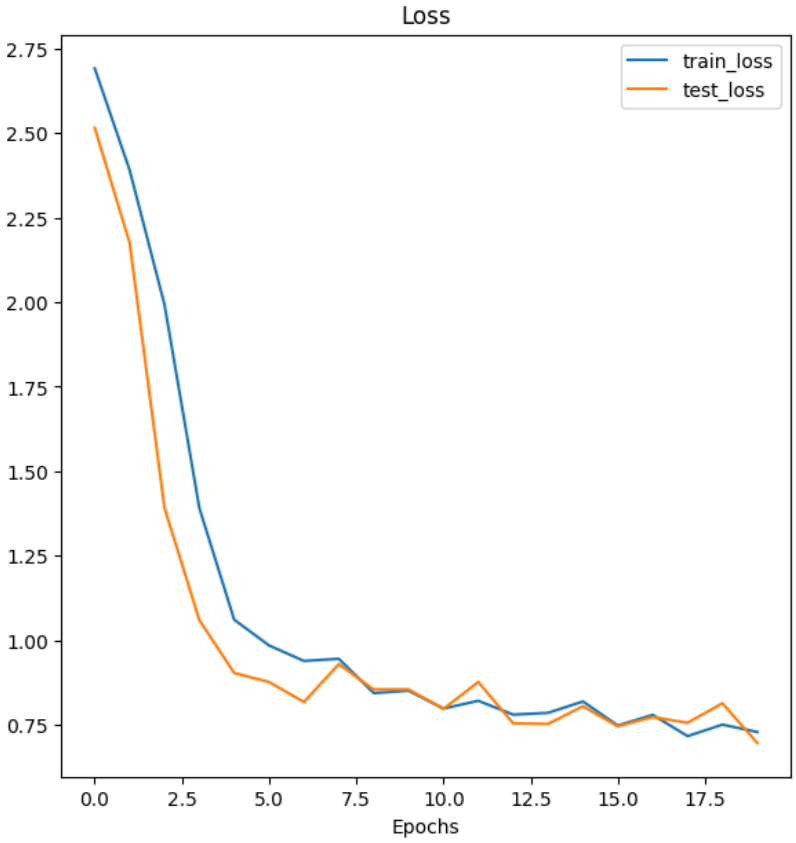
Se pueden encontrar los modelos en la carpeta: /fase2/modelos

Evaluación de modelos

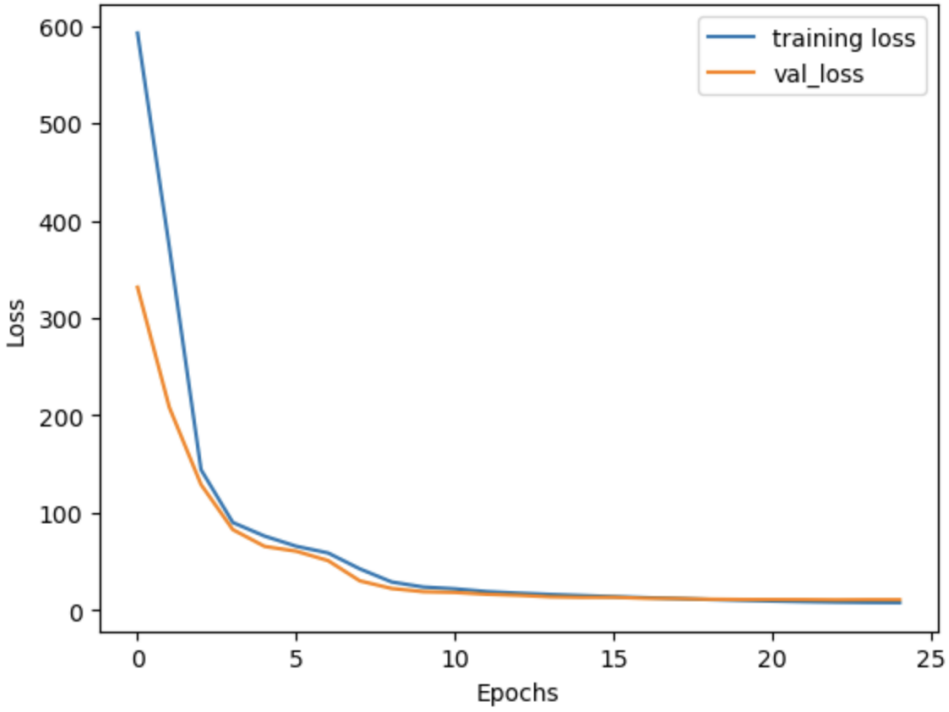
Modelo Transformer (Categorical Crossentropy)



