RoyaleNet: Deck Recommendation for Card-Based Competitive Online Videogames

Benjamín Farías
bffarias@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

Benjamín Lepe
balepe@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

Juan Romero
jaromero6@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

ABSTRACT

Los sistemas recomendadores han tenido un fuerte impacto en el mundo de los videojuegos, como es el caso del juego online para celular llamado *Clash Royale*, donde los jugadores se enfrentan en una batalla basada en mazos de cartas. Para este caso en específico, la elección de las cartas es fundamental para lograr la victoria, y por tanto, lo que se propone en este trabajo es crear un sistema recomendador capaz de mostrarle al usuario aquellas cartas con las que pueda maximizar sus probabilidades de ganar, todo esto según las cartas elegidas por su oponente. Para lograr este objetivo, se implementó una red neuronal que fue entrenada con un dataset de decenas de miles de partidas reales jugadas hace un par de años, de forma que el modelo pudiese aprender las combinaciones de cartas óptimas contra diversos tipos de oponentes. El resultado obtenido fue que el modelo propuesto supera con creces el rendimiento de un recomendador aleatorio, y más aún, logra una precisión que hace sentido incluso a los jugadores reales. De esta forma, nuestra propuesta logra cumplir con éxito los objetivos presentados.

CCS CONCEPTS

• **General and reference** → *Cross-computing tools and techniques*;
• **Computing methodologies** → *Machine learning*.

KEYWORDS

Neural Networks, MLP, Multi-Label Classification, Deck Recommendation, Mobile Games

1 INTRODUCCIÓN

Hoy en día la industria de los videojuegos ha crecido enormemente, resultando en uno de los hobbies más comunes en el mundo. En particular, los videojuegos competitivos son sumamente populares incluso entre jugadores casuales, lo que llevó a la aparición de una gran cantidad de videojuegos *mobile* de tipo **Multiplayer Online Battle Arena (MOBA).** Entre los exponentes más populares de este género se encuentra *Clash Royale*, un videojuego de estrategia en línea en el que cada jugador debe escoger un total de 8 cartas de entre las que tiene en su posesión para que conformen su mazo de batalla, el que posteriormente será utilizado dentro de una arena al enfrentar a su oponente. Para ganar la partida, los jugadores deben destruir la torre principal del oponente o intentar hacer el mayor daño posible a sus defensas.

Uno de los principales desafíos que enfrentan los jugadores es la gran cantidad de cartas para elegir (más de 100), por lo que se torna muy complicado escoger el mazo óptimo (o apropiado) para cada batalla en específico. Además, cada carta tiene características y estadísticas diferentes, lo que hace extremadamente complejo determinar aquellas que podrían maximizar las probabilidades de victoria. A lo anterior, se le debe sumar que cada jugador puede aplicar distintas estrategias a partir de un mismo mazo, y que puede existir más de un mazo que sea ventajoso contra otro(s).

En el flujo original el jugador no puede conocer las cartas que elegirá su próximo oponente, sin embargo, para aprovechar los datos recolectados, se cambiarán las reglas para que sí se puedan conocer las cartas a priori (para efectos de este proyecto). Dicho esto, el objetivo es entregarle al usuario una recomendación sobre las mejores cartas que podría elegir para ganar su próxima batalla, tomando en consideración factores tales como las cartas y niveles de su oponente. Esto tendría variadas aplicaciones en entornos competitivos, donde los jugadores se preparan de antemano para enfrentarse a todo tipo de oponentes en los torneos, haciendo válido el supuesto de conocer las cartas a priori.

