Modelo de Clasificación para leads en Pokemon VGC

Abstract

La manera en que los humanos pensamos al jugar juegos de estrategia está limitada por la cantidad de información que podemos procesar para identificar patrones o encontrar soluciones óptimas. En este trabajo se propone utilizar una arquitectura basada en embeddings para predecir con qué Pokémon iniciará el rival en un juego de Pokémon competitivo VGC. Se analiza la hipótesis de que este problema puede abordarse como una tarea de clasificación. Dada una secuencia de Pokémon, se espera que existan patrones que indiquen cuáles dos Pokémon probablemente serán seleccionados por el rival. El embedding se conecta a capas fully connected para que la red neuronal pueda identificar estos patrones y evaluar si ciertos Pokémon, al estar juntos en el mismo equipo, tienen mayor probabilidad de ser escogidos como leads.

Glosario de Pokémon

Lead: Pokémon con el cual inicia la partida el jugador o rival.

Open Team Sheet: Formato en donde se puede ver la información de los pokemones del equipo rival.

Introducción

El mundo del Pokémon competitivo está profundamente ligado a las matemáticas y al análisis de probabilidades, lo cual puede resultar abrumador para jugadores nuevos. Para quienes no juegan de manera competitiva, puede parecer trivial decidir con qué Pokémon iniciar una partida, pero en realidad esta elección es mucho más compleja.

Para entender por qué no es sencillo elegir tu Pokémon más fuerte, es necesario explicar las reglas básicas del formato oficial de Pokémon competitivo. Las partidas oficiales de Nintendo siguen ciertas reglas: el juego consiste en **batallas de 2 contra 2**, donde de los 6 Pokémon que cada jugador lleva, sólo se usan 4 en la partida. En otras palabras, no jugarás con todos tus Pokémon, por lo que es necesario decidir cuáles 4 traer en cada enfrentamiento.

En el formato del juego se aplica el **Open Team Sheet**, lo que significa que ambos jugadores conocen la información del equipo rival. Es decir, cada jugador sabe cuáles son los 6 Pokémon del oponente. Este conocimiento hace que elegir tus Pokémon sea más complicado, ya que no quieres traer un Pokémon que sea débil frente al equipo rival.

Teniendo toda la información de los Pokémon disponibles, es posible identificar patrones: al observar qué Pokémon trae el rival y cuáles traes tú, se puede aprender la probabilidad de que saque ciertos dos Pokémon como leads al inicio de la partida.

Dataset

El dataset utilizado en este proyecto proviene de los registros de partidas del servicio Pokémon Showdown, un simulador de batallas Pokémon de código abierto. Un agradecimiento especial al usuario “beelzebruno” por publicar su cuaderno en Kaggle y compartir el código utilizado para descargar las partidas de Pokémon Showdown, así como por proporcionar un script sencillo para obtener la información necesaria para la construcción del dataset.

Los registros de las partidas incluyen todas las acciones que se realizaron durante cada enfrentamiento, así como los Pokémon iniciales de ambos jugadores y sus leads. Mediante un simple regex, se logró extraer un total de 4,935 partidas válidas. El dataset contiene las siguientes columnas: los seis Pokémon del jugador 1, los seis Pokémon del jugador 2, los dos Pokémon leads del jugador 1 y los dos Pokémon leads del jugador 2.

Justificación

Para este problema se consideraron varias arquitecturas. En un inicio se planteó el uso de una red neuronal recurrente (RNN/LSTM), sin embargo, se descartó.
La razón es que, en este contexto, el orden de los Pokémon dentro del equipo no debe aportar información al modelo. Es decir, no queremos que la red aprenda patrones del tipo: “si el Pokémon X aparece en la primera posición, entonces es más probable que el siguiente sea el Pokémon Y”.
En realidad, el orden dentro del equipo es irrelevante: los seis Pokémon solo representan un conjunto, no una secuencia. Las RNN/LSTM están diseñadas para datos secuenciales donde el orden sí modifica el significado (como texto o series de tiempo). Aquí ocurriría lo contrario: podrían aprender relaciones que no existen entre posiciones que no reflejan el juego real.