Modelo Seq2Seq (Categorical Crossentropy)



Modelo CTC (CTCLoss)



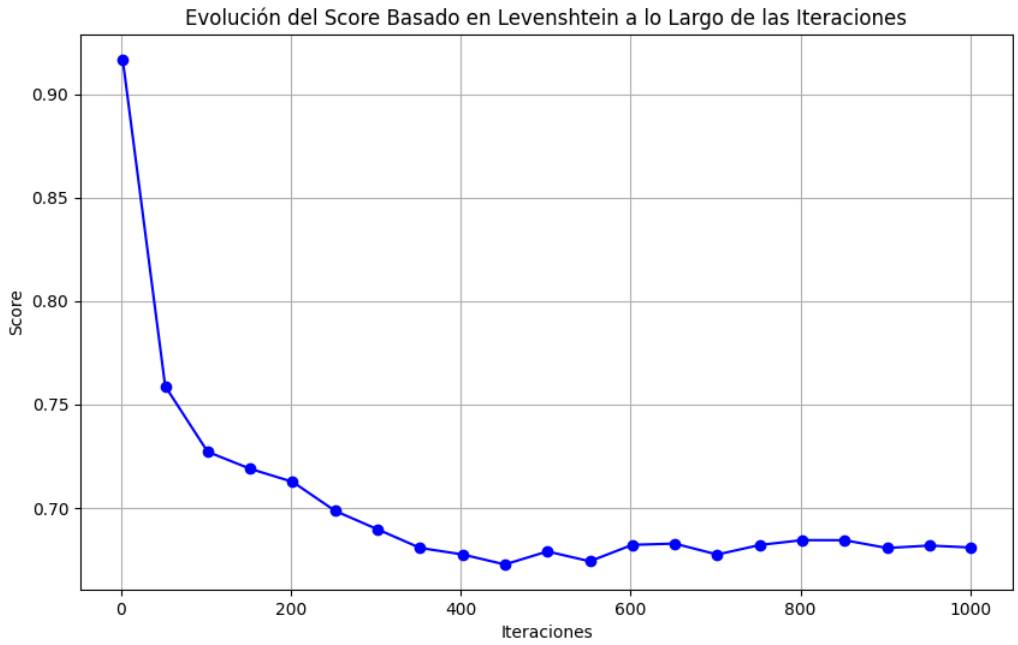
Luego de ver los correspondientes outputs, decidimos que el modelo CTC es el que otorga mejores resultados. Entonces decidimos calcular un score basado en la distancia Levenshtein. Siguiendo la siguiente fórmula:

Este cálculo produce un valor de precisión que indica qué tan bien coincide la predicción con la frase objetivo:

- Score = 1: Significa una coincidencia perfecta entre la predicción y la frase objetivo, sin errores de edición.

- Score < 1: Indica una falta de coincidencia, cuanto menor es el puntaje, más errores de edición son necesarios para hacer coincidir la predicción con la frase objetivo.

- Score = 0: Esto ocurriría si la predicción es completamente diferente de la frase objetivo, con una distancia de edición igual a la longitud de la frase objetivo.



Y en esta gráfica observamos el score a lo largo de 1000 iteraciones del grupo de datos de validación, como se puede observar, el score a lo largo de las iteraciones decrece desde 0.90 a aproximadamente 0.68 en promedio. Esto indica que el modelo tiene una menor coincidencia con las secuencias objetivo en las predicciones hacia el final de las iteraciones.

La reducción en el score podría sugerir que, aunque el modelo está ajustándose a los datos de entrenamiento, se está enfrentando a desafíos al generalizar a los datos de validación, o bien podría ser una señal de que aún necesita ajustes para mantener un rendimiento consistente. Una posible explicación de este comportamiento podría ser que el modelo se adapta inicialmente bien a patrones comunes, pero encuentra dificultades con secuencias más complejas o menos frecuentes, lo que reduce el score promedio conforme procesa más ejemplos.