2 ESTADO DEL ARTE

Actualmente existen varios papers que abordan temas similares al que se busca resolver, estando entre los más recientes *"Interpretable Contextual Team-Aware Item Recommendation: Application in Multiplayer Online Battle Arena Games"* [4], cuyo objetivo principal es realizar recomendaciones para un juego multijugador online en donde, dependiendo del personaje seleccionado por el jugador, se recomiendan ítems que podrán ser utilizados posteriormente en la partida. El problema abordado en dicho trabajo corresponde a que la cantidad de ítems disponibles es muy grande, y el tiempo que tienen para elegirlos es bastante pequeño, por lo que el sistema debía ser capaz de entregarle al usuario un listado de ítems que mejor funcionaran para la partida en cuestión. Esta recomendación depende de ciertos factores, tales como los personajes que están utilizando los enemigos/aliados y el rol que cumple cada jugador en sus equipos respectivos. Para llevar a cabo este modelo se utilizó un encoder que permite procesar las relaciones entre todos los atributos de entrada y luego, a través de una capa densa con función *Sigmoidal*[3], se entrega un ranking de los ítems favorables. Este trabajo entrega una base para las ideas aplicadas en la solución propuesta de nuestro proyecto, pues el problema que se busca resolver es similar.

3 SOLUCIÓN PROPUESTA

Se propone *RoyaleNet*, una red neuronal multicapa capaz de recomendarle a los usuarios un conjunto de cartas a partir del mazo

con el que jugará su oponente. El modelo debe recibir como *input* un vector binario v, en donde las posiciones que estén en 1 representarán las cartas que el oponente escogió para jugar y 0 en cualquier otro caso. Como se muestra en la **figura 1**, este vector se puede obtener como la concatenación de un conjunto de vectores c_{ij}, en donde el índice $i \in [1, 102]$ corresponde al *ID* de la carta y $j \in [1, 14]$ a su *nivel*.

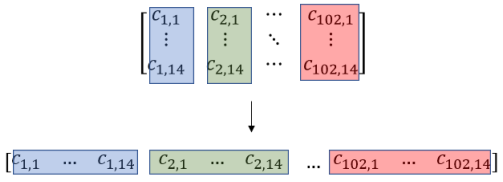

Figure 1: Transformación de la matriz de cartas en un vector de input colapsado.

La red neuronal está compuesta por un par de capas lineales con función de activación **ReLU** (la mejor al entrenar con grandes cantidades de datos)[3] y tendrá una salida de 1428 neuronas, las que, a través de una función de activación **Sigmoidal** (para decidir la utilidad de cada carta), serán transformadas en un vector de probabilidades. Este arreglo de probabilidades se ordenará de mayor a menor para poder obtener el *ranking* de las mejores 8 cartas, ya que su probabilidad indica la utilidad de esa carta dentro de las 8 del mazo al enfrentar al oponente ingresado en el input, considerando implícitamente su relación con las probabilidades de las otras cartas (es decir, la red aprende la utilidad conjunta de las cartas, no solamente la utilidad individual).

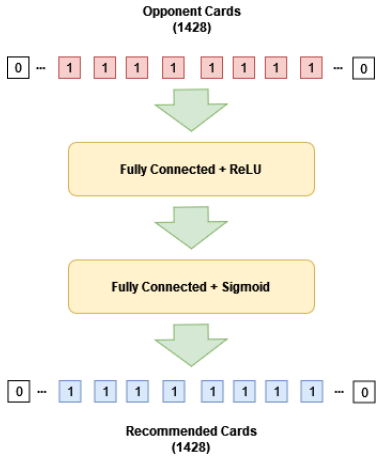

Figure 2: Arquitectura de *RoyaleNet*

Para el entrenamiento del modelo se utilizará el mazo del jugador ganador como el **valor real** que el modelo debe predecir, puesto que lo que se busca es recomendar los mazos que se sabe que lograron conseguir una victoria dentro del dataset, partiendo del mazo del jugador perdedor. Se ocupará una función de pérdida del tipo *Binary Crossentropy*, que es útil para problemas donde se deben elegir múltiples labels (en este caso las 8 cartas que conforman el mejor mazo deben tener label 1, y todas las demás 0)[2]. Cabe mencionar que esta función fue customizada para que aparte de calcular la *Binary Crossentropy* general, también considere un ponderador basado en la diferencia de coronas obtenidas en cada partida. Esta diferencia representa la calidad de la victoria, mientras mayor sea, se tiene que la victoria fue lograda con mayor facilidad. La idea de esto es que el modelo aprenda a elegir la mejor victoria cuando encuentre más de una forma de ganar entre los ejemplos del dataset.