Se pasó también usar Se pesó también usar una red neuronal convulsional pero recae en el mismo problema de que estaría captando patrones entre las posiciones más que en los mismos pokemones

Por esta razón se optó por una arquitectura basada en capas densas, que trata el equipo como un vector de características sin asumir una dependencia secuencial.

Modelo base

La arquitectura propuesta fue implementada utilizando una red neuronal basada en técnicas de representación mediante embeddings, comúnmente empleadas en tareas de Procesamiento de Lenguaje Natural. Dado que cada Pokémon es un valor categórico y no un vector numérico, el primer componente del modelo es una capa de embedding, la cual transforma cada Pokémon en un vector de dimensión 16.

El modelo recibe como entrada una secuencia de longitud fija compuesta por los 6 Pokémon del jugador A y los 6 Pokémon del jugador B. Tras pasar por la capa de embedding, los 12 vectores resultantes se aplanan en un único vector y se procesan mediante 2 capas densas de 64 . Dichas capas transforman progresivamente la representación para capturar características de composición de equipos y arquetipos comunes dentro del metajuego competitivo.

El modelo cuenta con dos salidas paralelas, cada una implementada como un clasificador de tipo softmax. La primera predice cuál será el primer Pokémon enviado al campo por el oponente, y la segunda predice el segundo Pokémon inicial. Ambas salidas se entrenan empleando sparse categorical cross-entropy y se evalúan mediante la métrica de exactitud (accuracy).

A diferencia de arquitecturas recurrentes o basadas en transformers, este modelo no aprende dependencias posicionales dentro de la secuencia, lo cual es deseable, puesto que el orden de los Pokémon dentro de una tabla no representa ningún significado real. Así, el equipo es tratado como un conjunto, no como una secuencia.

Comienza con el preprocesamiento de los registros crudos de batallas provenientes de Pokémon Showdown. Mediante expresiones regulares personalizadas se extraen únicamente combates válidos y se obtienen tres elementos:

