CACiC 2023
CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN

Desarrollo de un modelo con XGBoost intérprete de la Lengua de Señas Argentina

Facundo Recabarren, Fabrizio Amaya, Raúl Klenzi, María Isabel Masanet

Universidad Nacional de San Juan, San Juan 5400, Argentina
{facunicolasrecabarren, fabrizio.amaya88, rauloscarklenzi, mimasanet}@gmail.com

Resumen. Se propone el desarrollo de un modelo con XGBoost intérprete de la Lengua de señas Argentina (LSA), encontrando la correspondencia entre una secuencia de imágenes (video) y su apropiado texto en español. Para esto se plantea un modelo en árboles de decisión optimizado aplicado a un conjunto de datos preexistentes sobre el que se realizan tareas de preprocesamiento, muestreo y almacenamiento.

Palabras clave: Lengua de Señas Argentina, LSA, Machine Learning, Data Science, MediaPipe, XGBoost,

1. Introducción

Las lenguas de señas son lenguas que utilizan la modalidad visual-gestual para transmitir significado a través de articulaciones manuales en combinación con elementos no manuales como el rostro y el cuerpo. Sirven como el principal medio de comunicación para numerosas personas sordas y con problemas de audición. Al igual que los lenguajes orales, los lenguajes de señas son lenguajes naturales regidos por un conjunto de reglas lingüísticas y a diferencia de aquellos, la Lengua de Señas de un idioma en particular, no es una forma visual de ese idioma, sino que es un lenguaje propio. El procesamiento del lenguaje de señas es un campo emergente de la inteligencia artificial, centrándose su investigación en los aspectos visuales de estos lenguajes, cubriendo un subcampo tanto del procesamiento de lenguaje natural (NLP) como de la visión por computadora (CV). https://research.sign.mt/

La Comunidad Sorda es un grupo minoritario que lentamente ha logrado integrarse en las comunidades de todo el mundo. Si bien en algunos países su integración está muy avanzada, en nuestro país aún queda mucho por hacer para lograrlo.

Las personas sordas hacen un gran uso de la tecnología, priorizando el contexto "visual" debido a que la lengua de señas se entiende por el sentido de la vista. Es así que utilizan videollamadas, mensajes de texto, teléfonos móviles, notebooks y todo aquello que les permite comunicarse y darse a entender.

San Juan carece de los medios y tecnologías adecuadas para ellos, y no se cuenta con intérpretes de LSA en casi ningún ámbito, ya sea hospitales, bancos, edificios públicos, incluso tampoco los hay en escuelas, universidades y demás.

Este trabajo presenta el desarrollo de un modelo que interpreta señas de la Lengua de Señas Argentina (LSA) y forma parte de una tesis de Licenciatura en Ciencias de la Computación en ejecución.

2. Obtención de Datos

Para el entrenamiento de los distintos modelos a analizar es necesario contar con un Set de Datos con el cual llevar a cabo ese entrenamiento. Este Set de Datos puede construirse a medida para el problema en particular o también, en caso de ser posible, usar un Set de Datos ya construido y que pueda ser aplicado al contexto del problema.

En esta propuesta se hace uso del Set de Datos "LSA64" para llevar a cabo el entrenamiento, este Set de Datos fue obtenido como parte de los resultados de la Tesis Doctoral de Franco Ronchetti en [1] y se permite la utilización del mismo en [2].

2.1 Características del Set de Datos

El Set de Datos cuenta con un conjunto de 64 señas interpretadas, cada una de ellas, por 10 personas las cuales, a su vez, grabaron 5 videos para cada seña, obteniendo un total de 3200 videos (50 videos por cada seña). La particularidad de este set de videos, es que los intérpretes utilizan un guante de color diferente por mano (rosado en la mano derecha y verde en la izquierda), siendo en este caso la mano derecha la dominante.

"La base de datos fue construida en dos sets de grabación distintos. En el primero fueron grabadas 23 señas con una sola mano y se utilizó luz natural en un entorno abierto. En el segundo set, se agregaron 41 señas más (22 con dos manos, y 19 con una mano) y se utilizó luz artificial en un entorno semi-cerrado…. La base de datos es públicamente accesible bajo licencia Creative Commons Attribution-NonCommercialNoDerivatives 4.0 International sólo para uso académico. Se encuentra almacenada en https://midusi.github.io/lsa64/index.html." [2]

Otra característica de los videos es que estos fueron capturados en calidad FullHD o 1920x1080 y a 60 fotogramas por segundo (FPS).

De la lista original de 64 señas se han tomado un subconjunto de 21 señas que se realizan con una mano con el objetivo de poder formar palabras con sentido.