Por último, como se mencionó más arriba, el vector de salida es un conjunto de cartas, las que posteriormente deben ser ordenadas por su utilidad contra el oponente actual. Finalmente, se retornan las 8 cartas con mayor probabilidad, representando el mazo que se espera sea ideal contra dicho oponente.

4 METODOLOGÍA UTILIZADA

Se realizó el pre-procesamiento de los datos en Google Colab utilizando las librerías Pandas y Numpy, en donde se descartaron varias columnas, conservando sólo las más relevantes para nuestro modelo (cartas utilizadas por ambos jugadores y cantidad de coronas obtenidas). Luego, con las herramientas anteriores, se procesaron los datos para que estos pudieran ser alimentados al modelo, el que fue implementado en un Jupyter Notebook local haciendo uso del framework *Keras* y *Tensorflow*. A continuación, se separaron los datos en dos componentes: X e Y, en donde X correspondía a las cartas que habían ocupado los perdedores e Y a las cartas que habían utilizado los ganadores de la partida en cuestión. De esta forma, para entrenar y evaluar el modelo se utilizaron las cartas del jugador perdedor (X) como input, y el output entregado corresponde al ranking de las cartas, permitiendo así compararlas con las que fueron realmente elegidas por el ganador (Y).

La fase de entrenamiento fue realizada sobre un dataset de partidas del juego, el que poseía alrededor de 32000 entradas. Al finalizar cada época, se evaluó el modelo sobre un dataset de validación de unas 8000 entradas, permitiendo así realizar el análisis de parámetros explicado en la siguiente sección. En cuanto al set de testing, se usó otro dataset del mismo tamaño que el de validación.

Para tener una referencia inicial respecto al rendimiento de nuestro modelo en el set de testing, se consideró una función de recomendación *Random* que entrega cartas al azar al momento de aplicarla sobre el dataset. La comparación entre los modelos se basó en la pérdida y precisión obtenidas, las que se calcularon utilizando *Binary Crossentropy* y *Binary Accuracy*, respectivamente. Finalmente, se puso el modelo en práctica frente a jugadores de *Clash Royale*, permitiendo recibir *feedback* sobre las recomendaciones de mazos realizadas.

5 EXPERIMENTOS

5.1 Dataset

El dataset utilizado corresponde a un resumen de partidas jugadas en *Clash Royale* obtenido a partir de la API oficial del juego por un usuario de la plataforma Kaggle[1]. Los datos corresponden a un csv, conteniendo información tal como las cartas usadas por el jugador que ganó y el que perdió, los puntos de vida de las torres de cada jugador, el nivel de las cartas, etc. El dataset cuenta con información sobre 30 millones de partidas aproximadamente, teniendo un peso de 22GB. Para minimizar los datos en memoria, se eliminaron los atributos que no fueron usados por nuestro modelo. En concreto, se dejaron solamente los datos sobre las cartas y sus niveles, así como las coronas obtenidas por cada jugador en cada partida (que representan de cierta forma lo eficiente que fue la estrategia ganadora).

Al realizar la exploración de los datos se analizó la distribución de victorias en función de las cartas. En particular, se logró evidenciar que no existe una solución trivial donde las mismas 8 cartas sean las mejores ante cualquier oponente, es decir, que el juego está bien balanceado ante su gran variedad de mazos posibles **(figura 3).**

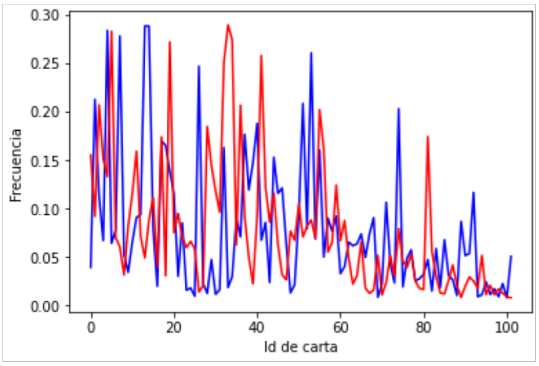

Figure 3: Distribución de victorias (azul) y derrotas (rojo) por id de carta.