Comparación entre modelos

Como se puede observar en las gráficas de las funciones de pérdida, los primeros dos modelos, el Transformer y el Seq2Seq, muestran una tendencia decreciente en la pérdida a lo largo de las épocas, lo cual indica que ambos están aprendiendo y mejorando su ajuste a los datos de entrenamiento. Sin embargo, la pendiente de reducción de la pérdida es diferente en cada modelo. Mientras que el modelo Transformer tiene una disminución suave y gradual, el modelo Seq2Seq muestra una reducción rápida en las primeras épocas, seguida de una estabilización en valores más bajos.

El modelo Seq2Seq comienza con una pérdida inicial mucho más alta (~2.75) en comparación con el Transformer (~0.9). Esto sugiere que el Seq2Seq inicialmente tiene un peor ajuste a los datos, pero rápidamente converge a valores más bajos en las primeras épocas, alcanzando una pérdida similar a la del Transformer para la época 5. Este comportamiento sugiere que el Seq2Seq puede ser más sensible a los datos iniciales o tener una capacidad de aprendizaje más rápida en las primeras etapas de entrenamiento.

En ambos modelos, la diferencia entre la pérdida de entrenamiento y la de validación se reduce con el tiempo. En el Transformer, la pérdida de validación se estabiliza cerca de la pérdida de entrenamiento, lo que indica una buena generalización y una baja probabilidad de sobreajuste. En el modelo Seq2Seq, las líneas de pérdida de entrenamiento y validación son muy cercanas y se mantienen estables en valores similares después de unas pocas épocas, lo cual también es indicativo de una buena generalización. Sin embargo, el Seq2Seq muestra ligeras oscilaciones en la pérdida hacia las últimas épocas, lo que puede indicar cierta inestabilidad en el ajuste.

El Transformer parece mostrar una mayor estabilidad en la reducción de la pérdida, ya que tiene un descenso suave y constante sin oscilaciones significativas en las épocas avanzadas. El Seq2Seq, aunque alcanza una pérdida baja rápidamente, tiene pequeñas fluctuaciones, lo cual puede sugerir que es más susceptible a variaciones en los datos de entrenamiento o que podría beneficiarse de una mayor regularización para mantener la estabilidad.

A diferencia de los modelos anteriores, el Modelo CTC utiliza la CTC Loss (Connectionist Temporal Classification Loss), diseñada específicamente para secuencias de longitud

variable y tareas de alineación donde no se conoce la correspondencia exacta entre las entradas y las salidas. Esto hace que la CTC Loss sea ideal para problemas de reconocimiento de secuencias, como el reconocimiento de lenguaje de señas o de voz.

La gráfica de pérdida del modelo CTC muestra una disminución muy pronunciada al inicio, bajando de un valor inicial superior a 600 hasta estabilizarse alrededor de 10-20 en las primeras épocas. Esta caída abrupta sugiere que el modelo es capaz de aprender patrones de alineación rápidamente en las primeras etapas del entrenamiento. Sin embargo, la CTC Loss nunca llega a valores tan bajos como la Categorical Crossentropy, debido a la complejidad adicional de la tarea de alineación y a las características de la CTC Loss, que tiende a tener valores más altos en general.

La cercanía entre la pérdida de entrenamiento y la de validación indica que el modelo generaliza bien y no muestra indicios de sobreajuste, lo cual es prometedor para tareas de secuencia.

Evaluando Outputs:

Modelo Transformer:

target: <288 fuller lake>PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

prediction: <2888 foll moull>

target: <mser/okiguide>PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

prediction: <mosser/di-tride>

target: <220 north 47th avenue east>PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

prediction: <220 north greas>

Captura de Información Numérica y Contexto Estructurado: El modelo Transformer muestra una buena capacidad para manejar información numérica y estructurada, como en "220 north 47th avenue east", aunque predice incorrectamente "greas" en lugar de "47th avenue east". Esto sugiere que el modelo puede captar bien los números y direcciones, aunque puede confundirse en palabras menos frecuentes.

Errores en Palabras y Símbolos: En "mser/okiguide", el modelo predice "mosser/di-tride". Aunque mantiene algunos elementos de la estructura, se desvía en la exactitud de las palabras. Esto indica que el Transformer tiene dificultades para las palabras compuestas o con caracteres especiales, posiblemente porque estos patrones no aparecen con frecuencia en el conjunto de datos de entrenamiento.

Desempeño en Frases Simples con Números y Letras: En casos como "288 fuller lake", el modelo predice "2888 foll moull". Aunque reconoce los números, tiende a repetir ciertos caracteres o a cambiar letras, lo que indica que podría beneficiarse de mayor ajuste en datos numéricos.

