Definición de perfiles de agentes inteligentes en el contexto del juego de cartas Toma6!

Sergio Barranco Aguilar

30 de junio de 2024

Resum– Este trabajo de fin de grado trata sobre el desarrollo de distintos agentes inteligentes para el juego de Toma 6! Este es un juego de cartas en el cual el objetivo es evitar acumular los puntos de las cartas. Para llevar a cabo este proyecto, se ha creado una interfaz por consola que permite jugar al juego y obtener los resultados de las partidas jugadas entre los modelos desarrollados. Para generar estos modelos, se ha tomado como base estudios aplicados a otros juegos de cartas, con el fin de crear unos propios para que sean capaces de jugar a este juego y enfrentarse tanto a jugadores humanos como a otros modelos ya entrenados. Se desarrollaron varios modelos utilizando diferentes tipos de algoritmos y mecanismos de aprendizaje, y posteriormente se analizaron los resultados obtenidos. Se concluyó que la mejor forma de jugar al juego es arriesgarse en las jugadas y ser calculador en cuanto a las probabilidades de que aparezcan otras cartas.

Paraules clau– Aprendizaje computacional, QLearning, Agentes Inteligentes, Algorítmica

Abstract– This final degree project is about the development of different intelligent agents for the game Toma 6! This is a card game in which the objective is to avoid accumulating card points. To carry out this project, an interface has been created per console that allows you to play the game and obtain the results of the games played between the developed models. To generate these models, different agents from other projects have been studied, in order to create our own so that they are capable of playing this game and facing both human players and other already trained models. Several models were developed using different types of algorithms and learning mechanisms, and the results obtained were subsequently analyzed. It was concluded that the best way to play the game is to take risks in your plays and be calculative about the probabilities of other cards appearing.

Keywords– Machine learning, QLearning, Intelligent Agents, Algorithmic

1 INTRODUCCIÓN

Poco a poco, estamos viendo un auge en los medios de comunicación sobre los grandes avances en el campo de la inteligencia artificial (IA). Sin embargo, estas tecnologías han estado presentes en nuestras vidas desde hace mucho tiempo, particularmente en el ámbito de los videojuegos. En este contexto, se programa un agente inteligente para enfrentarse a un jugador humano, eliminando la necesidad de otra persona para jugar. Al principio, todas estas IAs no eran muy complicadas y simplemente seguían patrones fijos. No obstante, a medida que los videojuegos evolucionaban, también lo hacían las IAs, llegando a desarrollar agentes complejos como los utilizados en juegos de ajedrez o en videojuegos como Dota [1] y StarCraft 2 [2], donde se deben tomar decisiones en tiempo real y múltiples factores pueden influir en el resultado de la partida.

1.1. Reglas del juego

Este proyecto se enfocará en desarrollar una IA/Agente Inteligente que juegue al juego de Toma 6! [3]. Toma 6! es un juego de cartas cuyo objetivo es evitar obtener puntos negativos. Para jugar se necesitan 104 cartas, numeradas del 1 al 103, con una puntuación llamada "bueyes"que repre-

• E-mail de contacte: 1606048@uab.cat
• Menció realitzada: Computació
• Treball tutoritzat per: Jorge Bernal del Nozal (Àrea de Ciències de la Computació i Intel·ligència Artificial)
• Curs 2023/24

sentan los puntos que los jugadores deben intentar evitar. ¿Cómo se juega?

Para comenzar, se reparten 10 cartas a cada jugador y se colocan 4 cartas en el centro, formando 4 filas. Cada jugador debe seleccionar una de sus 10 cartas sin mostrarla a los demás hasta que todos hayan hecho su elección. Luego, las cartas seleccionadas se revelan simultáneamente y se colocan en las filas siguiendo las reglas del juego.

Una vez que todos los jugadores hayan escogido sus cartas, se revelarán y jugarán de menor a mayor. Estas cartas se colocarán en las 4 filas del centro del tablero siguiendo las siguientes normas: Si la carta jugada es menor que cualquiera de las cartas al final de las 4 filas, el jugador que haya jugado dicha carta deberá escoger una de las filas para sustituirla con su carta. Al hacer esto, el jugador obtendrá los ”bueyes”, es decir, la puntuación negativa de todas las cartas que conformaban la fila seleccionada.

Cuando el jugador recoja dicha fila, la carta jugada se colocará en esta fila como la nueva carta inicial.

En caso contrario, la carta se colocará a la derecha de la carta con el valor más cercano. Por ejemplo, si se juega un 35 y hay una fila con un 15 y otra con un 30, se pondrá la carta en la fila del 30, ya que es la más próxima al 35. Esta acción solo se podrá realizar si la fila no contiene más de 5 cartas. En caso de que la carta se juegue en una fila que ya tiene 5 cartas, el jugador recibirá toda la puntuación negativa de esa fila. Tras recoger las cartas de la fila, el jugador colocará su carta como la primera carta de la nueva fila.

El proceso de jugar cartas se repite hasta que se acaben las 10 cartas repartidas. Luego, se barajan todas las cartas nuevamente, se reparten y se colocan las filas de nuevo. Este ciclo se repite hasta que uno de los jugadores obtenga un total de 66 puntos negativos o más. El jugador con la menor puntuación negativa será el ganador de la partida.

1.2. State of the Art

Para afrontar este proyecto, es necesario investigar qué tipo de aprendizaje o algoritmo sería más adecuado para la IA. En la literatura, se han encontrado diversos trabajos que persiguen objetivos similares a los de este proyecto.