Otro objetivo explorado fue la corroboración de que las partidas son justas, de forma que la victoria puede ser atribuida al mazo y estrategia y no al nivel del jugador (el que está relacionado a la cantidad de tiempo dedicado al juego) **(figura 4).**

5.2 Análisis de Parámetros

Se evaluó el modelo usando diferentes parámetros con la finalidad de encontrar los valores que maximizaran el rendimiento obtenido con el dataset de validación. Para lograr esto, se fueron realizando variaciones en el número de capas ocultas y neuronas por capa de la red **(tabla 1).**

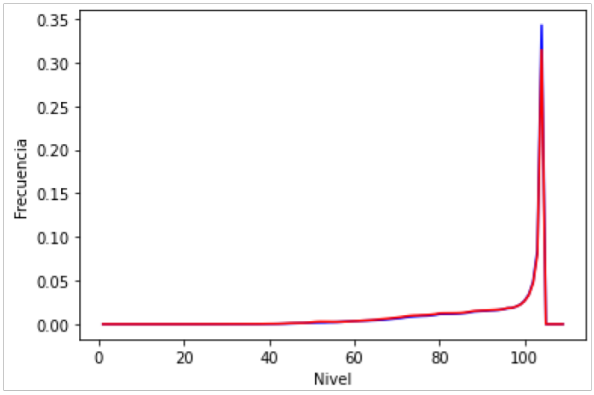

Figure 4: Distribución de la suma de niveles del mazo por victoria (azul) y pérdida (rojo).

	Acc	Loss
RoyaleNet @3000	0.9850	0.0292
RoyaleNet @10000	0.9853	0.0293
RoyaleNet @3000+2000	0.9843	0.0293
RoyaleNet @2000+5000	0.9842	0.0292
RoyaleNet @2000+5000+2000	0.9843	0.0294

Table 1: Resultados del modelo RoyaleNet variando el número de capas ocultas y neuronas.

Se evaluaron las siguientes versiones del modelo: *RoyaleNet* @3000, @10000, @3000 + 2000, @2000 + 5000, @2000 + 5000 + 2000. En cada versión, los números que le siguen al *@* indican el número de neuronas que hay en una capa oculta, las que a su vez están separadas por el símbolo *+*. Como se puede observar en la **tabla 1**, el modelo que obtuvo el mejor *accuracy* fue *RoyaleNet @10000*, es decir, los parámetros que nos permitieron optimizar el desempeño del modelo fue dejar sólo 1 capa oculta con 10000 neuronas.

Por otro lado, también se realizó un análisis sobre el número de épocas utilizadas para el entrenamiento. El principal resultado que se pudo observar es que desde la segunda época en adelante el modelo dejaba de aprender y tendía a sobreajustarse (*overfitting*) **(figura 5).**

5.3 Resultados

Se evaluaron los mismos modelos anteriores con el dataset de testing, agregando además el modelo *Random*. Los resultados se pueden apreciar en la **tabla 2**.

Observando los resultados, se puede ver que el modelo propuesto supera por bastante al recomendador aleatorio, lo que quiere decir que el sistema logró aprender lo suficiente como para recomendarle cartas útiles al usuario. Por otra parte, el modelo que obtuvo los

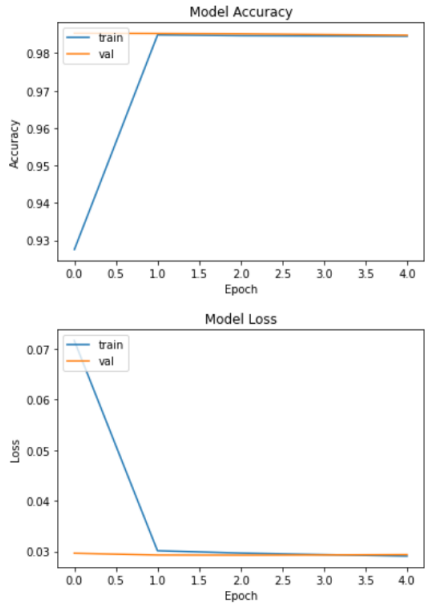

Figure 5: Precisión y Pérdida del modelo según el número de épocas de entrenamiento.

