Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

IIC3633 — Sistemas Recomendadores — 2' 2021

Informe de Avance Proyecto

Integrantes:

- Benjamín Farías
- Benjamín Lepe
- Juan Romero

Contexto, Problema y Objetivos

Hoy en día, la industria de los videojuegos ha crecido enormemente, resultando en uno de los hobbies más comunes en el mundo. En este proyecto nos centraremos específicamente en *Clash Royale*, un videojuego de estrategia en línea para dispositivos móviles en el que cada jugador debe escoger 8 cartas entre las que tiene en su posesión para que conformen su mazo, el que posteriormente será utilizado dentro de un campo de batalla. Para ganar la partida los jugadores deben tratar de destruir la torre principal del oponente o intentar hacer el mayor daño posible.

Uno de los principales problemas es que existen demasiadas cartas para elegir (más de 100), por lo tanto, es muy difícil para los usuarios escoger las óptimas (o cercanas a óptimas) para cada batalla en específico. Además, cada carta tiene características diferentes, lo que hace extremadamente complejo saber cuales serán las cartas que maximizarán nuestras probabilidades de ganar. A lo anterior, debemos sumarle que cada jugador puede usar distintas estrategias con un mismo mazo, y que puede existir más de un mazo que es bueno contra otro.

En el flujo original del juego, el usuario 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. Dicho esto, se buscará entregarle al usuario una recomendación sobre las mejores cartas que podría elegir para ganar su próxima batalla, tomando en consideración factores como el nivel de sus cartas y las de su oponente. Esto tendría aplicaciones en términos competitivos, donde los jugadores se preparan de antemano para enfrentarse a todo tipo de oponentes antes de entrar en las batallas en sí, por lo que el supuesto de conocer las cartas del oponente a priori aplicaría en dicho caso.

Análisis de los Datos

El dataset usado corresponde a un resumen de partidas jugadas en Clash Royale que se obtuvo a partir de la API oficial del juego por el usuario BwandoWando y está actualmente compartido en la plataforma de Kaggle. 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 eliminarán atributos que no sean usados para nuestra propuesta. En concreto, se dejarán 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).

En una primera instancia, interesa analizar la distribución de victorias en función de las cartas. En particular, se busca 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. Para el análisis se seleccionó una muestra correspondiente al 10 % del total de datos. Los resultados del análisis se pueden ver en la **figura 1.**

Se observa que la carta más usada no supera el tercio del total de usos, es decir, si bien hay cartas populares, estas no rompen el equilibrio del juego. Aún así es importante notar que algunas cartas populares en victorias, son al mismo tiempo impopulares en derrotas (y vice versa), lo que da cuenta de que existen mazos que son en general mejor que otros, aunque no exista uno que sea ideal.

Otro aspecto a considerar fue si las partidas eran justas, cosa que la victoria sea atribuida al mazo y estrategia y no al nivel del jugador. Esta distribución se puede apreciar en la **figura 2**.

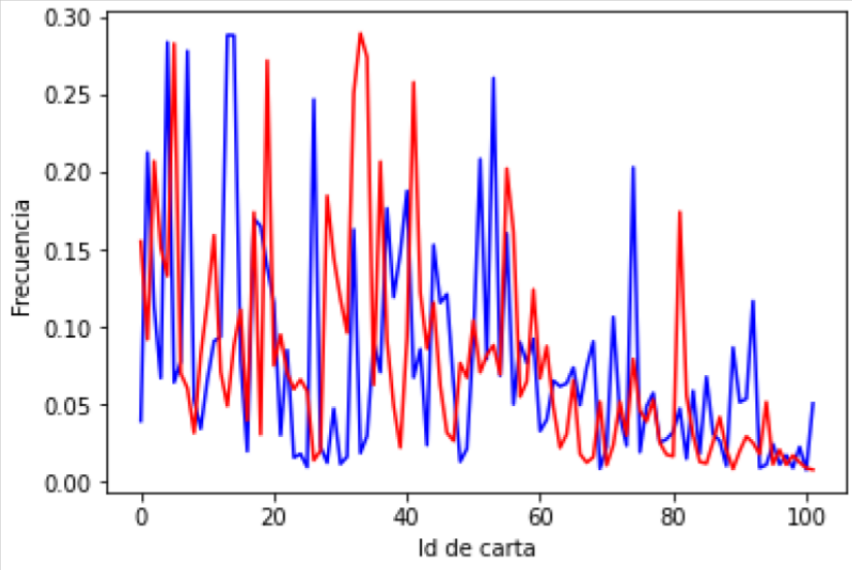

Figura 1: Distribución de victorias (azul) y derrotas (rojo) por id de carta.

