Joseph Lopez
Billy Davies
Conall Barrett
Group 46

Suggestion Algorithm for Auto-Completing a Magic: the Gathering Deck

Games are one of the most explored and answered areas of machine learning and artificial intelligence. Checkers, Chess, Go, and countless other games have AI players that can beat the most renowned world champions 100 on a consistent basis. However, there is no such bot in *Magic: The Gathering*. *Magic: the Gathering* is a competitive Trading Card Game in which players build their own deck of 60 cards and compete against other players to reduce their life total from 20 to 0. There are a variety of formats in which different cards are allowed to be played or are banned, some examples being Standard, in which cards from the last ~year and a half can be played, and Modern, in which cards from 2003 and later are legal to play in a deck, a total of 13,663 cards (Scryfall) and millions of combinations into decks. Decks have a limit of 4 copies of a card per deck -- with an exception made for land and some other special cases-- and are often built into “archetypes” like Control or Aggro, where they contain similar plans to win the game and similar themes among cards included. According to an independent research team, MtG is “the most computationally complex real-world game known in the literature”. MtG is also to have been proven to be turing complete this past year.

Figure 1. An Example deck in the Modern Format from MTGTop8.

After looking through the techniques listed in the provided project ideas section, we found that those which were utilized in the domains of recommendation engines and constraint satisfaction problems were the best suited for our problem. We chose to utilize a supervised learning model in the light of methods like alternating least squares regression and probabilistic PCA to recommend cards in order to complete a given incomplete decklist and constraint satisfaction to guarantee the deck was legal.

Although our project may seem to deviate from the guidelines, the professor gave an okay on the project on a recent zoom chat.

With our pioneering system, a user could select a particular meta relevant archetype or set of colors, and decklists which meet said criteria are utilized to impute a user's incomplete decklist and return a legal and competitively viable deck. We scraped decklists off of MTGtop8, a website that collects results and decks from online tournaments on one of Magic's online clients Magic: the Gathering Online (MTGO). MTGtop8 also classifies this data into archetypes, so we were able to manipulate our data easily using the provided "Archetype" field in the resulting data structure to make our algorithm work for each archetype. We used another site, MTGJson, to get text, card types, and other important data about all of the modern legal cards and place them into a dataframe.

When exploring the kind of data would be needed for this project, two things immediately stuck out. The first thing we reasoned, is that any project involving *Magic: the Gathering* would require some data about the number, the type, and the attributes of different cards. At first this seemed like it would be an easy thing to find. *Magic: the Gathering* has over 35 million players spread out across the globe. Additionally there is a large demographic overlap between data scientists, the people who would generate and use the type of data set that we need, and *Magic: the Gathering* players. However, despite our exhaustive searches, there were no readily available sets of card data. The closest thing that we found was MTGJson. This had a UI for displaying individual cards and attributes specific to each card. And thus, our first challenge presented itself. We ended up solving this challenge by creating a web-scraping bot using beautiful soup, selenium, and the requests library. This bot went to each page and gathered all the necessary information before moving onto the next page and repeating. This script ended up running for hours each time it was run.

The second realization we had during the data gathering phase is that we would need a heuristic for quantifying how similar cards were in order to generate these deck lists. One of the first ideas we had on how to generate this heuristic was to compare different attributes of the cards, and to output a float value of how similar these cards were. This heuristic would not require any additional data sets beyond the card data set that we had already gathered. However, this raised questions on how we would create legal decks with the correct ratios of card types and a good blend on mana costs. The second idea we had for a heuristic comparing the similarity of different cards was based on the relative frequency of appearance in the same deck. In order to implement this heuristic, we would need decklists, and we would need a lot of them. Fortunately, *Magic: the Gathering* provides a series of official decklists. Unfortunately, those decks are universally regarded as so bad they are unplayable within the *Magic: the Gathering* community. If we used those decklists as data for new decklist generation, our decks would be similarly unplayable. We were able to find an alternative source for decklists from MTGGoldfish, a popular website for all things *Magic: the Gathering*. Since each decklist was on a different webpage, we created a similar script as the one for scraping cards, also using beautiful soup, selenium, and the requests library.

