Gwen Dekker
Group 7

Magic: The Gathering Cube Multi-Purpose Model

Basic Idea

Magic: The Gathering (MTG) is a popular trading card game produced by Wizards of the Coast. The game contains over twenty thousand mechanically unique cards, and is enjoyed by millions of players around the world. MTG Cube is a way to play the game that involves a single person putting together a set of cards, and using that set to play a series of games with a group of players. MTG Cube is closer to a board game than a trading card game; the cube owner taking the role of game designer. During the design of a cube, the designer may take significant effort to choose which game pieces of the pool of over twenty thousand to include in their cube. They may want to test out their current iteration using sample playtests against simulated players, to help tune their cube and make it ready for human players. The MTG Cube Multi-Purpose Model seeks to be a tool for a cube designer to design their cube more effectively.

MTG Cube is played by constructing a series of packs, typically three packs of fifteen cards each for eight players. The players then draft the cards one pack at a time, picking one card, then passing the rest to the next player, and picking from the pack they've received. After all cards are drafted, each player builds a deck to play with against the other players using their drafted pool of forty-five cards plus some from a "free" pool.

There are three different goals this model addresses:

1. Recommend: What card(s) should I put in my cube? What card(s) should I remove?
2. Draft: What card should I take from this pack?
3. Build: Which cards from my pool should I play?

Data Source

Cube Cobra is an open source web application built by a community of players heavily enfranchised in MTG cube. Cube Cobra has approximately 100,000 cubes, and 1.4 million playtest drafts. This data is made public by the maintainers of the site, so that members of the community can build tools for everyone to use, such as the Lucky Paper Cube Map†. We can use this data to train our model.

Approach to Solution

The model will essentially be a denoising autoencoder, with three separate decoders, all sharing the same latent space encoding. We can represent the input of the model as a list of cards, a vector of dimensionality N, where N is the number of unique cards. Since each index corresponds to a card, a cube can be represented as this vector with 1's representing the presence of a card, and 0's representing the absence. Cubes are typically singleton, with no card repeated, but not all cubes use this stipulation. This is a limitation of our model – we will be blind to multiple copies of cards in a list. For building, a pool can also be represented in the same manner.

The Recommend decoder seeks to answer the questions, (1) what card(s) are missing from this cube? and (2) which card(s) should not be in this list? The output is the same shape as the input, but each dimension represents the probability that a user would add that card to the list. In dimensions where the cube doesn't contain that card, we can sort them and then provide a list of cards to the user

† **Lucky Paper Cube Map: https://luckypaper.co/resources/cube-map/changelog/**

of cards that maybe should be in the cube. The reverse can provide a list of cards that maybe shouldn't be in the cube.

I can perform data augmentation by adding random cards, and removing random cards, then training against the target of the original cube. To avoid the result of having the recommender always recommend adding the most popular card across all cubes, I can weight the adding of cards based on how generically popular those cards are. This added noise will serve as a form of regularization.

The Draft decoder seeks to decide which card to take from each pack during the draft. The input will represent the pool of cards already picked, and the output will be the same size vector where each dimension represents the probability a user would pick that card. If we mask this vector by the cards available in the pack, we can get a ranking of each card – and ultimately the best card to take.

The Build decoder seeks to choose what cards from a completed pool to use to build a deck. The output will be the same vector, where each dimension represents the probability that a user would play that card in their deck. We would take this output and apply a bit of old-fashioned logic to determine what free cards (from the set of unlimited cards) to add to build a forty-card deck.

Having all of these decoders operate on the same encoding will effectively provide regularization pressure. We don't want the encoder to learn some function specific to one of these tasks, we want to make sure to encode information of all related tasks to achieve a more meaningful representation of a set of cards.

The implementation will be using Tensorflow in Python. Some data wrangling will be required, which I will use JavaScript for, as the source data is all JSON.

Assessment Methodology

I will construct my model in such a way that I can give the input annotated with which of the three functions to use, and have it provide the error of the combined autoencoder. I will use binary cross entropy for my loss function regardless of which function is being used. I will be evaluated the model on a sample of the data that was not part of the training data.

Data Ablation

The format of the source data needs a bit of data processing to get it into the vectors I've described. For cubes, each cube is a list of card IDs. I'll first need to create a mapping of card IDs to the respective index. I'll need to then use this mapping to transform the list of IDs to the correct vector shape. For the Draft input, the data needs even more processing. The source data contains an initial state of the contents of each pack before they're opened, as well as a pick order for each seat. I'll need to reverse engineer the cards present in each pack as the player seat is making each choice, and then transform each deck into a list of pack, pool, pick triplets. Decks in the source dataset are formatted the same as a cube, and will need the same transformation.

Related Work

Pan, Y., He, F. & Yu, H. (202) A correlative denoising autoencoder to model social influence for top-*N* recommender system. *Front. Comput. Sci.* **14**, 143301 Available from https://doi.org/10.1007/s11704-019-8123-3

† **Lucky Paper Cube Map: https://luckypaper.co/resources/cube-map/changelog/**

Ning, X. & Karypis, G.. (2010). Multi-task Learning for Recommender System. *Proceedings of 2nd Asian Conference on Machine Learning*, in P*roceedings of Machine Learning Research* 13:269-284 Available from https://proceedings.mlr.press/v13/ning10a.html.

† **Lucky Paper Cube Map: https://luckypaper.co/resources/cube-map/changelog/**