	Accuracy	Loss
Random	0.0056	0.973
RoyaleNet @3000	0.984	0.0289
RoyaleNet @10000	0.985	0.029
RoyaleNet @3000+2000	0.984	0.0284
RoyaleNet @2000+5000	0.984	0.029
RoyaleNet @2000+5000+2000	0.984	0.029

Table 2: Resultados de los distintos modelos evaluados en el set de testing.

mejores resultados fue nuevamente *RoyaleNet @10000* con un 0.985 de *accuracy*, demostrando una precisión muy alta al momento de recomendar mazos.

5.4 Feedback de Jugadores

Para poder corroborar que el resultado obtenido fuese útil y tenga sentido como aplicación real, se puso en práctica el modelo frente a jugadores de *Clash Royale*, preguntándoles su opinión sobre las recomendaciones en persona **(figura 6)**. Los resultados obtenidos fueron favorables, acarreando opiniones positivas de parte de los jugadores.

Algunos **testimonios de los usuarios** se muestran a continuación:
"El mazo es de bajo costo, permite atacar mucho con el Montapuerco"
“Me ha tocado jugar con personas que usan ese mazo”
"Las flechas son letales contra el ejército de esqueletos y el barril de duendes"

Figure 6: Ejemplo sobre una recomendación de *RoyaleNet*.

6 CONCLUSIONES

En este trabajo se introdujo a *RoyaleNet*, un sistema basado en una red neuronal que es capaz de aprender qué cartas son eficientes para combatir contra un determinado mazo oponente dentro del videojuego *Clash Royale*. El sistema desarrollado es una alternativa a resolver un respectivo problema de optimización o de teoría de juegos relacionado a la elección de una baraja óptima, correspondiendo a una versión relajada de este. Los experimentos y feedback de los jugadores indican una gran precisión al momento de ponerlo en práctica, logrando satisfacer el objetivo propuesto en este proyecto. En cuanto a trabajo futuro, se espera que *RoyaleNet* sea extendido para ser capaz de explicar la elección de cartas realizada, haciendo uso de mecanismos más avanzados de auto-atención dentro del modelo. Por otro lado, también existe una potencial aplicación en el contexto de los desarrolladores del juego: ser capaz de detectar cartas que sean muy fuertes o débiles en comparación al resto, apoyando así a cumplir el objetivo de mantener un equilibrio en la dificultad del videojuego. Para finalizar, una forma de poder mejorar un poco más el algoritmo es tomando en cuenta la historia que tiene un jugador en específico con ciertas cartas y, a partir de esto, poder encontrar las cartas que mejor le funcionen al usuario en sí, haciendo el proceso más personalizado y, por ende, con mejores resultados.

REFERENCES

[1] [n. d.]. Clash Royale kaggle dataset. https://www.kaggle.com/bwandowando/clash-royale-season-18-dec-0320-dataset. Accessed: 2021-12-10.
[2] Katarzyna Janocha and Wojciech Marian Czarnecki. 2017. On Loss Functions for Deep Neural Networks in Classification. (2017). arXiv:1702.05659 [cs.LG]
[3] Tomasz Szandała. 2020. Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. (10 2020).
[4] Andrés Villa, Vladimir Araujo, Francisca Cattan, and Denis Parra. 2020. Interpretable Contextual Team-Aware Item Recommendation: Application in Multiplayer Online Battle Arena Games. (2020). https://doi.org/10.1145/3383313.3412211