Text Generation Using Recurrent Neural Networks for Magic: The Gathering Cards

Brandon Lynch

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Bachelor of Science

Computer Science and Systems with Honors

University of Washington Tacoma

2022

Advisors:

Dr. Raghavi Sakpal

Dr. Chris Marriott

Program Authorized to Offer Degree:
School of Engineering and Technology

University of Washington Tacoma

Abstract

Text Generation Using Recurrent
Neural Networks for
Magic: The Gathering Cards

Brandon Lynch

Advisors:
Dr. Raghavi Sakpal

Dr. Chris Marriott

Creating new expansion sets is an important aspect of successful trading and collectible card games such as Pokémon, Yu-Gi-Oh, and Digimon, amongst others. Magic: The Gathering (MTG) is one of the greatest trading and collectible card games in the market and is constantly releasing new cards. Though designing new game levels in the field of AI and games is relatively common, designing new cards is still in its infancy. As part of our research, we have used eecurrent neural networks (RNN) to generate new and novel Magic cards that are linguistically correct, within the rules of the game, and provide gameplay value. In this thesis, we first describe the design of two different RNN models: 1) Character-based RNN and 2) Word-based RNN for card generation. Then, we cover the results generated from both these models which are evaluated based on their accuracy, contextual similarity to the existing dataset of Magic cards, adherence to the game rules, and usefulness within the game of MTG.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1
INTRODUCTION

Magic: The Gathering (MTG) is one of the most popular Collectible Card Games (CCG) in the world [11][27]. With new cards being released at least once a quarter [11], there is expectation for the developers of the game to design new, unique, and exciting cards. The effects of cards in-game are determined by rules text and other information printed on the cards, so with an understanding of the foundational rules, new cards can be introduced and players can quickly understand what they do [28].

The existence of foundational rules and natural language in this game lends itself well to machine learning. It is well-established that recurrent neural networks (RNNs) are effective at learning text sequences [17], so by turning the information on MTG cards into text sequences, an RNN could be trained to learn these sequences and used to generate new cards. Milevicz explored this area using a simple character-based RNN, and showed that unique and functional cards can be generated when training the model on all existing MTG cards [19]. Reinforcement learning (RL) has also been used in the field of imperfect card games, as seen in the work done by Vieira et al [13]. However, this kind of work usually focuses on solving games (i.e. gameplay), whereas the purpose of this project is to focus on designing new MTG cards.

Our research objective was to design and compare two RNN models: 1) A character-based RNN, and 2) A word-based RNN, with the purpose of generating new MTG cards. In this thesis, we describe the designs of various iterations of the character and word-based RNN models, and present the evaluation results. The generated cards were evaluated based on correctness (format, grammar, spelling), adhesion to the game rules, and gameplay value.

Chapter 2
RELATED WORK

2.1 Magic: The Gathering

Magic: The Gathering (MTG) is a Collectible Card Game (CCG) where players each build a deck of cards and use them to deplete their opponent's total life. MTG is the most popular tabletop CCG, with 35 million players in 2018 [27] and more than 20,000 unique cards [11], with more being released every year. Each MTG card contains a text box with rules text describing the effects of the card, so once there is an understanding of the foundational rules of the game, new cards can be introduced and easily used by players [28].

