MTG Draft Bot

Nick Richter

Abstract

Magic: The Gathering (MTG) is a popular trading card game involving thousands of cards. There are many different ways to play MTG, with many different sets of rules. One particular ruleset is called Draft; think of a fantasy sports draft. A group of players gets together and opens packs of Magic cards, and passes the packs around, picking cards from each pack to make their decks. The goal of this project was to create an intelligent agent that, given a set of cards, could draft a deck based on a set of conditions. This was achieved by creating a value function that generated a unique value for each card, and then compared the values of all available cards at each step of the draft. This goal was reached, to a degree, although, with more work, the value function can be improved to allow the agent to draft better collections of cards.

Introduction

MTG is a card game in which each player creates a deck, typically of either 60 or 40 cards, depending on the ruleset being used. These cards can be any cards in their collection. The goal is to make the best deck and beat other players in one versus one games. There are thousands of unique MTG cards, and these cards are broken into installments called sets, typically with around 250 cards each. New sets are released every three months. A MTG Draft involves only one set of cards, limiting the possible cards in play to around 250. A Draft works as follows: a group of players (usually 8) each opens a pack of the desired set. From this pack, they choose one card to add to their collection. They then pass the rest of the cards in this set to the next player in order. That player then selects one card, and pases the remaining to the next player. This continues until all the cards are picked, and each player has 15 cards. Then, every player opens another pack, and the same picking process occurs. Finally, a third pack, with the same picking process is opened. By the end of the draft, each player has 45 cards to make a deck with. The players then use those decks to play each other. Obviously, succeeding in a Magic Draft is heavily influenced by how good one is at drafting. While it is extremely difficult to create an intelligent agent that can effectively play the game of Magic, it is possible to create an agent that can (with relative success) draft against other humans. This project attempts to create an intelligent agent that is able to draft cards given a set of possible picks and a collection of previously picked cards.

Background

This project created an agent that can draft the set Theros Beyond Death (THB), the second most recent set to be released. The agent uses an online Magic Drafting simulator from cardsphere to simulate drafts. Because Magic cards do not have standardized power levels, it is impossible to know exactly which card is best overall. Additionally, some cards are better in certain scenarios, even if they are worse overall. So, creating a drafting agent is not as simple as telling it the best and worst cards, because there is no default card list; it is often objective. While no card decisions can be universal, nearly always a decision can be made about which card is better to draft. To give a base ‘ranking’ for the value function, this project used data from draftsim, a site that creates in depth guides to MTG draft strategies. Draftsim ranked many of the cards in pick order, i.e. 1 is best and 254 is worst. It also ranked them based on archetypes. An archetype is a certain type of MTG deck. To understand this concept, it is necessary to understand card colors. There are five colors in MTG, and each color requires a certain type of mana to play it. Each deck contains a certain amount of mana; usually about 40% of the deck is mana. So, if you are playing a deck with two colors, you will need both kinds of mana in your deck. THB has 10 archetypes, each a combination of two colors. This is important for drafting because, if you are deciding what card to pick, you need to choose a card you can play in your archetype. For example, if your deck is red and blue, you should not pick a black card because you won’t be able to play it in your deck. So, the draft agent is able to tell which archetype it is drafting based on the cards it has already picked. The base values for each card for each archetype is pulled from the draftsim card archetype rankings. The data pulled from draftsim is stored in a single json file, thbCardList.json. This json file has detailed data about each card in THB, such as the card name and its archetype rating. This file was created as part of the project, drawing much of the data from mtgjson, an open source project that creates json files for MTG cards. To understand the project, the concept of archetype values must be understood.

Related Work

Several others have made intelligent agents that can rationally draft MTG. This project is not based off of them, although it did consult them out of interest. These projects used large datasets of existing human drafts, and utilized machine learning techniques to create an agent that would draft in a similar manner. That was not possible for this project, as the data is private and was obtained from the source directly, instead of being publicly available. This would be an excellent way to build on this project, and to compare an agent using a developed value function and an agent that learned from actual human drafts.