El primer documento relevante es ”Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass”[4], que proporciona una visión general sobre los agentes inteligentes y los distintos enfoques que se pueden tomar en los juegos de cartas. También discute diversos tipos de algoritmos, como el Monte Carlo [5], el tree search [6] y el reinforcement learning [7], entre otros. En resumen, este documento será de gran utilidad para obtener una idea clara sobre cómo enfocar la IA y los diversos modelos de entrenamiento que se pueden aplicar.

Por otro lado, también se ha consultado un artículo titulado ”Teaching a Neural Network to Play Cards”[8], en el cual se utiliza el reinforcement learning para que la IA aprenda a jugar desde cero. Este artículo menciona los problemas que enfrentó el autor durante el desarrollo de la IA y describe cómo utilizó el reinforcement learning para superarlos. Por lo tanto, este artículo se tendrá en consideración para el proyecto, evaluando si este enfoque puede ser funcional en el juego de Toma 6! y si sería la mejor opción.

También se ha seguido con atención discusiones en foros como ”Best techniques for an AI of a card game”[9], donde varias personas debatían y compartían sus puntos de vista sobre cuál es el mejor método para las IAs en los juegos de cartas. En estos debates, se destacó el método Monte Carlo como uno de los mejores, ya que permite a la IA calcular movimientos futuros y tomar mejores decisiones. Con este método, se calcula la probabilidad de las cartas que pueden tener los oponentes y se juega con esta probabilidad para escoger el mejor movimiento a futuro y así ganar. Por lo tanto, este foro será útil para considerar distintos métodos y ver opiniones de personas que ya han experimentado con ellos en juegos de cartas.

También se buscó documentación de trabajos similares y se encontró el artículo “Automated Playtesting in Collectible Card Games Using Evolutionary Algorithms” [10], en el cual se creó un modelo de IA con algoritmos de autoaprendizaje que aprenden de sus éxitos pasados. Este modelo se entrenó para jugar al juego de cartas HearthStone [11]. Aunque este juego es más complejo que el que se desarrollará para este trabajo de fin de grado, el artículo servirá para aportar ideas y formas de aplicar los modelos.

Durante la elaboración del proyecto, se investigaron nuevas formas de enfocar la IA, una vez que ya se habían creado algunos de estos modelos y se tenía más experiencia en el enfoque necesario para el juego. Debido a la experiencia recolectada, se buscaron métodos de entrenamiento que pudieran ofrecer mejores resultados que los obtenidos con los modelos iniciales.

Los modelos que se investigaron incluyen el Q-learning [12], un modelo de reinforcement learning que se basa en generar una tabla de valores de ponderación de acciones según el entrenamiento realizado y las recompensas obtenidas previamente. Este modelo funciona poniendo a jugar a una IA con un comportamiento aleatorio y premiándola con una puntuación según las acciones que realice durante el juego.

La puntuación puede ser mayor o menor dependiendo de la importancia del beneficio de la acción realizada. Por ejemplo, si se quisiera entrenar una IA con Q-learning para jugar al fútbol, se le premiaría con un punto por pasar el balón a otro compañero, pero se le otorgarían 10 puntos si marcara un gol en la portería rival. Además, hay modelos que utilizan puntuaciones negativas para indicar que ciertas acciones llevan a resultados desfavorables, como un gol en propia.

2 OBJETIVOS DEL PROYECTO

El objetivo de este trabajo de fin de grado será desarrollar un Agente Inteligente capaz de enfrentarse a un jugador humano en el juego de cartas Toma6!, ajustando la forma en que la IA actuaría dentro del juego. Para que el proyecto sea satisfactorio, se impondrán los siguientes objetivos. Estos serán los que se desarrollarán a lo largo de toda la instancia del TFG y darán forma al producto final y al objetivo final del proyecto.

2.1. Objetivo principal

Programar un Agente Inteligente para emular el comportamiento humano en el juego Toma6!. Este se programará para que pueda adoptar distintos tipos de comportamientos, como: agresivo, pasivo, calculador, etc.

- **Desarrollo de la primera versión de la IA:** Se desarrollará una IA simple para que sepa jugar al juego de forma lógica.
- **Desarrollo de IA con distintos comportamientos:** Se harán modificaciones a la IA base para que pueda adoptar distintos comportamientos.
- **Desarrollo de una primera versión del juego:** Como primer paso, se desarrollará una primera versión del juego con una interfaz visual por consola y sin IA.

3 Requisitos de Software/Hardware

Por la parte del Software se utilizará Visual Studio, ya que es una de las mejores opciones para programar en todas las variantes de C, ya que esta es la primera que se enseña en la carrera.

Requisitos	
	Visual Studio
Versión Sistema Operativo	Windows10(en adelante),64-bit
CPU	X64 arquitectura SSE2
GPU	DX10, DX11, y DX12
Memoria RAM	8GB

4 Riesgos del proyecto

Caso	Riesgo	Plan	Probabilidad
Mala planificación	Alt	Replantear toda la estructura temporal para poder hacer frente al proyecto y entregar a tiempo	Media
Problemas con la integración de la IA	Alt	Intentar de cambiar de modelo	Media
No poder generar más de un perfil diferente de IA	Bajo	Hacer un modelo que tenga mucho *winrate*	Media
Modelos con dificultades para ganar	Medio	Destinar más tiempo a mejorar los modelos	Media
El modelo se rompe o se pierde	Alto	Tener backups de los modelos o destinar más tiempo a rehacerlos	Baja
El ordenador se rompe	Alto	Mirar de trabajar en otro dispositivo o repararlo	Baja

5 Metodología

Para mantener el ritmo de trabajo se utilizará la metodología *Agile Kanban* [13] de uso bastante extendido, que permite realizar un seguimiento del trabajo hecho y que hay por hacer.