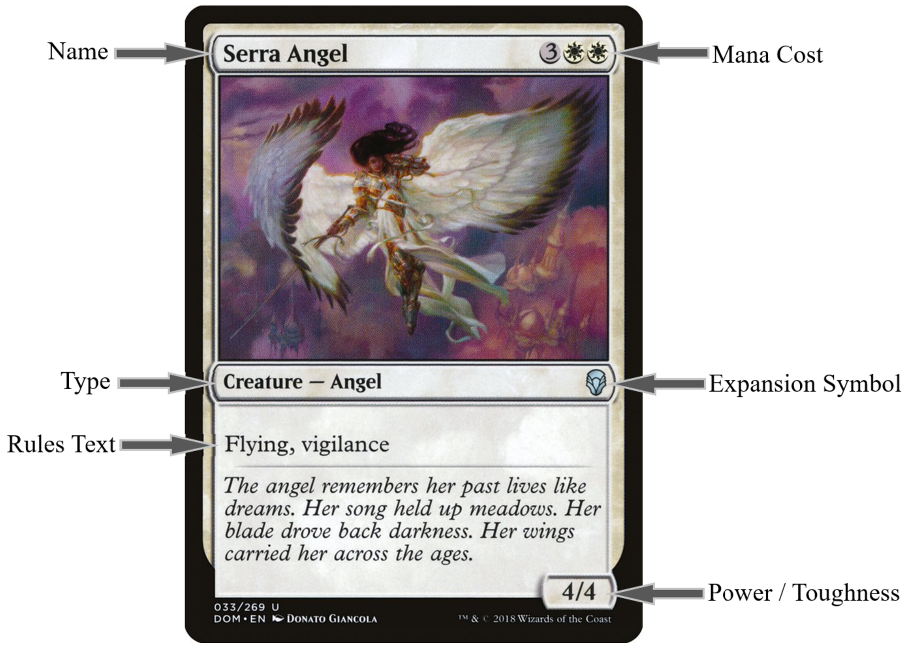
Name
Serra Angel
Mana Cost
Type
Creature — Angel
Expansion Symbol
Rules Text
Flying, vigilance
The angel remembers her past lives like dreams. Her song held up meadows. Her blade drove back darkness. Her wings carried her across the ages.
4/4
Power / Toughness
033/269 U
DOM • EN DONATO GIANCOLA
™ & © 2018 Wizards of the Coast

Figure 2.1: A labelled creature card from MTG

MTG cards have a number of features that are important when generating new cards, as seen in the example in figure above (see Figure 2.1). These features are explained on

MTG's official website [19]. The name of the card is a unique identifier that no other card can share. The mana cost specifies the types and amounts of mana that must be spent in order to play the card. The type includes card types (such as "creature") and subtypes (such as "angel"). The text box contains both rules text and flavor text. Flavor text has no gameplay function, and is only there to provide snippets of story context. Finally, creature cards will include power/toughness values, indicating how much damage they can deal and how much must be dealt to it in a single turn to destroy it. Generally, the larger these numbers, the more powerful the creature. Many of these features have relationships with each other. For example, a card with a higher mana cost will usually have a higher power/toughness. Or, a card with the creature subtype "Angel", such as in Figure 2.1, will almost always have the keyword "Flying" in the rules text and have a white mana symbol in the mana cost [22]. Of these features, the only ones that don't have direct gameplay relevance are the expansion symbol (a symbol showing the card's rarity and the expansion set it was printed in) and the flavor text in the bottom half of the text box. These features were ignored for the purposes of generating new cards.

2.2 Artificial Intelligence & Card Games

Artificial Intelligence (AI) has historically been used to learn games such as Chess or Go, where there's a small number of game pieces that each have simple and well-known rules (e.g. in chess, the King can move one space in any direction) [12]. MTG represents a challenge in the field of game AI, where there is an incredibly large number of unique game pieces (cards). This causes imperfect information card games like MTG to have a very high branching factor (multi-player, large state space), thus making this area of game AI challenging to solve.

Due to the advancement of deep reinforcement learning (RL) algorithms, AI has made significant progress in the field of games. RL has been combined with other methods in order to play games such as Go (DeepMind's AlphaGo Zero) and Atari 2600 games [25][10]. In recent years, researchers have focused on achieving expert-level performance in one of the most challenging games in AI, Poker [3][23]. Poker has remained challenging as it deals with long horizons, multiple agents, large decision space or sparse reward. For example, Texas

Hold'em is played by multiple players with large decision space, where each player needs to play against the other player and reason about the other players' cards that are hidden from their sight [20]. Another example is a popular Chinese poker game Dou Dizhu [13], which suffers from long sequences, sparse reward, and large action space with an explosion of card combinations. In recent years significant progress has been done in solving imperfect information card games as seen in RLCard, a toolkit for RL developed by the Data Lab at Texas A&M [8][9]. Thus, we can say that RL algorithms like the ones in the RLCard toolkit could be successful with card games like MTG, that fit in the category of imperfect information games that involve multiple agents, large state space, large action space and sparse rewards. As seen in work done by Cowling et al [7], Monte-Carlo Tree Search (MCTS) has shown some success for MTG by exploiting the extremely high branching factor and imperfect information of the game state and action space. However, using machine learning to design new cards that are within the constraints of MTG rules is an unexplored research area, and is the focus of this project.