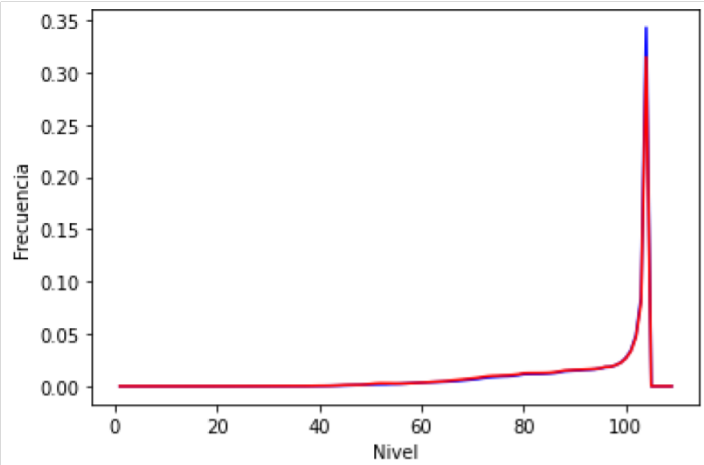

Figura 2: Distribución de la suma de niveles del mazo por victoria (azul) y pérdida (rojo).

Como se puede ver, las peleas fueron justas, ya que por cada nivel de los jugadores existe una cantidad similar de victorias y derrotas.

Solución Propuesta

Se propone *RoyaleNet* una red neuronal capaz de recomendarle a los usuarios un conjunto de cartas a partir del mazo con el que jugará su oponente, considerando cartas y niveles. El modelo debe recibir como *input* un vector v que contenga solo 1s y 0s, 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 3**, 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*.

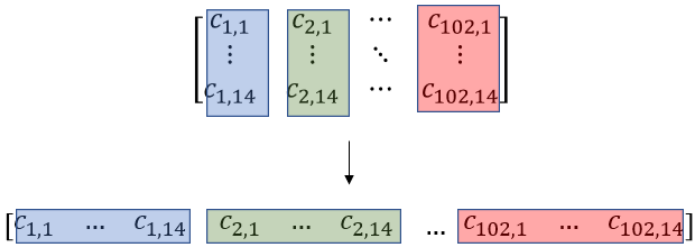

Figura 3: Transformación de la matriz de cartas en un vector de input

La red neuronal estará compuesta por un par de capas lineales con función de activación **ReLU** (la mejor al entrenar con grandes cantidades de datos) y tendrá una salida de 1428 neuronas, las que, a través de una función de activación **Sigmoide** (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 usar esa carta dentro de las 8 del mazo contra el 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).