6 Planificación Temporal

Trabajo	Inicio	Fin
Estudiar Modelos de IA	28/02	06/03
Implementar en código Toma6!	07/03	18/03
Programar IA Simple	19/03	10/04
Revisar IA/Modificar	11/04	17/04
Explorar nuevos modelos de IA	3/05	16/05
Analizar resultados IA	17/05	23/05
Pruebas finales y cierre de prototipo	24/05	14/06
Documentación del proyecto	15/06	28/06

7 Desarrollo

Para abordar este proyecto, se han desarrollado los siguientes puntos:

7.1. Juego en Terminal

Para comenzar a desarrollar los agentes inteligentes, es necesario primero establecer una base donde estos puedan realizar las acciones que se llevarían a cabo en una partida normal de Toma6!.

Se ha creado el juego desde cero en Visual Studio [15], implementando las siguientes clases para hacer funcional el juego:

- ***Card***: Esta clase actúa como representación de las cartas del juego, almacenando el número de la carta y su valor. También guarda el ID de los jugadores que han jugado esa carta para calcular quién ganará los puntos.
- ***Deck***: Representa la baraja o pila de cartas del juego. Almacena las 104 cartas y proporciona funciones para repartirlas de manera adecuada según las reglas del juego. Por ejemplo, las cartas terminadas en 5 valen 2 puntos, las terminadas en 0 valen 3 puntos, las múltiplos de 11 valen 5 puntos, y el 55, que cumple con dos de estas reglas, vale 6 puntos.
- ***Hand***: Clase que representa la mano de un jugador, almacenando las cartas que tiene disponibles.
- ***Player***: Clase que representa al jugador, guardando su puntaje, nombre y su mano (objeto de la clase Hand).
- ***AI***: Clase que representa a las inteligencias artificiales que jugarán contra otros jugadores o entre sí. Estas clases utilizan datos de la partida y aplican diferentes modelos.

Estas clases se integran en el juego para simular una partida completa. El juego se muestra al jugador en una interfaz de consola, como se muestra en la Figura 1, seguido de un breve tutorial en la Figura 2, donde se explica el significado de los números en las cartas. Además, en la Figura 2 se le pide al jugador que asigne a un jugador como IA y el tipo de IA. Si no se selecciona una IA, se solicita el nombre del jugador. La Figura 3 muestra cómo sería el tablero de juego y la mano del jugador con sus respectivos puntos.

En este modo de testing, también se ha implementado una función para generar datos necesarios para algunos modelos que requieren entrenamiento. Por ejemplo, hay opciones para guardar datos para modelos como Q-learning o simplemente resultados. Además, se puede especificar para qué jugador se deben guardar los datos.

Una vez finalizadas las X partidas de entrenamiento, los datos se guardarán en un documento dentro de la carpeta de datos, utilizando el nombre del tipo de datos guardado. Si los resultados son satisfactorios, se guarda con un nombre diferente para poder generar más archivos sin sobrescribir los anteriores.

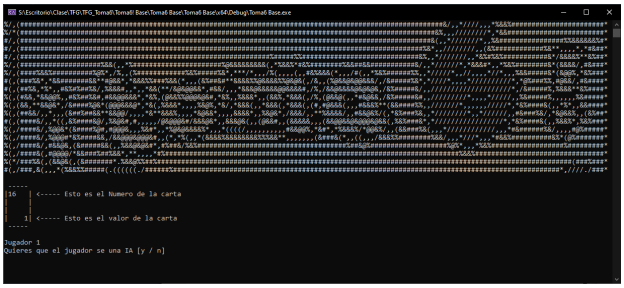

Fig. 1: Página principal

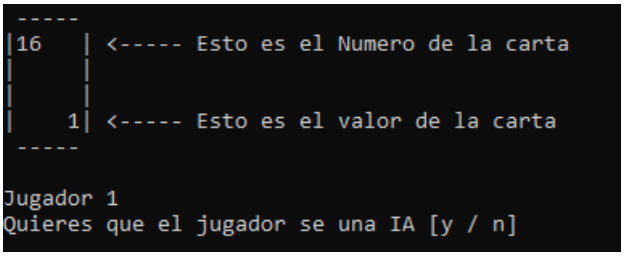

Fig. 2: Pequeño tutorial y configuración de Jugadores

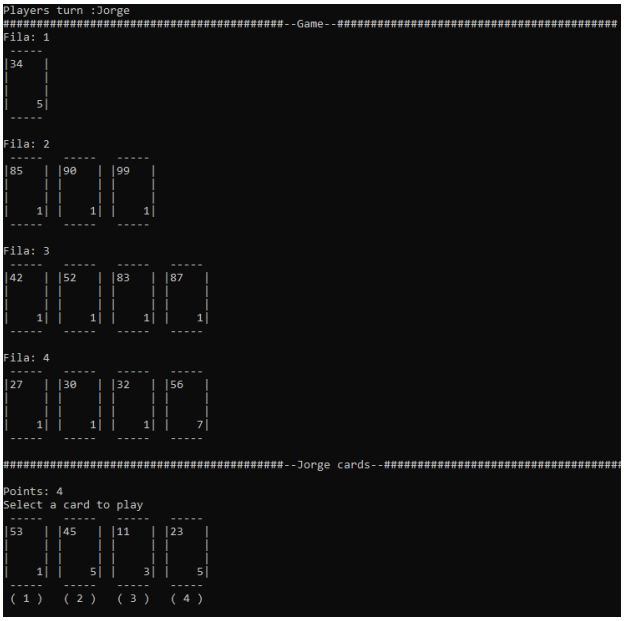

Fig. 3: Pantalla de juego

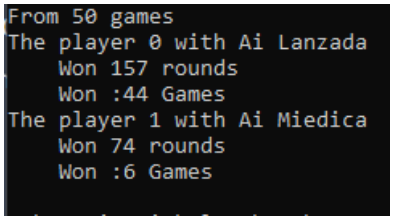

Fig. 4: Información del modo testing.