los seis Pokémon del jugador A,

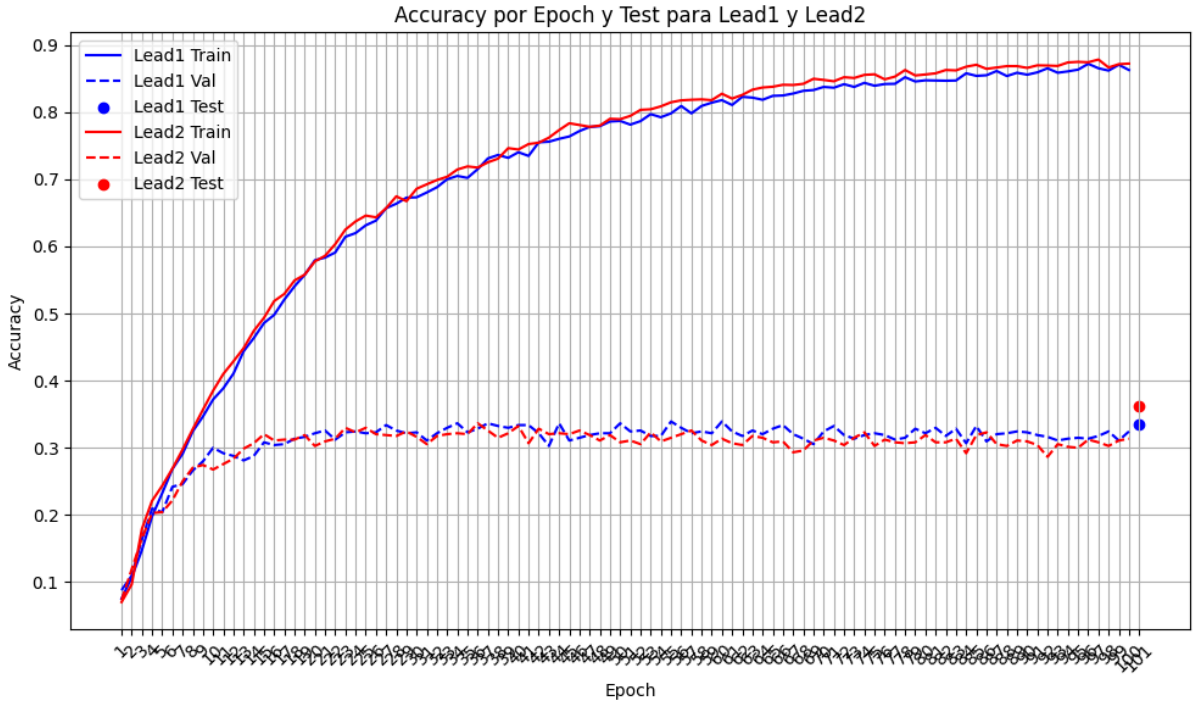
los seis Pokémon del jugador B, y

los dos Pokémon iniciales que entraron al campo en cada lado.

Los nombres de Pokémon se convierten a valores enteros categóricos, generando un vector numérico de entrada de tamaño 12. El conjunto de datos se separa en entrenamiento, validación y prueba.

Durante el entrenamiento, los vectores pasan por la capa de embedding, posteriormente son aplanados y procesados por la red neuronal. Los resultados se evalúan mediante exactitud, tanto para la predicción del primer lead como para la del segundo. Se utilizó el optimizador Adam durante 100 épocas de entrenamiento.

Resultados del primer modelo



Train accuracy lead1 0.86%
Val accuracy lead1 0.32%
Test accuracy lead1 0.33%

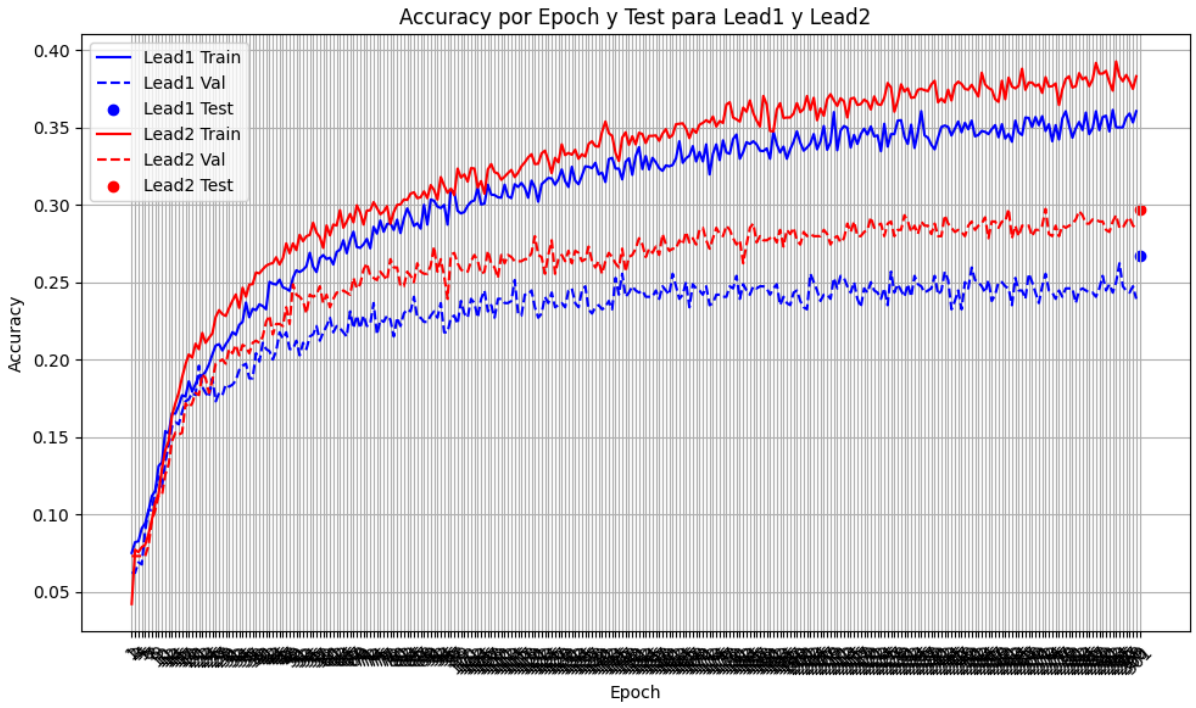
Train accuracy lead2 0.87%
Val accuracy lead2 0.31%
Test accuracy lead2 0.36%

Se puede observar que el primer modelo presentó un caso evidente de overfitting.
Este modelo utiliza un embedding de tamaño 16 y dos capas densas de 64 neuronas, sin aplicar ninguna técnica de regularización como dropout.