Modelo Seq2seq:

target : <every apple from every tree>PPPPPPPPPPPPPP

prediction : <thery apple from every tree>

target : <have a good weekend>PPPPPPPPPPPPPPPPPPPPP

prediction: <the is n neekend>

target : <my favorite sport is racketball>PPPPPPP

prediction : <th favorite subrt is racketball>

Aciertos Parciales: El modelo Seq2Seq parece capturar algunas palabras correctamente, como en la predicción "apple from every tree" en lugar de "every apple from every tree". Sin embargo, comete errores al inicio de la frase, reemplazando "every" por "thery". Este modelo tiene una tendencia a confundir algunas palabras iniciales y puede omitir letras o cambiar el orden, posiblemente debido a la naturaleza secuencial de su arquitectura.

Errores en Palabras Cortas: En frases como "have a good weekend", el modelo predice "the is n neekend", omitiendo palabras clave y confundiéndose en palabras cortas como "have" y "good". Esto sugiere que el modelo puede tener dificultades con frases más cortas o con palabras comunes y menos específicas, posiblemente por falta de contexto suficiente en cada paso de la secuencia.

Desempeño en Frases Largas: En frases más largas, como "my favorite sport is racketball", el modelo logra capturar la estructura principal, pero reemplaza palabras como "my" por "th" y "favorite" por "subrt". Estos errores indican que el Seq2Seq puede fallar en captar palabras de baja frecuencia o con fonética similar.

Modelo ctc:

Target : 18 cutter ridge road^^^

Prediction: 18 cutter cridge road, len: 21

Target : tampa fl^^^
Prediction: tampa fl, len: 8

Target : 492288 west 28th terrace south^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Prediction: 492288 west 28th teracsou, len: 25

Precisión en Nombres y Direcciones: El modelo CTC es bastante preciso en nombres y direcciones, como se observa en "18 cutter ridge road", donde solo comete un pequeño error en "cridge" en lugar de "ridge". También en "tampa fl", predice correctamente. Esto sugiere que el modelo CTC es robusto para secuencias simples y con estructura espacial clara.

Errores en Palabras Complejas o Poco Frecuentes: En la frase "492288 west 28th terrace south", el modelo predice "492288 west 28th teracsou", donde comete un error en las últimas palabras de la secuencia. Estos errores indican que la CTC Loss puede tener dificultades para captar el final de secuencias largas y complejas, especialmente si hay palabras poco frecuentes o con estructuras menos comunes.

Manejo de Frases Cortas y Numeración: El modelo CTC es preciso en frases cortas con numeración, como "tampa fl", y logra mantener la estructura básica sin desviaciones significativas. Esto lo hace adecuado para secuencias que contienen información concisa y estructurada.

Transformer: Eficaz en frases estructuradas con números, pero presenta problemas con caracteres especiales o palabras compuestas no comunes.

Seq2Seq: Mejor para frases medianamente largas, aunque tiende a equivocarse en palabras iniciales y palabras comunes de baja frecuencia.