2.3 Designing New Cards

Designing new cards is still a relatively new and less explored area in the field of AI and Machine Learning. The only reference we found on this was the work done by Wilewicz [19], which uses a simple character-based RNN to generate new MTG cards. The Twitter account @RoboRosewater (a reference to MTG's lead designer, Mark Rosewater) would regularly post cards generated by this model [19]. However, cards generated using said model are still prone to language or rules errors, and may not provide gameplay value, even if it doesn't break any game rules. Our objective is to refine both character-based and word-based RNN models and compare their effectiveness, in order to ultimately design a model that can generate new cards with as much consistency and accuracy as possible.

To achieve this we calculate the cosine similarity to find the similarity in the vocabulary between the generated card and the existing database of cards. We use this as a rough score to evaluate the semantic correctness of a card. While this method is generally effective [18][24][21], it may not take into account aspects of the text that are important to evaluating the quality of a generated card, such as grammar. Other, more sophisticated approaches may be more effective.

We found another technique proposed by Lee [16], which uses textual similarity measurements to evaluate and control text generation. One of the specific evaluations proposed by Lee included the use of two embedding spaces-one for the generated texts and one for the existing texts, and to find a contextual similarity between the two embedding spaces. This work was done in the context of patents, which are relatively structured texts, so this technique could be useful to evaluate the accuracy of generated MTG cards, which are also relatively structured. We haven't explored using this technique in our work, but it could be something to explore in future work.

Chapter 3
IMPLEMENTATION

3.1 Dataset

Data was retrieved from mtgjson.com [14]. The dataset used in this project was formed from the AllPrintings.json dataset, which includes all printings of all cards in JSON format [14]. We then refined this to only include the most recent printings of all creature cards, using their oracle text. Sometimes, there are changes to the rules of MTG or to the wordings of specific cards, and the oracle text is defined as the most recent and up-to-date version of a card [20]. So, even if an older card has different text, the rules of the game say to treat the card as if it had the oracle text [20]. Using the most updated version of every card keeps the language on the cards consistent, which is beneficial for the learning of the network.

The relevant data of each card (see Figure 3.1) was extracted and formatted in a way to support both our models as follows [4]:

- Each card was preceded by an arbitrary start flag S|| and succeeded by an arbitrary end flag E||.
- The fields of the card were separated by the arbitrary character: |.
- Since new lines in the card text field have semantic meaning (each ability of a card will be on a new line), newline characters in the data were arbitrarily replaced with the % character.
- Any reference to a card's own name in its text was replaced with the arbitrary character @.
- To avoid confusion between the two meanings of "counter" in the game, any instances of the word involving countering a spell, ability, etc. were replaced with "uncast".

```
S|| Serra Angel | {3}{W}{W} | Creature Angel | Flying, vigilance | 4/4 ||E
```

Figure 3.1: Format of card for character model training data

Additionally, for the word model, each word, punctuation, and symbol was spaced out so the data could be split using space characters as the delimiter (see Figure 3.2). To reduce the size of the vocabulary and consolidate redundant appearances of words, all data was converted to lowercase (except for the mana cost).

```
S|| serra angel | {3} {W} {W} | creature angel | flying , vigilance | 4/4 ||E
```

Figure 3.2: Format of card for word model training data v1

Based on initial results (see Results), we found that the card name was unimportant for the word model, so the card name was removed entirely in later iterations (see Figure 3.3). Additionally, the order of the fields was adjusted to have the card text appear last (see Models).

```
S|| {3} {W} {W} | creature angel | 4/4 | flying , vigilance ||E
```