Regular decklists carefully balance cards they include to ensure they balance both ways they plan to win the game and disrupt their opponents plan. When running

through and executing our project, we had several instances in which decks that were simulated lacked either win conditions or the cards needed to disrupt. We utilized card selection and satisfaction of the decklist constraint for imputation. Cards to be selected to be put into the decklist upon each round of imputation were the highest average correlated cards over the set of currently included cards. We realized our imputations were liberal and as such we had to utilize conservative heuristics to select cards for the deck. This likely due to picking too many cards using our simulation algorithm at a time, and cards that were normally run with less copies -- which are generally reliant on other cards (Example 1 copy of Batterskull requires 4 Stoneforge Mystic) -- would get thrown in due to the deck needing one or two copies to be slotted into the deck. As such, we lowered the number of cards added in each iteration, so less random 1 copy cards were included and the decks were more accurate at the cost of a longer runtime. With archetypes that require less combinations of 1-of combo cards, the model runs well and gives better approximations, although some archetypes have more variance and result in longer runtimes of a few minutes.

Many of the different evaluation techniques of probabilistic graphical models are easy to deduce directly from the problem that is being solved. For example, a movie recommender system can be evaluated and fine tuned based on how often users select the movies that are being recommended. A bayesian classification system can be evaluated based on the percentage of classifications that are correct. However, with our *Magic: the Gathering* deck generator there is no one easy way to evaluate how successful our generations are. As the saying goes, one *Magic: the Gathering* player's trash, is another's treasure. One player might view a deck as a competition caliber deck, while another player might view the same exact deck as unplayable. With the absence of any robust artificial intelligence simulators to test out win rate between decks, there is no good way to objectively evaluate a deck. Although players can not agree on whether a deck is good, they can usually agree on if a deck is bad.

We chose to evaluate our decks on several bases, due to the complexity of building and playing Magic: the Gathering decks. Our evaluation compared a deck scraped off the MTGtop8 website, and a deck created using our algorithm with a random number of cards (around 75%) cut. While we do measure a card-to-card analysis between the original and new decks,, we also compare other variables to see if cards

Figure 2. A side-by-side comparison of Ugin and Karn. They fulfill the same roles in deckbuilding

that are replaced are direct replacements that offer similar effects or costs to our original, as that could confound the deck analyzer. To do this we also used average Converted Mana Cost (CMC), which is the cost (i.e. how many resource cards (lands) are required to play the card), of cards in the deck and compared them, as decks that are more aggressive want lower cost to play cards, where as Eldrazi decks want an average mana cost much higher than that of most decks due to their ability to generate the most resources of all decks. Decks also want the same count for each card type between the imported list and the replacement list, so we count the number of instants, sorceries, lands, creatures, and other types separately and compare them. This gives a more accurate depiction of the deck building algorithm and compares true similarity rather than similarity on a card-to-card basis. For instance, Eldrazi has a wide variety of win conditions, so replacing a card like Karn Liberated with a copy of Ugin, the Spirit Dragon is functionally equivalent, as they both win the player the game, they are of the same type, and similar mana costs (7 and 8 respectively, FIgure 2) . We also run our Mean Squared Error test on a general consensus of the format by averaging the CMC, type distribution, and cards shared from all the decks in the archetype to attain an average similarity score based on how the most ideal or average deck in the format would play.

In Figure 3 we display the deck generation process on a test list. We selected for only a quarter of cards to be kept within the list and compared our imputed list to the original in terms of card type ratios as well as converted mana cost for each card type. We computed the mean squared error for the average ratios and converted mana costs of card types within a deck between the test and the imputed deck as well as between the test and the archetypes average converted mana cost and ratios. Our imputed deck got a rmse of around 5.61 while our test deck got an rmse of 12.98 when compared to the archetype's average. This finding indicates that there is so much in-archetype variance among decklists that it is advisable to utilize a tool like ours in order to generate a competitive decklist. We believe that deck imputation based on meta relevancy is the future of deck building across all trading card games.