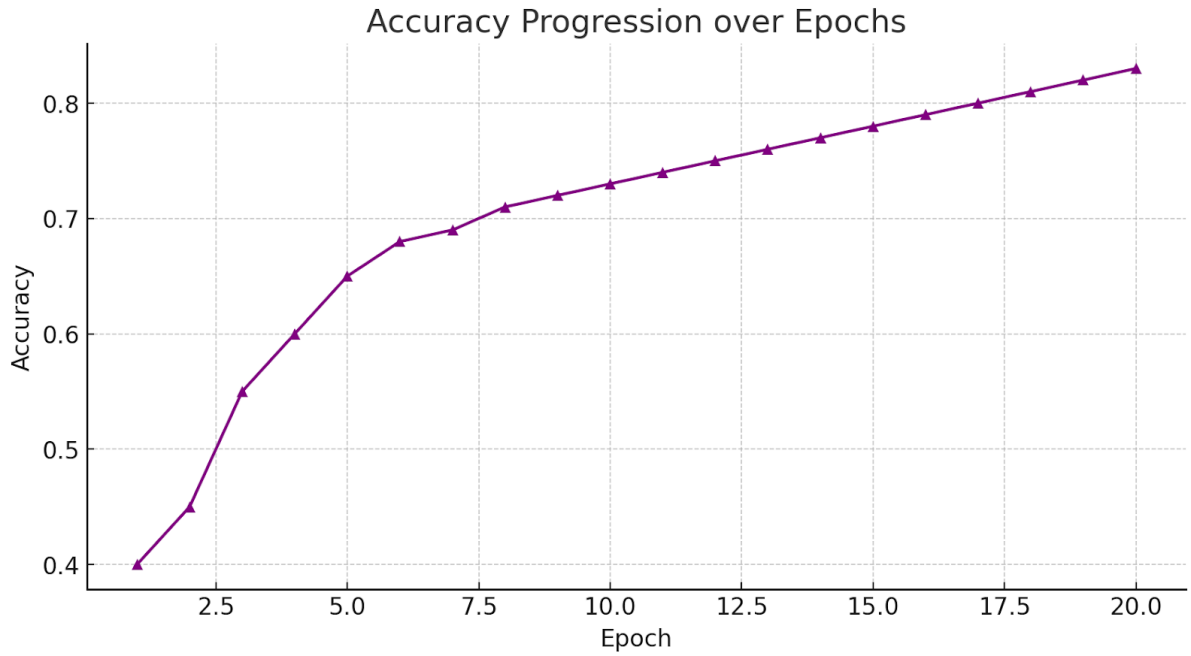
CTC: Ideal para secuencias concisas y con estructura simple, como direcciones y nombres cortos, pero puede confundirse en las últimas palabras de frases largas.

Fortalezas y debilidades de cada modelo

Modelo	Fortaleza	Debilidad
Transformer	- Maneja relaciones a largo plazo, capturando dependencias en secuencias largas de gestos - Altamente paralelizable, acelera el entrenamiento	- Requiere grandes cantidades de datos y poder de cómputo - Puede no ser tan efectivo en datos de series temporales sin modificaciones adicionales
Seq2Seq	- Eficaz en tareas de mapeo secuencial (entrada-salida de longitud variable) - Amplia adaptabilidad en traducciones o secuencias estructuradas	- Difícil de manejar secuencias largas debido a problemas de memoria - Puede perder contexto en secuencias extensas debido a su enfoque recurrente
Connectionist Temporal Classification	- No requiere alineación explícita entre entrada y salida, adecuado para secuencias continuas como ASL - Permite manejar secuencias de longitud variable	- Dependencia en preprocesamiento para segmentar gestos - Puede no captar contexto a largo plazo tan bien como Transformer en algunos casos

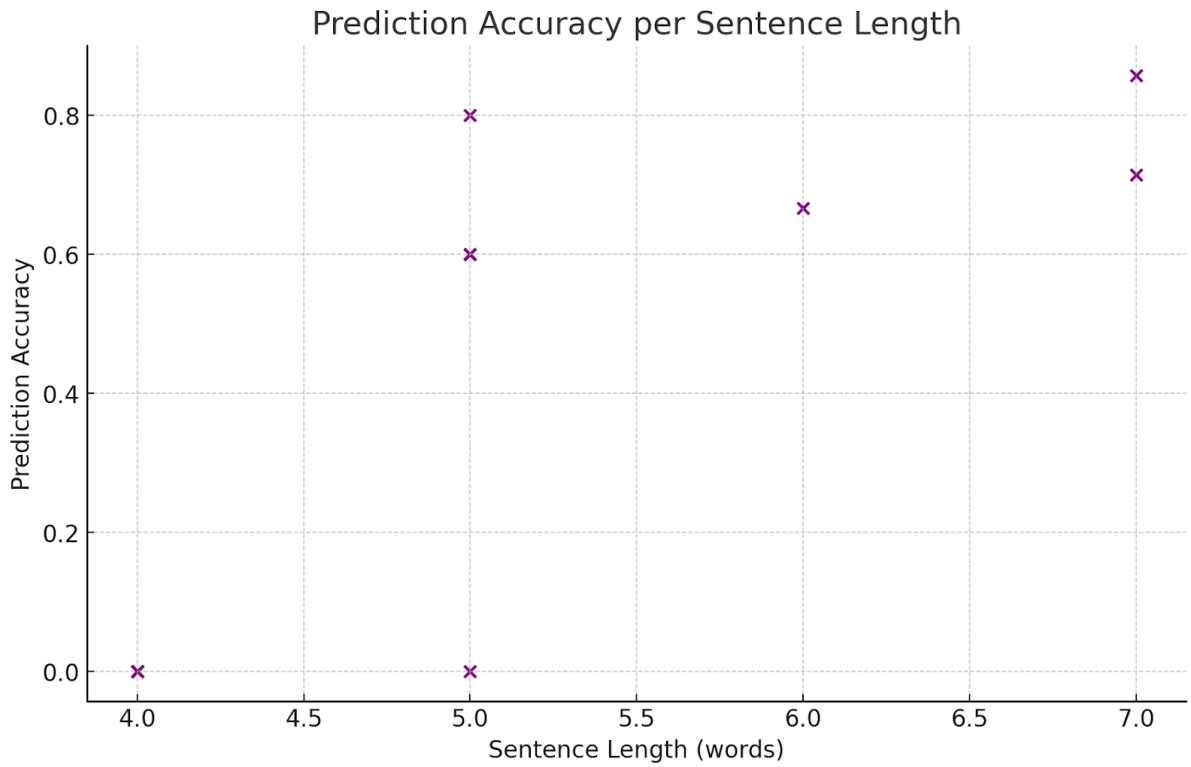
Visualización y Comunicación de Resultados