Figure 3.3: Format of card for word model training data v2

3.2 Models

We decided to use two types of RNN models and compare them: character-based and word-based. The models were built using Python with the Tensorflow [1] and Keras libraries [6]. Cards were compiled into a text file (see Implementation), one card per line. A sequence file was generated, with each line being n+1 characters long. The network was trained using the first n elements as input to predict the n+1 character as output. Since the number of elements in a card is usually much larger than n+1, and it is important to the generation of a card that all previous characters can have influence on the card as a whole, Long Short-Term Memory (LSTM) layers were used. Both our RNN models used the Adam optimizer with a Softmax activation on the output Dense layer, which are the generally recommended optimizations for deep learning [2]. The word-based model includes an embedding layer, as well as an additional Dense layer, this time with a Rectified Linear Activation Function (ReLU) activation. This is to help with saturation in the learning process [5], since the word data has an approximately 112 times larger vocabulary (10,463 words) compared to the character data (93 characters).

A character-based RNN was chosen because it provides the ability to generate new words that haven't been seen by the model before. For areas of the card such as the name and type, new words are welcome. Each card should have a unique name, which will sometimes reference fantasy names from the game's story (which is constantly being developed), so the ability to create new names for people, places, or creature types is helpful, even when the new word does not exist in English.

While a word-based RNN does not have the ability to generate new words like the character-based RNN, looking at sequences at the word-level lets us have a larger scope, and leverage information from much in the card than the character-based RNN was able to see. This allows the word-based RNN to form more cohesive and well-formed sentences in the rules portion of the card.

The technique used to generate cards using the trained models saw two primary versions:

1. Always pick maximum-probability output.

2. Pick weighted random choice probability output.

With the first version, we start with the start flag input, then pick the next predicted element (character or word) with the maximum-probability. That element is added to the input, and this process is repeated until the end flag is generated. In the second version, instead of always picking the maximum-probability element, the next element was chosen based on a random weighted-probability. For example, if an input has two possible outputs with respective probabilities [0.9, 0.1], instead of always selecting the first output since it has the highest probability, it would be selected only 90% of the time, with the second output being selected the other 10%. Only the character-based RNN never used the first technique, since the second technique was found to be much more effective once it was devised.

3.2.1 Character-Based RNN Model

Model	Layers	Sequence Length	Output Shape	Total Params
1.1 - 1.3	1 LSTM 1 Dense	10+1 (1.1) 20+1 (1.2) 15+1(1.3)	$(75) \rightarrow (93)$	57,768 (1.1, 1.2) 86,993 (1.3)
1.4	2 LSTM 1 Dense	15+1	$(15, 100) \rightarrow (100) \rightarrow (93)$	167,393

Table 3.1: Summary of the iterations of the character-based model.

The initial iteration of the character-based RNN, which we will call Model 1.1, consists of a single LSTM hidden layer and a single Dense output layer, with a sequence length of 10+1 (see Table 3.1). In Model 1.2, the sequence length was increased to 20+1 to let the model see a larger pattern of the input characters in the sequence when training. Results from Model 1.2 showed that longer sequences helped generate more cohesive cards without errors. To see if a slightly smaller sequence length would yield similar results, in Model 1.3 the sequence length was decreased to 15+1. In addition, the number of LSTM units was increased from 75 to 100, to help the model retain more information about the sequence.

In Model 1.4 a second LSTM layer was added to once again encourage retaining more information about the character sequence to help form more cohesive cards. With the success of the second LSTM layer in Model 1.4, a third layer was added to investigate if it had any further benefit, but there was no significant difference in the results. Table 3.1 summarizes the Model parameters followed by Table 4.1 with the validation results.

We observed that the ability of this approach to generate new words is more detrimental