Project Description

The bulk of this project involved obtaining data, reliably connecting to a simulated draft, and automating an agent to draft on its own. This project was created using Python, and all scripts are available in the github repo. This will be broken down step-by-step, below:

1. Generate thbCardList.json
   Before any draft attempts could be made, basic data from all cards in the set was needed. This began by pulling data from mtgjson.com, where a file for THB is publicly available. This file included valuable data about each card, but did not have any value system. The json file was edited and combined with pick data from draftsim.com/theros-beyond-death-early-analysis. thbCardList.json was generated using the script thbCardListBuilder.py, which pulled necessary data from draftsim and injected

it into the existing json file from mtgJson. Additionally, draftsim did not include archetype ratings for rare cards, so a separate script, rareArchetypeAssignment.py, pulled overall pick rankings from draftsim and used it to create archetype ratings.

2. Connect to draft.cardsphere.com and obtain draft simulation data
   Once basic rating data was obtained for all cards in the set, the agent had to have access to data of an actual draft. Cardsphere has an online draft simulator that provides data for a draft via HTTP requests. The script draftBot.py, which does a bulk of the work in drafting, first utilizes the python requests library to retrieve information from draft.cardsphere. It then passes this data to environmentBuilder.py, which obtains the needed data from the response object and passes it back to draftBot.py. This data includes the cards that can be picked this round, as well as the cards already picked by the agent (these are necessary to tell which archetype is being drafted).
3. Select a card to draft, based on available cards and previously picked cards
   Once the agent has the draft information, possible picks and previous picks, available, it must select a card to draft. This calculation is done in draftBot.py. It begins by first iterating through the previously picked cards and discovering which archetype is most present. It does this by finding, for each archetype, the average rating based on all previous picks. Once the agent knows which archetype is most predominant in its collection, it will be biased towards cards that fit best in that archetype. However, there is a value, pickThreshold, which offsets this bias. pickThreshold can be input when conducting a trial. The bias is lower earlier in the draft, to reflect being 'open' to switching archetypes. As the draft goes on, the bias becomes larger because switching archetypes becomes a worse idea. So, the agent uses the cards base archetype rating, combined with the pickThreshold, to decide which card to pick.
4. Automating the agent to conduct full drafts
   The agent described above only makes a single decision, based on the state it receives from cardsphere. In order to make the agent conduct a draft, interaction with the site must be automated. This project utilized the selenium library for python to automate the process of drafting (clicking the cards, loading the site up, etc.). This is done in draftAutomator.py. A number of trials can be specified, denoting the number of drafts to undertake.
5. Managing multiple trials in parallel
   Automating a draft takes a while (around 1 minute per draft) as the agent needs to wait for responses and simulate clicks on the page. To speed up testing, this project utilizes the multiprocessing library for python. The file trialManager.py generates a specified number of parallel draft instances, each that will run a specified number of drafts. This allows for multiple tests to run at once.
6. Recording data
   The results and summaries of the draft instances are written into files, which are stored in the logs folder of the project. Each instance creates a file 'draftLog*x*.txt' depending on its instance number. Once a trial session is finished, it records its summaries in trialResults.txt.

Empirical Results

Many tests were run for this project, but the final trial ran 150 drafts for each of ten different pickThreshold values, in an attempt to find an optimal value for pickThreshold. The summary of these results can be viewed in logs/trialResults.txt. Detailed logs for each draft using each value of pickThreshold can be viewed in draftLog*x*.txt. To optimize the agent, an ideal value for pickThreshold should be identified. Unfortunately, this project did not discover an ideal value. The summarized results do not point to a value in particular, and it is a future goal of this project to use more advanced algorithms to pinpoint this value. This agent can successfully, however, make rational decisions given a set of possible picks and a collection of previously picked cards.

Conclusion

An agent with the capabilities of selecting a card from a set of options was successfully created. The agent uses a value function to determine which card is best at any given state. However, an ideal value for pickThreshold was not identified. Moving forward, that is the next step for this project. A learning algorithm could be implemented to pinpoint a value for pickThreshold that maximizes average ratings for draft collections. Additionally, this project could benefit from exposure to draft data from actual human drafters, and could combine its value function with a machine learning agent that produces decisions based on human choices. Also, the value function of this agent could be improved to take more specific card attributes into account, making the agent even more optimal. Personally, I learned to use many Python libraries I had not known (Selenium, multiprocessing) and I grew very comfortable generating/organizing data. To future students, I advise them to choose something they are interested in, because this project was a lot of fun for me. I would also recommend they get started early, as I did much of my work well before finals, and I wouldn't have had a chance to do nearly as much if I didn't start early.

Sources

https://draftsim.com/ryan-saxe-bot-model/
https://draft.cardsphere.com/
https://draftsim.com/theros-beyond-death-early-analysis/
http://mtgjson.com/