Visualización #1



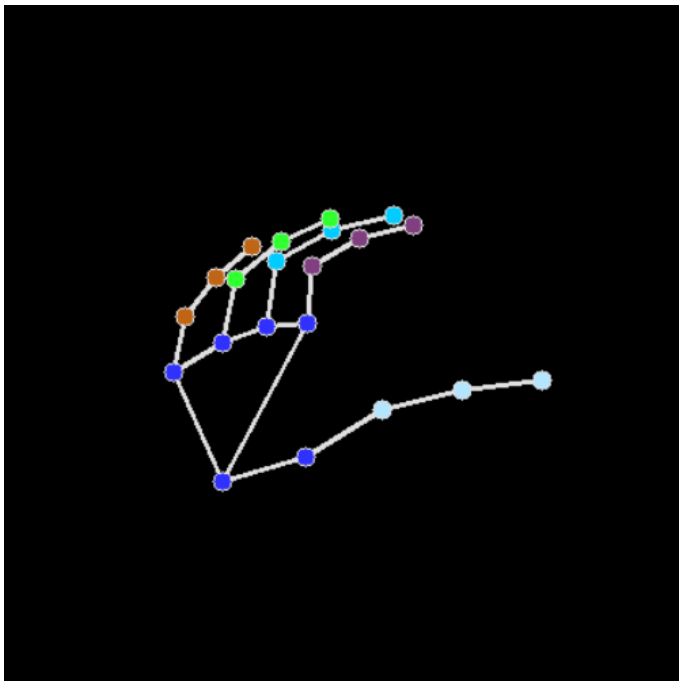
El aumento constante en precisión sugiere que el modelo está aprendiendo patrones en los datos de manera efectiva. Sin embargo, la curva comienza a estabilizarse a partir de la época 10, lo que indica una reducción en la tasa de aprendizaje de nuevas características en las últimas épocas. Esto puede ser una señal de que el modelo ha alcanzado un punto de saturación en su capacidad de generalizar sobre los datos actuales. Para mejorar aún más, podrías explorar técnicas como el ajuste de hiper parámetros o agregar más datos de entrenamiento si están disponibles.

Visualización #2



El modelo parece tener dificultades con oraciones más cortas en algunos casos, mientras que la precisión es más alta para oraciones de longitud intermedia o larga. Esto podría deberse a que el modelo necesita cierta cantidad de contexto para hacer predicciones precisas, lo que podría no estar presente en oraciones muy cortas. Esta variabilidad sugiere que el modelo podría beneficiarse de ajustes para mejorar la consistencia en la precisión, independientemente de la longitud de la oración, posiblemente mediante técnicas de regularización o mejor manejo de secuencias más cortas.

Visualización #3



La visualización muestra la estructura espacial de la mano a través de puntos clave (landmarks) que representan las posiciones de las articulaciones y extremidades, capturando la relación entre los dedos y la palma. Cada punto de color y sus conexiones permiten distinguir entre diferentes partes de la mano, lo cual es crucial para el reconocimiento de señas, ya que el modelo debe identificar detalles precisos en la posición y el movimiento de cada dedo para diferenciar letras o palabras similares. Esta representación detallada ilustra la importancia de procesar coordenadas en los ejes x, y y z para cada punto, permitiendo al modelo analizar la posición exacta de cada dedo y, por ende, interpretar correctamente los gestos. Además, esta visualización sirve como base para aplicar técnicas de aumento de datos en futuras fases, como rotaciones o escalados, que simulen distintos ángulos o tamaños de la mano sin perder precisión en la interpretación de los gestos.