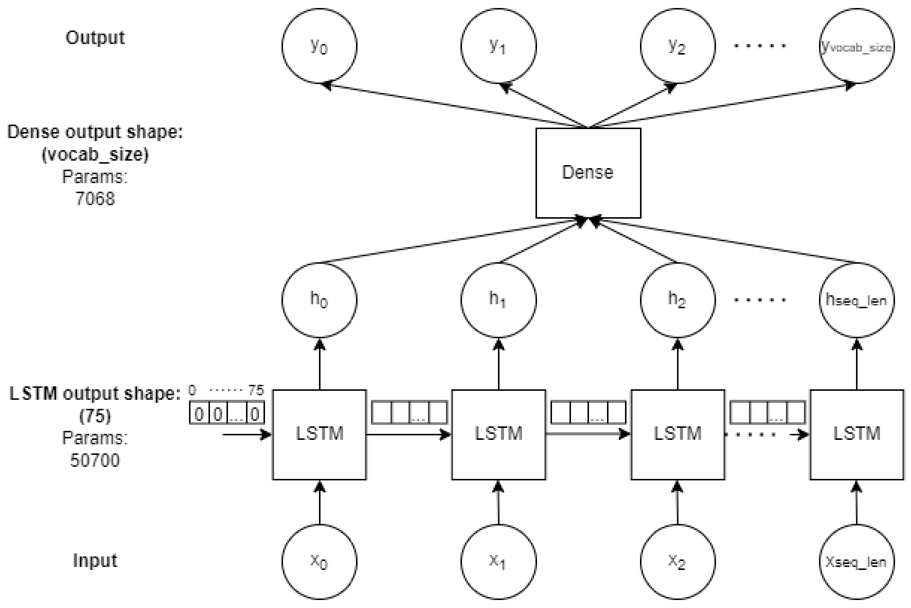

Figure 3.4: RNN architecture of models 1.1 - 1.3.

than helpful when it comes to the rules text. This portion of the card uses a particular vocabulary, so any new words would need to 1) already exist in the English language, and 2) make grammatical sense in the sentence. When a new word is generated, it contradicts the above points. Hence, we decided to use a word-based model to address this issue.

3.2.2 Word-Based RNN Model

The initial iteration for the word-based RNN, Model 2.1, started with what was found to be most successful with the character-based RNN: 2 LSTM layers with 100 units. This first iteration investigated using a higher sequence length of 50+1. The low quality results of Model 2.1 (see Results) confirmed that other sequence lengths would need to be investigated. A length of 10+1 was used for Model 2.2. While this is a much smaller length than in Model 2.1, it is still a significant upgrade from the scope of the character-based RNN.

After seeing how Model 2.2 was weak at generating the start of a card but strong at generating the rules text (see Results), we went back and rethought the order of the information in the text sequences. Originally, the order used was simply the order the

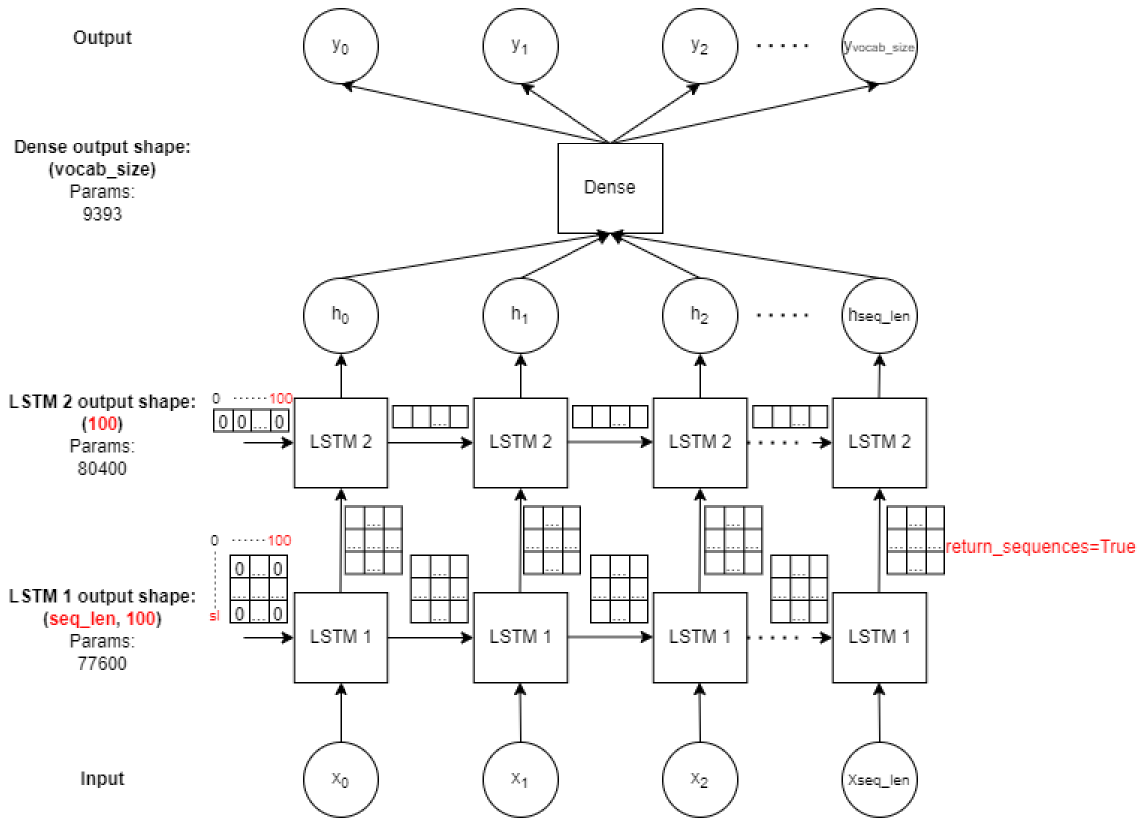
Output
Dense output shape: (vocab_size)
Params: 9393
LSTM 2 output shape: (100)
Params: 80400
LSTM 1 output shape: (seq_len, 100)
Params: 77600
Input
Dense
LSTM 2
LSTM 1
return_sequences=True
y_0 y_1 y_2 y_{vocab_size}
h_0 h_1 h_2 h_{seq_len}
x_0 x_1 x_2 x_{seq_len}
0 100