7.2. Agentes Inteligentes

Como se ha mencionado, este proyecto se centra en la creación de agentes inteligentes para el juego Toma6!. Se han desarrollado varios modelos de agentes inteligentes con enfoques distintos. Todos estos agentes comparten una función común: al descartar una columna, seleccionan la de menor valor para minimizar la cantidad de puntos obtenidos.

Estos son los agentes inteligentes integrados en este proyecto:

7.2.1. Agente Miedica

El primer agente desarrollado fue el Agente Miedica. Este agente es simple y adopta un comportamiento cauteloso similar al de una persona que evita riesgos en la partida. Siempre juega la carta más pequeña disponible para asegurarse de no llevarse una columna completa de cartas, ya que es poco probable que juegue una sexta carta.

Sin embargo, este comportamiento conservador hace que el Agente Miedica no sea muy efectivo, ya que termina acumulando puntos innecesarios al limpiar las filas con frecuencia, lo que facilita a los demás jugadores.

7.2.2. Lanzada

El segundo agente desarrollado, Lanzada, adopta un enfoque opuesto al de Miedica, pero con una estrategia más calculada. Este agente disfruta del riesgo y está dispuesto a ”tirarse a la piscina”para interferir con otros jugadores, jugando la carta más cercana al valor presente en la fila. Esta estrategia aumenta las posibilidades de jugar una carta sin llevarse toda la fila.

Este modelo de juego ha demostrado ser el más exitoso en términos de *winrate* contra otros modelos implementados.

7.2.3. MonteCarlo

Este agente juega utilizando probabilidades. Calcula la probabilidad de jugar una carta entre las cartas disponibles en su mano y las cartas en la fila. La fórmula utilizada para calcular esta probabilidad es la siguiente:

$$\frac{(Nc - Cj)}{CartasTotales} = Probabilidad \tag{1}$$

Donde:

- *Nc* es el número de cartas entre la carta a jugar y la carta más cercana en la fila. Por ejemplo, si se juega la carta número 78 y la carta más cercana en la fila es el 72, entonces $Nc = 5$.
- *Cj* representa las cartas que ya se han jugado durante la ronda actual.
- *CartasTotales* es el número total de cartas en el juego.

El agente pondera estas probabilidades junto con el valor actual de la fila para decidir qué carta jugar. De esta manera, busca maximizar la probabilidad de jugar una carta sin llevarse toda la fila, similar al enfoque del agente Lanzada, haciendo que otros jugadores puedan llevarse la fila si juegan una carta con una mejor probabilidad de no llevarse la fila.

7.2.4. Aleatoria

Este agente, como su nombre indica, juega una de las cartas disponibles de manera aleatoria. La utilidad de este agente radica en generar datos para la creación de un dataset. También podría entenderse como un jugador novato que está aprendiendo probando diferentes estrategias. Este sería el primer paso antes de aprender de los modelos entrenados.

7.2.5. Q-learning

QLearning es un modelo de aprendizaje reforzado donde la IA es recompensada por tomar decisiones correctas o favorables. Se le otorga una recompensa positiva cuando no obtiene puntos y no se le recompensa cuando obtiene muchos puntos. Este modelo entrena con el modelo Aleatorio para generar datos y ponderar estos por medio de recompensas.

- **Base:** Este modelo recibe una recompensa si no recibe puntos o si recibe 2 puntos o menos. Los parámetros de este modelo están especificadas para cada una de las particiones que se han realizado del mazo total de cartas. Por ejemplo, si se juega una carta y la media del tablero está en 40, se le otorgará una recompensa en la partición correspondiente a donde esté situado el valor 40. Este modelo utiliza estas particiones para tomar decisiones estratégicas durante el juego.

- **Cartas:** Al observar que el modelo anterior no era efectivo, se decidió cambiar el parámetro de partición para que fuera específico de cada carta en lugar de la media del tablero. Ahora, en lugar de basarse en la media del tablero, el parámetro se basa en las 4 cartas presentes en el tablero. Además, se introdujo un nuevo concepto de recompensa que otorga un punto adicional a la carta que se juega. Por ejemplo, consideremos la carta 40 mencionada anteriormente y las cartas en el tablero son 80, 35, 1 y 10. Si la carta 40 se juega y no se obtienen puntos, entonces la carta 40 recibiría una recompensa en las cartas 80, 35 y 1, siendo la carta 35 la que recibiría más puntos debido a que es la que más cercana está a la carta 40.

- **CartasCount:** Este modelo es una evolución del modelo anterior, mejorándolo al añadir una dimensión extra que considera el número de cartas en cada fila del tablero. Siguiendo el mismo ejemplo del modelo anterior con la carta 40 y las mismas cartas en el tablero, la carta 40 recibiría puntos basados en el número de cartas presentes en cada fila del tablero.

7.2.6. Quimera

Finalmente, se desarrolló un agente inteligente que combina los dos modelos que más se acercaron al agente Lanzada: el agente Q-learning CartasCount y el agente MonteCarlo.