3. Procesamiento de videos

Se utiliza un framework de Machine Learning llamado "MediaPipe" [3] el cual es capaz de procesar videos e imágenes entregando como resultado la detección de personas en la imagen recibida como entrada.

La salida que entrega el modelo es una secuencia de coordenadas conocida como skeletal data. Estos datos se encuentran normalizados dentro del intervalo [0.0, 1.0] con respecto al alto y ancho de la imagen a procesar.

Dentro de este framework se encuentran distintos modelos pre-entrenados y listos para usar, utilizándose en este caso el llamado "Holistic Tracking" dado que permite obtener una estimación de puntos corporales correspondientes a ambas manos, la cara y la posición corporal con un único modelo.

3.1 Preprocesamiento

Los videos originales atraviesan una primera etapa de procesamiento en la cual se obtiene de cada video un equivalente de los frames o imágenes que lo componen, pero representado por una lista de puntos corporales asociados a cada frame detectado por el modelo holístico. Así, cada video se representa por una lista de puntos corporales o coordenadas para la persona detectada en cada frame, de suma relevancia dado que el algoritmo XGBoost se destaca por sobre otros, cuando procesa datos tabulares [4] [5].

3.2 Muestreo o Sampling

Una vez obtenidos los datos numéricos a través del preprocesamiento de los videos y a los efectos de reducir la tasa de datos a procesar, se procede a capturar ciertos frames de forma aleatoria o siguiendo alguna técnica empírica específica.

Seguidamente, se detallan 3 alternativas para acotar los videos a 10 y 30 frames (X):

- Muestreo Random: Se toma para cada video una cantidad de X frames aleatoriamente, siendo estos frames tomados del conjunto de frames/imágenes que componen cada video.
- Muestreo Random con Porcentajes: En este caso se separa cada video en 3 secciones: inicial, intermedia y final. La sección inicial está compuesta por frames aleatorios provenientes del primer 20% del video, luego el siguiente 60% de frames conforma la sección intermedia y el 20% restante representa la sección final. De cada sección se tomará la cantidad necesaria de frames de manera de poder completar el primer 20% de X, luego 60% y 20% restante. Por ejemplo: para un muestreo de 10 frames, se tomarán 2 frames de la sección inicial, 6 frames de la sección intermedia y por último 2 frames de la sección final.
- Muestreo Random con Poda: El tercer muestreo realiza una poda de frames del 15% de frames los iniciales y finales, es decir, una poda del 30%, exceptuando aquellos videos donde al quitar ese 30% la cantidad de frames restantes no es suficiente para completar la cantidad de X frames necesarios.

Las 3 técnicas se utilizaron para armar sets de datos acotados a 10 y 30 frames, además, se armaron sets de datos con y sin datos faciales con el objetivo de analizar su performance. Como salida de este paso, se obtuvieron 12 sets de datos, resultantes de la combinación de los distintos muestreos realizados.

4. Entrenamiento

El siguiente paso consiste en la construcción, entrenamiento y optimización del modelo sobre los distintos sets de datos obtenidos en el paso previo. El modelo elegido está construido con XGBoost (Xtreme Gradient Boosting) la cual "es una librería de optimización del gradiente distribuida y optimizada para ser altamente eficiente, flexible y portátil" [4].

4.1 XGBoost

XGBoost proviene del inglés “Extreme Gradient Boosting” o “Refuerzo extremo del gradiente” es una herramienta desarrollada para entrenar mediante el aprendizaje supervisado árboles de decisión. Cada árbol que se genere aprenderá de sus predecesores e intentará disminuir el error incurrido en ellos.

El funcionamiento puede describirse como:

“Se obtiene un árbol inicial F0 para predecir la variable objetivo ‘y’, el resultado se asocia con un residual (y - F0).

Se obtiene un nuevo árbol h1 que ajusta el error del paso previo.

Los resultados de F_0 y h_1 se combinan para obtener el árbol F_1, donde el error cuadrático medio de F_1 será menor que el de F_0:

$$F_1(x) = F_0(x) + h_1(x)$$

Este proceso se sigue iterativamente hasta minimizar el error de la siguiente forma:

$$F_m(x) = F_{m-1}(x) + h_m(x)$$” [5]

Los datos obtenidos en la fase de preprocesamiento se utilizan como entrada directa para este modelo, trabajando con datos tabulares .

4.2 Evaluación

Los sets de datos construidos anteriormente son divididos en 3 subconjuntos: entrenamiento, validación y testeo. Luego de realizar un primer entrenamiento se procede a una siguiente etapa de “Optimización de Hiperparámetros” con el objetivo de encontrar una combinación de valores que eleve el rendimiento de cada modelo.