Figure 3.5: RNN architecture of model 1.4.

information appears on the card, but since the word-based RNN appeared to perform better with the rules text, that portion was put at the end. This was done to 1) investigate the effects of the order of information on the learning of the model, and 2) to set up the word-based RNN in such a way that output generated from the character-based RNN could potentially be used as input for the word model to generate the rules text. Additionally, since the name of the card is irrelevant to the word-based RNN, it was removed from the sequences. This also had the side effect of greatly reducing the word vocabulary, which made training faster and removed unnecessary words that should not get used in the rules text.

Model	Layers	Sequence Length	Output Shape	Total Params
2.1 - 2.2	1 Embedding 2 LSTM 2 Dense	50+1 (2.1) 10+1 (2.2)	$(50, 50) \rightarrow (50, 100) \rightarrow (100) \rightarrow (100) \rightarrow (10464)$ (2.1) $(10, 10) \rightarrow (10, 100) \rightarrow (100) \rightarrow (100) \rightarrow (10464)$ (2.2)	1,730,964 (2.1) 1,296,404 (2.2)
2.3	1 Embedding 2 LSTM 2 Dense	10+1	$(10, 10) \rightarrow (10, 100) \rightarrow (100) \rightarrow (100) \rightarrow (2226)$	381,986

Table 3.2: Summary of the iterations of the word-based model.