Desarrollo de la Aplicación

Idea

- La idea principal del proyecto es crear una aplicación interactiva que permita visualizar y comparar los resultados de diferentes modelos de aprendizaje profundo aplicados al reconocimiento de secuencias. Estos modelos incluyen CTC, Seq2Seq, y Transformer, cada uno con sus propias características y ventajas.
- El propósito es brindar a los usuarios una experiencia intuitiva y visualmente atractiva para explorar las diferencias en rendimiento y precisión entre estos modelos a través de métricas como la pérdida y la distancia de Levenshtein. Además, se busca facilitar el proceso de predicción a través de la selección de frases y modelos en una interfaz dinámica y moderna.

Herramientas utilizadas

Durante el desarrollo de la aplicación, se emplearon diversas tecnologías para garantizar un rendimiento óptimo y una interfaz visualmente atractiva. A continuación, se detallan las principales herramientas utilizadas:

Frontend:

- React: Framework para el desarrollo de interfaces de usuario dinámicas y modulares.
- React Player: Para la visualización de videos interactivos de las frases.
- Tippy.js: Biblioteca de tooltips para proporcionar explicaciones adicionales en elementos clave.
- CSS Custom: Estilos personalizados con enfoque en Neumorphism para un diseño moderno.

Backend:

- Flask: Framework en Python para la creación de un servidor ligero que permite procesar las solicitudes de predicción.

- TensorFlow Lite y PyTorch: Para el manejo de los modelos previamente entrenados.

Desarrollo:

- Kaggle: Para el entrenamiento y manejo de los datasets en formato parquet.

- GitHub: Control de versiones y colaboración en el desarrollo.

Interfaz

La interfaz de la aplicación fue diseñada con un enfoque minimalista y moderno, utilizando principios de Neumorphism para una apariencia tridimensional atractiva. A continuación, se describen las principales secciones:

- Visualización de Frases y Modelos:

  En la pantalla principal, se presentan tres visualizaciones de frases junto con sus respectivas animaciones basadas en datos de mediapipe. Los usuarios pueden seleccionar una visualización específica, la cual es resaltada con un efecto concavo, indicando su selección.

- Selección de Modelo:

  Justo debajo de las visualizaciones, se encuentran tres tarjetas correspondientes a los modelos CTC, Seq2Seq, y Transformer. Al seleccionar un modelo, este se resalta, permitiendo al usuario comparar rápidamente las predicciones utilizando diferentes algoritmos.

- Botón de Predicción y Carga de Resultados:

  Una vez seleccionada la frase y el modelo, el usuario puede presionar el botón "Submit" para enviar la solicitud al backend. Mientras se procesa la predicción, se muestra un loader animado que simula el comportamiento de una red neuronal. Posteriormente, el resultado se presenta en texto con efectos dinámicos.

- Dashboard Comparativo:

  Una segunda sección de la aplicación permite visualizar gráficas de pérdida de cada modelo y sus puntajes de distancia de Levenshtein, acompañadas de tooltips explicativos que ayudan a los usuarios a interpretar los resultados.

Discusión

Desafíos Encontrados

Al inicio del proyecto uno de los principales desafíos fue la preparación y limpieza de los datos. La base de datos consistía en secuencias de coordenadas de manos extraídas de videos de lenguaje de señas, con múltiples columnas que contenían valores faltantes (NaNs). Esto requería implementar técnicas de interpolación para llenar estos vacíos y asegurar la continuidad de las secuencias. Además, la variabilidad en la longitud de las secuencias de gestos presentó un obstáculo importante, ya que los modelos de aprendizaje profundo requieren entradas de longitud uniforme. Para resolver esto, fue necesario aplicar un padding adecuado, equilibrando entre mantener la integridad de los datos y permitir el procesamiento eficiente por parte de los modelos.