Esta combinación ha dado como resultado un agente que utiliza la ponderación de cartas obtenida mediante Qlearning en CartasCount, junto con la ponderación basada en la probabilidad de que una carta específica se juegue entre la carta de la fila y la que se considera jugar. Con estas estrategias combinadas, este agente es el que más se aproxima al rendimiento del agente Lanzada.

7.3. Modo de testing

Para experimentar con las IAs y obtener datos y estadísticas de estas, se han proporcionado parámetros para realizar un número X de partidas automáticas contra las IAs preseleccionadas. Como se muestra en la Figura 4, se visualiza un display con el número de partidas jugadas y las estadísticas de cada IA, incluyendo el número de rondas ganadas y el total de partidas ganadas.

8 DATASET

Para desarrollar un agente inteligente con altas probabilidades de ganar, se implementó una función dentro del juego para guardar datos en un archivo de texto (TXT). Estos datos se formatean posteriormente para crear un dataset que pueda ser utilizado en el entrenamiento y evaluación de los agentes.

Para tratar los datos, primero se extraen los valores de cada partida fila por fila. Cada fila representa una partida completa. Luego, se dividen los datos usando '$' como separador. Los ganadores de cada ronda se identifican utilizando el separador ")". Se almacenan las manos de los jugadores usando '/' como separador. Finalmente, los turnos se separan uno a uno con '}', y se guarda la carta jugada por cada jugador usando ']' como separador, junto con el estado de las filas en ese turno, separados por '_'. Este proceso permite registrar quién ganó cada ronda en cada partida.

Además, se generaron datasets específicos para cada modelo que utiliza Q-learning. Estos datasets contienen las ponderaciones obtenidas por el modelo durante el entrenamiento. Cada premio obtenido en cada ronda se suma al acumulado de rondas anteriores en cada partida, proporcionando así las ponderaciones actualizadas de cada carta para cada modelo correspondiente.

Al utilizar estos datasets durante las partidas contra los modelos entrenados, cada modelo utiliza el dataset específico entrenado con las características y ponderaciones correspondientes. Esto permite intercambiar datasets en cualquier momento para cambiar el entrenamiento realizado por otro con características similares o mejoradas.

9 RESULTADOS

Como resultado del proyecto, se han evaluado varios aspectos clave:

9.1. Modelos de IA

El proyecto incluye varios modelos de Agentes Inteligentes diseñados para jugar al juego Toma6!. Estos modelos fueron evaluados mediante 100 partidas jugadas contra cada uno de ellos. A continuación se presentan los resultados obtenidos:

Como se puede observar en la figura 5, el modelo que tiene más capacidad para ganar partidas es el modelo Lanzado, cuya estrategia consiste en jugar la carta más cercana a las existentes en la mesa. El siguiente en rendimiento es QLearningCount, entrenado con 50,000 partidas, el cual utiliza recompensas para ponderar las acciones basadas en las cartas en la mano, las cartas en la mesa y la cantidad de cartas en cada fila. Después de este modelo

En cuanto al rendimiento, el algoritmo Monte Carlo también muestra buenos resultados, alcanzando el segundo mejor *winrate* 37 % contra el modelo Lanzado.

Claro, aquí tienes el texto corregido y formalizado:

El modelo Quimera, que combina características de los modelos QLearningCount y Monte Carlo, se aproxima al