Resultados del segundo modelo

El sobreajuste ocurre porque la red memoriza los datos de entrenamiento en lugar de aprender patrones generales que permitan predecir casos nuevos. Como el modelo es relativamente pequeño en cuanto a número de capas, una buena estrategia inicial no es reducir aún más la arquitectura, sino ajustar otros parámetros para limitar su capacidad de memorización.

Se redujo el tamaño del embedding a 8 dimensiones con el objetivo de disminuir la capacidad del modelo para memorizar los datos y obligarlo a generalizar mejor.
Además, las dos capas densas internas se ajustaron también a 8 y 16 neuronas cada una, reduciendo así la cantidad total de parámetros del modelo. Finalmente, se agregó una capa dropout con una tasa de 0.1 en la última capa densa para prevenir que el modelo dependiera excesivamente de neuronas específicas durante el entrenamiento y mitigar el sobreajuste.



Train accuracy lead1 0.36%
Val accuracy lead1 0.23%
Test accuracy lead1 0.26%

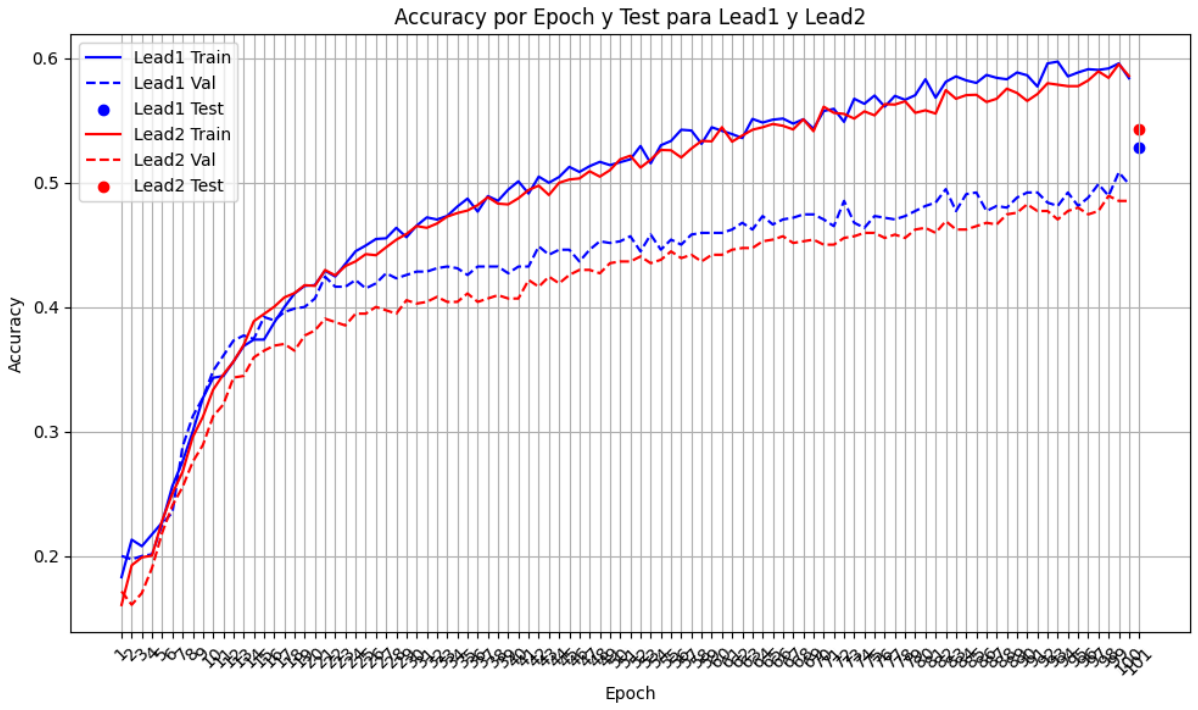
Train accuracy lead2 0.38%
Val accuracy lead2 0.28%
Test accuracy lead2 0.29%