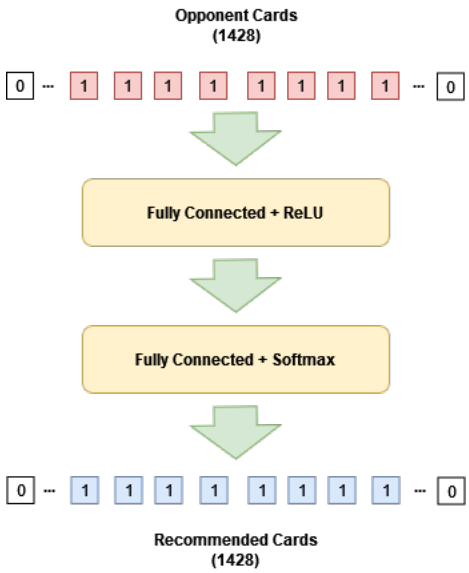

Figura 4: Arquitectura del modelo propuesto

Luego, para la parte del entrenamiento se utilizará al mazo del jugador ganador como el valor real que el modelo debe aprender, puesto que lo que se busca es recomendar las cartas que se sepa lograron conseguir una victoria dentro del dataset. Además, se ocupará una función de pérdida del tipo *Binary Crossentropy*, que sirve para problemas donde se deben elegir múltiples clases (en este caso las 8 cartas). Cabe mencionar que esta función fue customizada para que aparte de calcular la *Binary Crossentropy* considere un ponderador basado en la diferencia de coronas obtenidas entre el ganador y perdedor. Esta diferencia representa la calidad de la victoria, mientras mayor es significa 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.

Por último, como se mencionó más arriba, el vector de salida es un conjunto de cartas, las que posteriormente deben ser ordenadas por su probabilidad de victoria. Puede existir el caso en que alguna de las 8 cartas recomendadas aún no haya sido adquirida por el jugador dentro del juego, en cuyo caso simplemente se debe omitir la carta y se toma la carta que viene después en el ranking para completar las 8. Esto sería implementado post-predicción, ya que no afecta al modelo.

Avances Realizados y Problemas Encontrados

Se realizó el pre-procesamiento de los datos en Google Colab utilizando Pandas y Numpy, en donde descartamos varias columnas, quedándonos con las más relevantes para nuestro modelo (mencionadas en la sección de análisis de datos). Luego, con las herramientas anteriores, se procesaron los datos para que estos pudieran ser utilizados por el modelo, el que fue implementado en un Jupyter Notebook local con *Keras* y *Tensorflow*.

Se alcanzó a realizar un entrenamiento con 10000 ejemplos, pero queda pendiente hacerlo a mayor escala, ya que eran demasiados datos que se debían procesar y, por ende, es necesario optimizar las funciones lo más posible y luego agrupar los datos de entrada por batches para que quepan en memoria. Aún así, evaluamos el modelo con este entrenamiento a baja escala y lo comparamos con un recomendador aleatorio.

En general, los problemas encontrados de momento corresponden al gran tamaño del dataset completo, lo que va a requerir una estrategia eficiente de entrenamiento para el modelo final.

Resultados

Para evaluar el desempeño de la red se utilizó de momento la función de pérdida mencionada anteriormente. A continuación se muestran los resultados:

Modelo	**Loss**
Random	0.8865
RoyaleNet (sin entrenar)	0.8829
RoyaleNet (entrenada por 10 épocas)	0.029

Plan de avance

- Generar una estrategia eficiente para cargar los datos mediante batches, de forma que se pueda entrenar el modelo con más ejemplos. Además, separar los datos entre training, validación y testing y quizás agregar una métrica para la precisión aparte de la loss.
- En base a los resultados que se obtengan, optimizar los parámetros del modelo y pensar en agregar mejoras o funcionalidades adicionales dependiendo de su viabilidad. Una posibilidad sería tratar de explicar las recomendaciones agregando nuevos componentes al modelo, mejorando los objetivos de *justification* y *transparency* dentro de los sistemas recomendadores.
- Implementar modelos clásicos para intentar comparar su rendimiento contra el nuestro, en caso de que sea posible aplicarlos sobre este problema.
- Testear las recomendaciones con jugadores reales para obtener feedback en la práctica.

Bibliografía

- **Interpretable Contextual Team-aware Item Recommendation, Application in Multiplayer Online Battle Arena Games:**
  `https://dparra.sitios.ing.uc.cl/pdfs/TTIR-2020.pdf`
- **Predicting Human Card Selection in Magic The Gathering with Contextual Preference Ranking:**
  `https://arxiv.org/pdf/2105.11864v2.pdf`
- **Q-DeckRec, A Fast Deck Recommendation System for Collectible Card Games:**
  `https://arxiv.org/pdf/1806.09771v1.pdf`
- **DouZero, Mastering DouDizhu with Self-Play Deep Reinforcement Learning:**
  `https://arxiv.org/pdf/2106.06135v1.pdf`
- **Clash Royale S18 Ladder Dataset:**
  `https://www.kaggle.com/bwandowando/clash-royale-season-18-dec-0320-dataset`