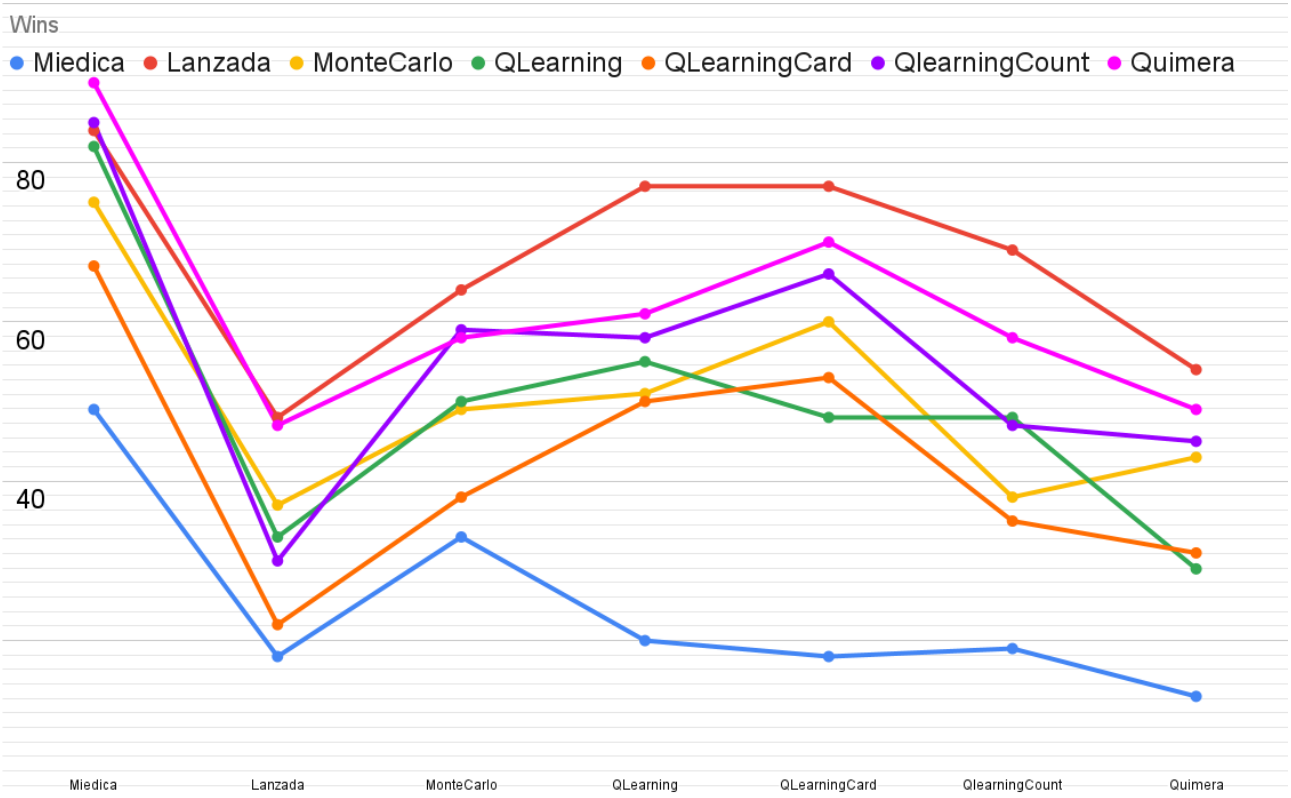

Fig. 5: Gráfica de Partidas Ganadas por cada Modelo

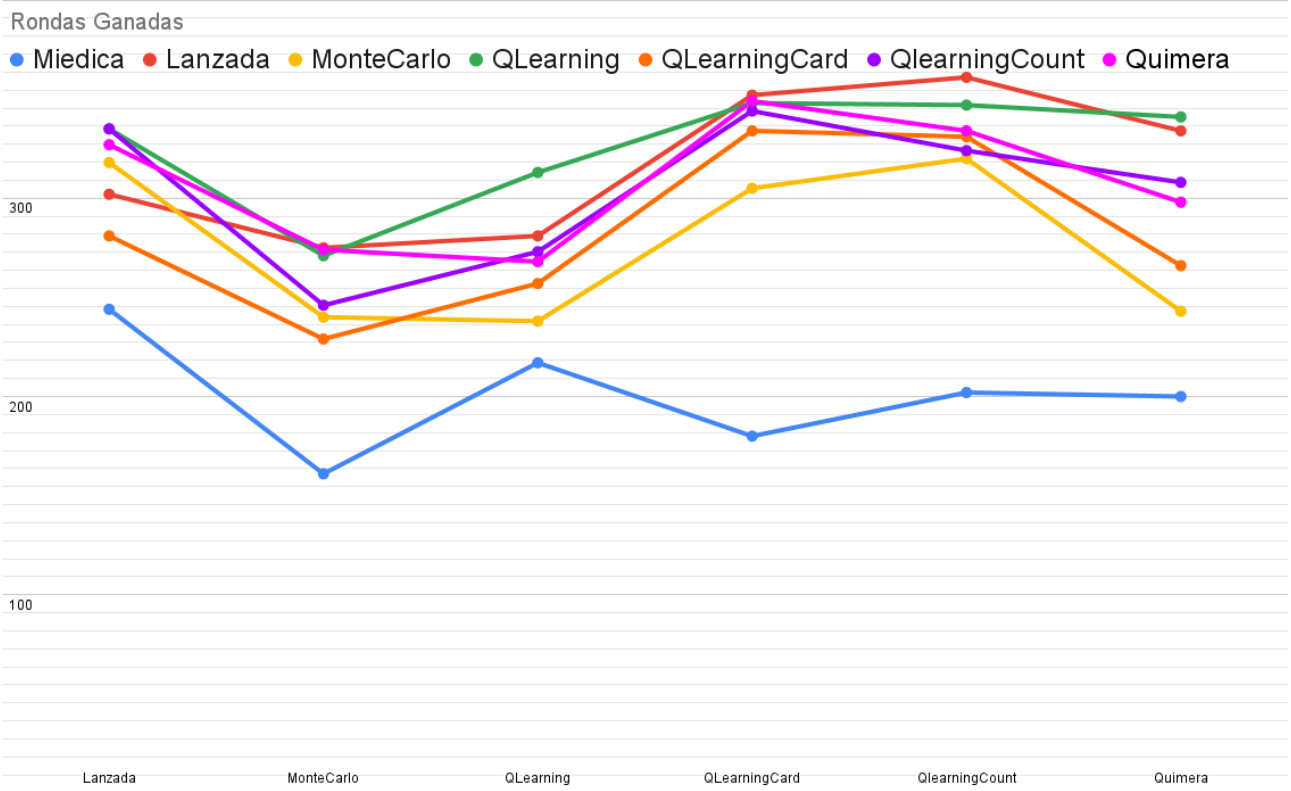

Fig. 6: Gráfica de Rondas Ganadas por cada Modelo

rendimiento del modelo Lanzado y destaca por su buen *winrate* contra cualquier otro modelo. Sobresale especialmente al ser el que finalmente consiguió un *winrate* del 47 % contra el modelo Lanzado.

En cuanto a las rondas ganadas por cada modelo (figura 6), se observa una distribución similar a la de las partidas ganadas, aunque con menos diferencia entre ellos. Esto sugiere que los modelos que ganan más partidas tienden a ganar menos rondas, posiblemente debido a decisiones que llevan al otro jugador a obtener más puntos y terminar la partida antes.