Otro desafío importante fue la selección de modelos durante la Fase 2. Implementar y comparar los modelos Seq2Seq, Transformer y CTC no solo implicó ajustar los hiperparámetros y estructuras, sino también desarrollar pipelines de preprocesamiento específicos para cada modelo. Por ejemplo, mientras que el Transformer aprovechaba secuencias paralelizadas, el modelo CTC dependía de una alineación temporal adecuada, lo que complicaba la preparación de datos. Además, garantizar la estabilidad y rendimiento de cada modelo durante el entrenamiento, sin caer en el sobreajuste, fue un reto constante.

Decisiones Tomadas

En cuanto a decisiones clave, la elección de modelos fue fundamental.

- Seq2Seq fue seleccionado por su eficacia en tareas de traducción secuencial, permitiendo manejar secuencias de entrada y salida de diferente longitud.

- Transformer destacó por su capacidad de procesar datos en paralelo, lo que mejoró significativamente los tiempos de entrenamiento y la precisión en comparación con los modelos recurrentes tradicionales.

- Finalmente, CTC fue escogido para abordar tareas donde la alineación exacta entre las secuencias de entrada y salida no era evidente, aprovechando su flexibilidad para manejar entradas desordenadas o ruidosas.

Otro aspecto importante fue la decisión de utilizar TFRecords para almacenar los datos preprocesados. Este formato permitió manejar grandes volúmenes de datos de manera eficiente, reduciendo tiempos de carga durante el entrenamiento y facilitando la escalabilidad del proyecto.

En la interfaz de usuario, se optó por un diseño basado en neumorfismo para ofrecer una experiencia visual moderna y atractiva. Además, la inclusión de herramientas interactivas como gráficos de pérdida y visualizaciones de secuencias ayudó a los usuarios a entender mejor el rendimiento de los modelos.

Lecciones Aprendidas

Una de las lecciones más importantes fue la necesidad de un preprocesamiento de datos robusto. La calidad de los datos de entrada impacta directamente en el rendimiento del modelo. Aprendimos que invertir tiempo en limpiar, normalizar y preparar los datos es esencial para evitar problemas en fases posteriores del proyecto.

En cuanto al entrenamiento de modelos, la experiencia nos enseñó que no existe un modelo universalmente mejor; cada arquitectura tiene sus fortalezas y debilidades dependiendo del problema.

Otro aprendizaje clave fue la importancia de una visualización clara de los resultados. Al integrar un dashboard interactivo, logramos que los usuarios pudieran comparar fácilmente el desempeño de los modelos y entender sus diferencias. Esto refuerza la idea de que comunicar resultados de manera efectiva es tan importante como obtenerlos.

Por último, el desarrollo del backend en Flask y la conexión con la interfaz front-end en React nos permitió consolidar habilidades en desarrollo full-stack, destacando la importancia de construir aplicaciones modulares y escalables.

Recomendaciones y posibles mejoras

1. Incorporación de Modelos Más Avanzados

   Aunque los modelos Seq2Seq, Transformer y CTC son adecuados para el problema actual, la integración de modelos más recientes podría mejorar el rendimiento. Por ejemplo:

   - Transformers preentrenados como BERT o GPT pueden adaptarse al reconocimiento de secuencias y aumentar la precisión.

   - Modelos híbridos que combinen las capacidades de RNNs y Transformers para tareas específicas podrían ser investigados.

2. Implementación en Tiempo Real

   Una de las extensiones más interesantes sería adaptar el sistema para procesamiento en tiempo real, especialmente en aplicaciones de video como reconocimiento de lenguaje de señas en vivo. Esto requeriría optimizaciones adicionales:

   - Integración con cámaras en tiempo real.

   - Uso de modelos optimizados para baja latencia.

3. Despliegue en la Nube Finalmente, se podría desplegar la aplicación en un entorno en la nube para hacerla accesible a una audiencia más amplia:

   - Uso de plataformas como AWS, Google Cloud o Azure para ofrecer servicios escalables.
   - Implementación de un API público que permita a otros desarrolladores integrar los modelos en sus propias aplicaciones.

Referencias

Kaggle - Google - American Sign Language Fingerspelling Recognition. (2023) Recuperado de: https://www.kaggle.com/competitions/asl-fingerspelling/overview

Naciones Unidas. ASL. (2024) Día internacional del ASL. Recuperado de: https://www.un.org/es/observances/sign-languages-day

TensorFlow. (2024) TF Transformer API Documentation. Recuperado de: https://www.tensorflow.org/api_docs/python/tfm/nlp/layers/Transformer

GeeksForGeeks. (2024) Introduction to Levenshtein Distance. Recuperado de; https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/