Los hiperparámetros evaluados son: learning_rate, n_estimators, max_depth, min_child_weight, subsample, colsample_bytree, reg_alpha y reg_lambda.

Durante el proceso de entrenamiento se hace uso de técnicas como cross_validation. Para evaluar los resultados que presenta el modelo se seleccionaron las siguientes métricas: Precision, Recall, F1-score, F05-score, F2-score y Accuracy.

En Tabla 1 se presentan los resultados obtenidos sobre 2 sets de datos, con distinta técnica de muestreo, antes y luego de haber realizado el proceso de optimización:

Tabla 1. Resultados de entrenamiento para subconjunto de testeo en modelo base y optimizado.

Métrica/Muestreos	**criterio Random 30 frames sin cara**		**criterio Proporción 30 frames con cara**	
	Base	Optimizado	Base	Optimizado
Precision	0.93	0.96	0.95	0.95
Recall	0.92	0.95	0.93	0.95
F1	0.92	0.95	0.93	0.95
F05	0.93	0.95	0.94	0.95
F02	0.92	0.95	0.93	0.95
Accuracy	0.92	0.95	0.93	0.95

5. Conclusión

La etapa de entrenamiento al ejecutar el algoritmo XGBoost con cross-validations demora entre 5 y 8 minutos. El equipo donde se trabajó posee los siguientes componentes: Intel I5 8va generación, GPU GTX 1050 2GB, aunque también actualmente se están realizando pruebas sobre Google Colab.

La Tabla1. muestra como mejoran las métricas tras la optimización de hiperparámetros. Puede observarse cómo el proceso aumenta en un pequeño porcentaje (0.01-0,03) el rendimiento, siendo este rendimiento evaluado en base a la seña que el modelo interpreta para cada video de entrada no procesado previamente (subconjunto de testeo). La elección de los sets de datos para Tabla1. está basada en aquellos que presentaron un mayor porcentaje de mejora en el proceso de optimización de hiperparámetros. De ser necesaria la elección de un modelo final, en base a lo observado en Tabla1., el modelo construido sobre el set de datos "criterio Random 30 frames sin cara" sería el elegido, debido a que, con menor cantidad de información (no se cuenta con datos faciales), se logran resultados similares y/o iguales.

Actualmente se están analizando modelos construidos con capas LSTM/GRU pertenecientes a Redes Neuronales Recurrentes de los cuales próximamente se contarán con los resultados pertinentes y se realizarán comparaciones que resulten necesarias.

Así mismo, con la asistencia de intérpretes de LSA se pretende generar desde el set de videos originales una mayor cantidad de secuencias de imágenes con frases coherentes y con ello lograr reconocer los inicios y finales de cada palabra en la secuencia.

Referencias y Bibliografía

1. Ronchetti, Franco: Reconocimiento de gestos dinámicos y su aplicación al lenguaje de señas. Tesis Doctoral, Facultad de Informática (2017).
2. Ronchetti, Franco and Quiroga, Facundo and Estrebou, Cesar and Lanzarini, Laura and Rosete, AlejandroLSA64: A Dataset for Argentinian Sign Language. Recuperado el 01 de Agosto del 2023 de http://facundoq.github.io/datasets/lsa64/
3. MediaPipe Framework Recuperado el 04 de Agosto del 2023 de https://developers.google.com/mediapipe/framework
4. XGBoost. Recuperado el 04 de Agosto del 2023 de https://xgboost.readthedocs.io/en/stable/
5. Espinoza, J.: Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. Ingeniería Investigación y Tecnología volumen XXI, 3 (2010).
6. Sign Language Procesing. Recuperado el 01 de Agosto del 2023 de https://research.sign.mt/ Sai B. Padigala, Gogineni H. Madhav, Saranu K. Kumar, Dr. Narayanamoorthy M: Video based sign language recognition using CNN-LSTM. IRJET, pp 1658-1663 (2022)
7. Ko, S.-K., Kim, C. J., Jung, H., & Cho, C.: Neural Sign Language Translation Based on Human Keypoint Estimation. MDPI, pp 2683 (2019) http://dx.doi.org/10.3390/app9132683
8. DONGXU LI, Cristian Rodriguez, Xin Yu, HONGDONG LI; Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp 1459-1469 (2020).
9. Zhou, Z., Chen, K., Li, X. et al. Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. Nat Electron 3, pp 571–578 (2020). https://doi.org/10.1038/s41928-020-0428-6
10. HCDPS. Leyes Provinciales N° 7412 (2003) y N° 761-S (2014).