Además, se realizó un estudio adicional enfrentando a los tres mejores modelos entre sí:

Los datos obtenidos en esta gráfica representan el resul-

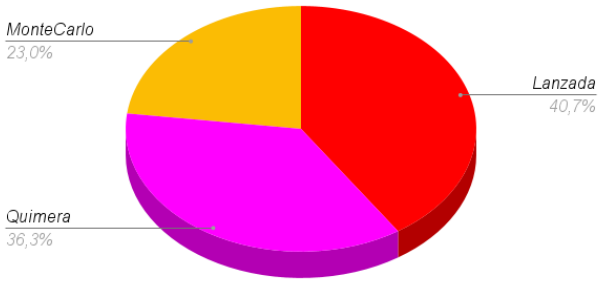

Fig. 7: Gráfica de enfrentamiento entre los tres mejores modelos

tado de enfrentar los modelos Lanzado, Monte Carlo y Quimera en múltiples sesiones de 100 partidas. Se puede observar que el modelo Lanzado tiene un *winrate* del 40 %, seguido por Quimera con un 36 % y finalmente Monte Carlo con un 23 %.

Estos resultados sugieren que la estrategia de jugar la carta más cercana a las demás, utilizada por el modelo Lanzado, es efectiva y robusta en diferentes escenarios de juego. La Quimera, gracias a su combinación de aprendizaje y probabilidades, también demuestra ser una estrategia competitiva, aunque ligeramente menos efectiva que el Lanzado en este estudio. Monte Carlo, por otro lado, muestra un rendimiento inferior, posiblemente debido a que su estrategia se ve contrarrestada por la Quimera.

En resumen, jugar la carta más cercana a las demás parece ser una de las mejores estrategias en el juego Toma6!, ya que proporciona una jugada relativamente segura y reduce las posibilidades de llevarse la fila completa, permitiendo al jugador evitar acumular puntos innecesarios.

10 Conclusiones

Para realizar una explicación de las conclusiones obtenidas en este trabajo, se han dividido en tres secciones. En primer lugar, se presentarán las conclusiones generales, donde se discutirán los resultados obtenidos y las implicaciones derivadas de estos hallazgos. En segundo lugar, se abordará el aprendizaje adquirido por el alumno a lo largo del desarrollo de este proyecto. Por último, se explorarán las posibles líneas de investigación futuras que podrían seguirse en caso de continuar con el proyecto.

10.1. Conclusiones generales del trabajo

A través de los resultados obtenidos en este proyecto, se han extraído varias conclusiones significativas. En primer lugar, se logró desarrollar completamente el proyecto, incluyendo una interfaz gráfica funcional y la lógica del juego para permitir la participación de un número variable de jugadores. Además, se crearon múltiples modelos de inteligencia artificial capaces de desafiar a los jugadores, cada uno con comportamientos distintos.

Centrándonos en los resultados de los modelos desarrollados, comenzamos con el agente “Miedica”. Este agente demostró ser eficaz con un comportamiento pasivo y cauteloso, lo que corresponde al perfil de jugador pasivo buscado.

Por otro lado, el agente “Lanzado”sobresalió al mostrarse audaz y siempre dispuesto a arriesgarse por la victoria. Sorprendentemente, este agente obtuvo uno de los mejores índices de victoria entre todos los modelos desarrollados. En conclusión, el modelo de juego adoptado por el agente ”Lanzado”se establece como un enfoque efectivo para lograr victorias rápidas y refleja el perfil de un jugador dispuesto a asumir riesgos en juegos de cartas.

Posteriormente, se implementó el modelo “MonteCarlo”, el primero en incorporar una lógica detrás de sus decisiones. Este modelo se destacó por su enfoque calculador, evaluando las cartas y optando por las jugadas con mayores probabilidades de éxito. Sin embargo, a menudo adoptaba decisiones más conservadoras para evitar perder puntos en la ronda actual, lo que podría perjudicarlo en rondas futuras. A pesar de esto, el modelo cumplió con el perfil de jugador calculador, aunque más inclinado hacia una estrategia pasiva.

El siguiente modelo, el “Aleatorio”, se diseñó inicialmente para generar datos para modelos entrenados, aunque podría simular el comportamiento de un jugador novato que explora estrategias aleatorias para aprender el juego.

Los modelos basados en Qlearning aprendieron a partir de trazas de recompensas derivadas de jugadas aleatorias durante el entrenamiento. El modelo ”Base”fue el más básico de estos, centrando sus decisiones en la posición media de las cartas en el momento del juego, aunque no logró resultados sobresalientes, sirvió como punto de partida para desarrollos posteriores.

El modelo “Qlearning Cartas”mejoró al considerar las cuatro cartas del tablero y elegir la que ofreciera la mejor probabilidad de éxito. Finalmente, evolucionó al modelo “Qlearning CartasCount”, que no solo consideraba las cartas individuales, sino también el número total de cartas en cada fila, ajustando su estrategia según la seguridad de la jugada.

Estos modelos representaron un avance significativo en el desarrollo del proyecto, demostrando buenos resultados que beneficiaron a jugadores más experimentados al recordar movimientos previos que resultaron exitosos.

Por último, se diseñó el modelo “Quimeraçon el objetivo de competir eficazmente contra el modelo ”Lanzado”. Este modelo combinó elementos del avanzado “Qlearning CartasCountçon estrategias del “MonteCarlo”, resultando en un agente que equilibraba la cautela con la experiencia previa en partidas. Esto llevó al modelo ”Quimera.ª casi empatar con el rendimiento del ”Lanzado”, convirtiéndolo en uno de los modelos más exitosos desarrollados.

En conclusión, este proyecto sobre el juego de Toma6! ha proporcionado varios modelos con diferentes patrones de comportamiento, demostrando que el juego favorece a aquellos dispuestos a asumir riesgos. Cada modelo desarrollado logró los resultados esperados según su diseño y estrategia específica.