Con esta arquitectura fue posible reducir significativamente el sobreajuste, pasando de un nivel alto a uno moderado, aproximadamente alrededor del 10% de diferencia entre el entrenamiento y la validación.
Sin embargo, el nuevo problema es que el modelo continúa mostrando un desempeño bajo en la tarea principal. La precisión de predicción promedio se mantiene por debajo del 30%. Esto indica que, aunque ahora el modelo generaliza mejor y ya no memoriza los datos, su capacidad para aprender patrones útiles sigue siendo limitada.

Resultados del tercer modelo

El competitivo de Pokémon no es sencillo de modelar. Cada jugador toma decisiones basadas en su propio estilo, experiencia y estrategia, y este comportamiento no siempre es consistente entre partidas. A falta de un conjunto de datos que permita analizar el historial individual de cada jugador, es extremadamente complejo predecir con exactitud absoluta cuál será su elección de lead.

Sin embargo, esto no implica que no existan patrones. Aunque el modelo no pueda identificar con certeza el 100% de la selección del oponente, sí puede aprender tendencias comunes dentro del metajuego y, a partir de ellas, sugerir un pequeño conjunto de opciones probables. En otras palabras, incluso sin precisión total, un modelo que reduzca las posibilidades a tres Pokémon con alta probabilidad ya representa una ventaja estratégica, mostrando que existe un comportamiento recurrente y predecible en una parte del juego competitivo.



Train accuracy lead1 0.58%
Val accuracy lead1 0.49%
Test accuracy lead1 0.52%

Train accuracy lead2 0.58%
Val accuracy lead2 0.48%
Test accuracy lead2 0.54%

Los resultados son los esperados. Al permitir un rango correcto de predicciones (top-3) en lugar de una sola, todas las métricas mejoraron. Aproximadamente el modelo acierta un poco más del 50% cuando se considera que alguno de esos 3 Pokémon está presente. Sin embargo, esto no resuelve por completo el problema de que el modelo predice mal. Es necesario revisar a qué se debe esto, más allá de la arquitectura.

Conclusiones y siguientes pasos

Para poder continuar es necesario construir una matriz de conjunción (confusión/contingencia) que nos muestre qué está prediciendo el modelo y cuáles son sus tendencias. Con esa matriz podremos diseñar un nuevo loss que se ajuste mejor al comportamiento observado. Aunque todavía no tengamos la matriz final, conviene dejar en claro algunos puntos que ya se han detectado y que debemos abordar.

Primero, hay que actualizar la función de pérdida. Actualmente la métrica se calcula como top 3, pero la pérdida sigue siendo sparse categorical crossentropy (top 1). Eso obliga al modelo a optimizar solo si acierta exactamente la primera opción, y no lo incentiva a reconocer que la predicción correcta puede ser difícil o que lo útil es proponer un conjunto reducido de pokemones posibles. Podemos diseñar un loss que penalice menos si la etiqueta real está dentro del top-k o que incorpore prioridades distintas para lead1 y lead2.

Segundo, ambos softmax (lead1 y lead2) a veces predicen el mismo Pokémon. En el juego real no es posible enviar el mismo Pokémon dos veces al inicio. Aunque en la práctica la diferencia entre (X,Y) y (Y,X) rara vez importa, y hablo de casos únicos, no queremos que el modelo “minimice el loss” repitiendo la misma predicción en ambas salidas.

Tercero, el dataset podría ser insuficiente si solo usamos los nombres de los Pokémon. Sería recomendable dar más detalle a las entradas con movimientos, habilidades, objetos y estadísticas, que están disponibles en los registros de Pokémon Showdown. Estos features aportan estrategias reales que ayudan a distinguir roles y propósitos dentro de un equipo.

Por último, una buena mejora sería preentrenar embeddings con skip-gram sobre los equipos, lo que permitirá vectores iniciales que capturen la “quimica” entre pokemones, antes de entrenar el modelo.