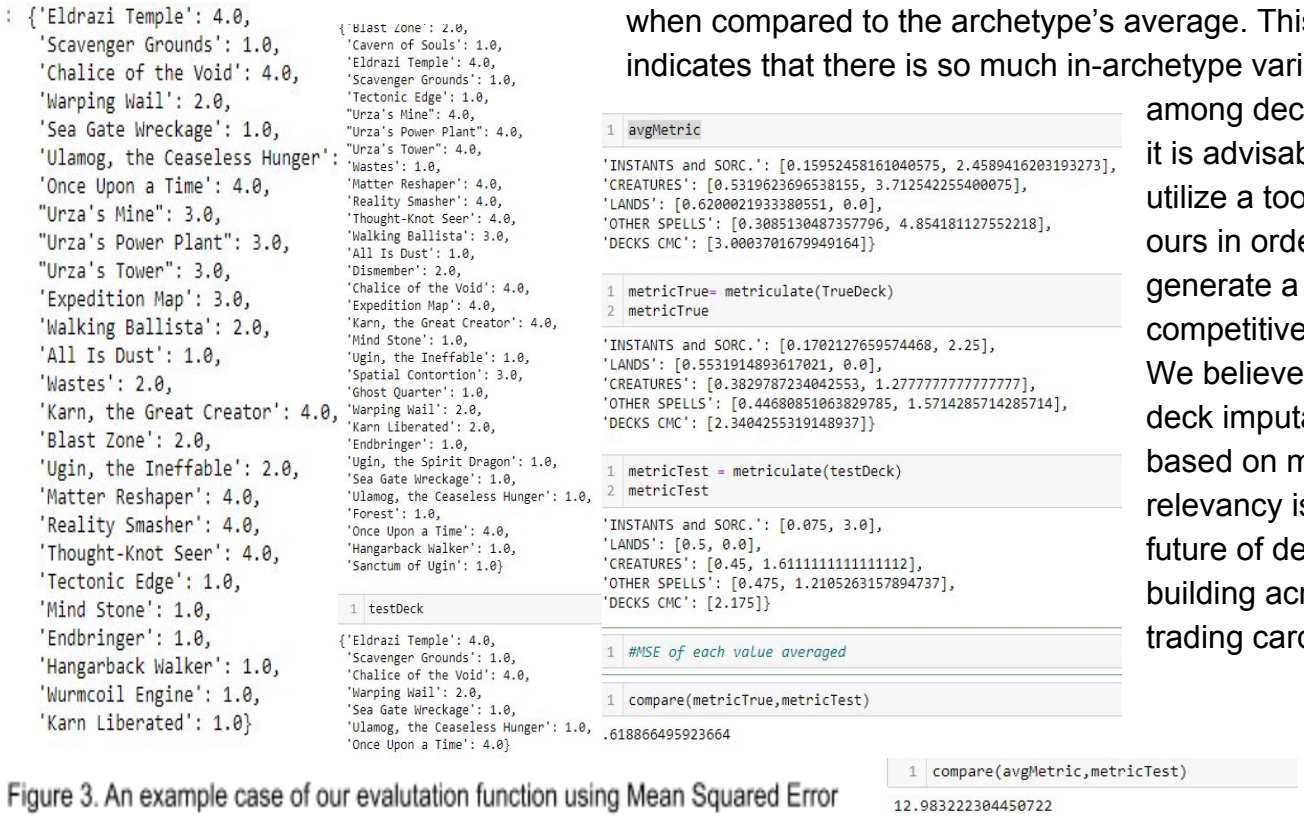

Figure 3. An example case of our evalutation function using Mean Squared Error