10.2. Aprendizaje del alumno y experiencia personal

Como aprendizaje en este proyecto, he adquirido varias habilidades significativas. La más destacada ha sido la ges-

tión completa de un proyecto desde cero, incluyendo la planificación de horarios y el manejo del tiempo dedicado semanalmente a diferentes tareas.

Además, he aprendido a desarrollar interfaces gráficas en consola utilizando Visual Studio, prescindiendo de programas externos o librerías adicionales.

Uno de los aprendizajes más importantes ha sido la creación de agentes inteligentes. Este proceso me obligó a familiarizarme con las diferencias entre modelos utilizados en juegos de cartas y a considerar los parámetros específicos necesarios para el juego de Toma6!. Esta experiencia me ha enseñado a valorar qué modelos son más adecuados según el contexto del problema. Durante este proceso, también he profundizado en el funcionamiento de diversos modelos con los que trabajé y exploré otros que podrían aplicarse en proyectos futuros.

En cuanto a las asignaturas que más me han ayudado a desarrollar este proyecto, destacaría las siguientes:

- Inteligencia Artificial: Esta asignatura fue fundamental, ya que introdujo los algoritmos y la lógica aplicada a casos reales, proporcionando la base para entender los modelos de agentes inteligentes.
- Análisis y Diseño de Algoritmos: Aquí perfeccioné mi comprensión de los algoritmos aplicables en aplicaciones y modelos, además de aprender técnicas para simplificar y optimizar el código.
- Laboratorio de Programación: Esta asignatura fue esencial al brindarme la experiencia práctica de trabajar en proyectos desde cero y desarrollar interfaces, habilidades que fueron directamente aplicables en este proyecto.
- Aprendizaje Computacional: Aunque no utilicé completamente el aprendizaje profundo (deep learning), esta asignatura me introdujo a conceptos que podrían ser relevantes a largo plazo. Sin embargo, terminé entrenando modelos con Q-learning manualmente.

En resumen, este proyecto sobre el juego Toma6! me ha permitido desarrollar diversas competencias, desde la gestión de proyectos hasta la implementación práctica de técnicas de inteligencia artificial, reflejando un aprendizaje integral y significativo en mi formación académica.

10.3. Trabajo futuro

Como objetivos a futuro para este proyecto reforzaría los siguientes puntos:

- Diversificación de perfiles de jugadores en agentes inteligentes: Ampliar la variedad de modelos de IA mediante el uso de deep learning, utilizando el dataset generado en este proyecto como punto de partida. Se explorará la implementación de TensorFlow para entrenar modelos de IA con un alto *winrate* en el juego de Toma6!.
- Mejora y posible migración de la interfaz gráfica: Considerar mejoras significativas en la interfaz gráfica actual del juego. Se evaluará la posibilidad de migrar la interfaz a Unity[16], una idea inicial del proyecto que no se implementó completamente debido a la priorización de los agentes inteligentes.
- Automatización de la generación de estadísticas: Implementar un sistema automatizado para recopilar datos estadísticos de las partidas jugadas por los agentes inteligentes. Este sistema eliminará la necesidad de ejecutar cada juego manualmente y permitirá la exportación directa de los resultados a formatos como Excel, facilitando el análisis y la visualización de datos de manera eficiente.

AGRADECIMIENTOS

Como agradecimientos quiero agradecer a Jorge Bernal, por ser tan buen tutor y ayudarme en todo lo que ha estado en sus manos y también guiarme en el procedimiento para realizar todo el proyecto y memoria, espero que todo le vaya muy bien y que le agradezco mucho la ayuda que me ha brindado.

REFERENCIAS

[1] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C. and Józefowicz, R., 2019. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

[2] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J. and Quan, J., 2017. Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782.

[3] Toma6!: `https://es.wikipedia.org/wiki/6_nimmt!`

[4] Niklaus, J., Alberti, M., Pondenkandath, V., Ingold, R., Liwicki, M. (2019, June). Survey of artificial intelligence for card games and its application to the Swiss game Jass. In 2019 6th Swiss Conference on Data Science (SDS) (pp. 25-30). IEEE.

[5] Levine, R.A. and Casella, G., 2001. Implementations of the Monte Carlo EM algorithm. Journal of Computational and Graphical Statistics, 10(3), pp.422-439.

[6] TreeSearch: Kaindl, H., 1990. Tree searching algorithms. Computers, Chess, and Cognition, pp.133-158.

[7] Wiering, M.A. and Van Otterlo, M., 2012. Reinforcement learning. Adaptation, learning, and optimization, 12(3), p.729.

[8] Teaching a Neural Network to Play Cards: `https://tinyurl.com/krxxjsdp`

[9] Best techniques for an AI of a card game: `https://tinyurl.com/43tu5uuk`

[10] García-Sánchez, P., Tonda, A., Mora, A. M., Squillero, G., Merelo, J. J. (2018). Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone. Knowledge-Based Systems, 153, 133-146.

[11] https://es.wikipedia.org/wiki/Hearthstone

[12] DAYAN, Peter; WATKINS, C. J. C. H. Q-learning. Machine learning, 1992, vol. 8, no 3, p. 279-292.

[13] Ahmad, M.O., Markkula, J. and Oivo, M., 2013, September. Kanban in software development: A systematic literature review. In 2013 39th Euromicro conference on software engineering and advanced applications (pp. 9-16). IEEE.

[14] Felicia, P. (2017). Unity From Zero to Proficiency (Foundations): A step-by-step guide to creating your first game with Unity (Vol. 1). Patrick Felicia.

[15] Visual Studio: `https://visualstudio.microsoft.com`

[16] Unity: `https://unity.com/`