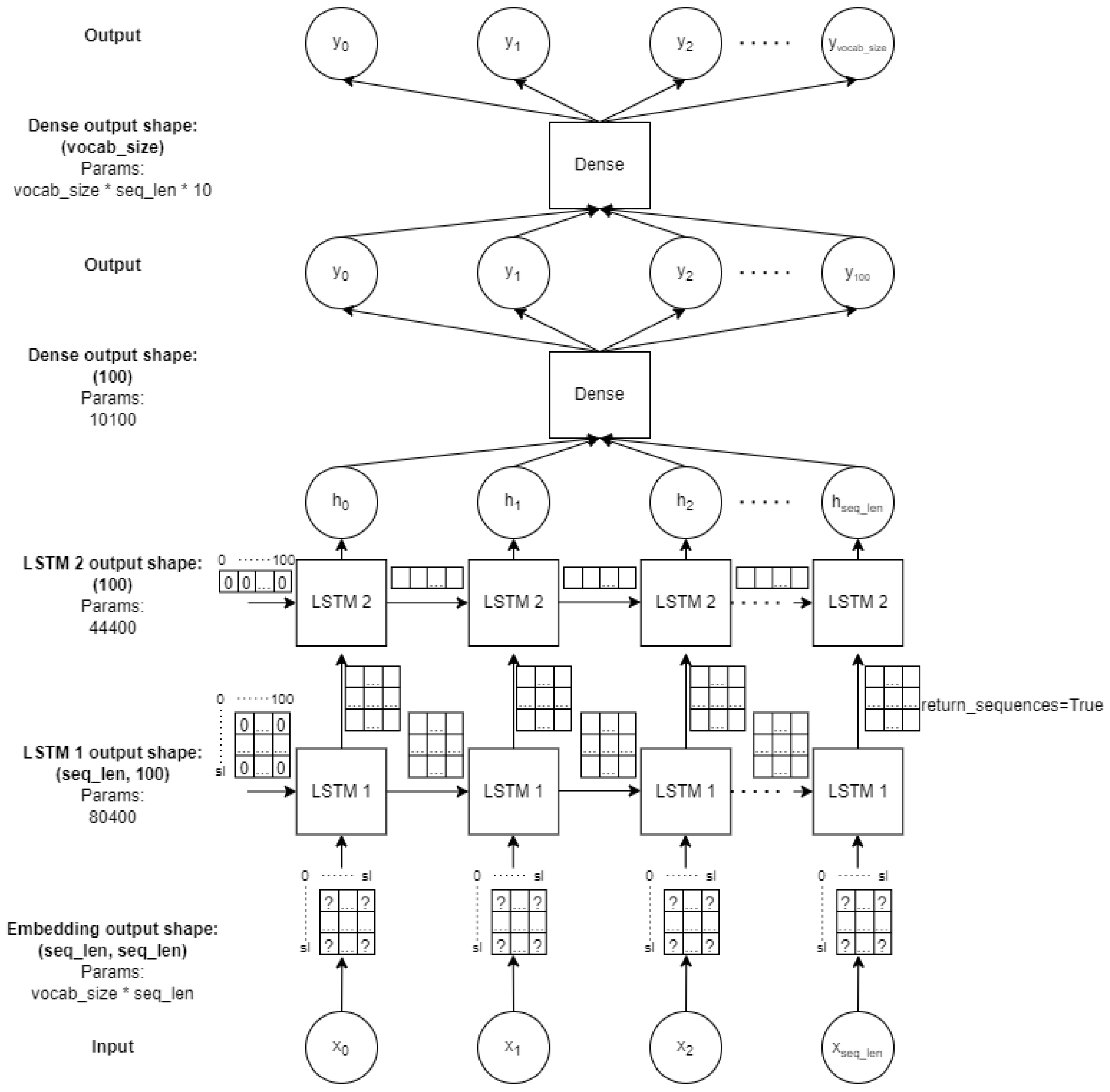

Figure 3.6: RNN architecture of models 2.1 - 2.3.

Chapter 4
RESULTS

The character-based models outperformed the word-based models in both loss and accuracy (see Table 4.1, Figure 4.1, and Figure 4.2). Loss describes the difference in the model's predicted value and the true value [1]. Accuracy describes how often the model predicts the correct value [1]. While plain loss and accuracy are based off the training data, validation loss and validation accuracy uses a small subset of the dataset that's been set aside for evaluation. The validation results show how the model performs with input it hasn't seen before. Part of the reason for this is most likely because the word-based model has a much higher vocabulary (10,463 words) compared to the character-based model (93 characters). This means that the prediction probabilities are spread much thinner, and will be less likely to match up with the true output from the dataset.

In our evaluation, we use cosine similarity to roughly score the generated cards on their semantic correctness, but this method is not perfect and requires some human observation as well. Likewise, whether or not a card adheres to the game rules and/or provides in-game value requires human evaluation. As a consequence, some cards have been evaluated as "good", despite having lower cosine similarity scores compared to other "poor" examples produced by the same model.

Next, let's look at the quality of cards being generated using the character-based RNN and word-based RNN.

4.1 Character-Based RNN Results

The initial models 1.1 and 1.2 already do a good job at generating cards that are well-formed. That is: each field is generated, there are little-to-no errors, and the text makes sense both grammatically and within the rules of the game. The set of results is very small, since maximum-probability selection was solely used to generate cards from these

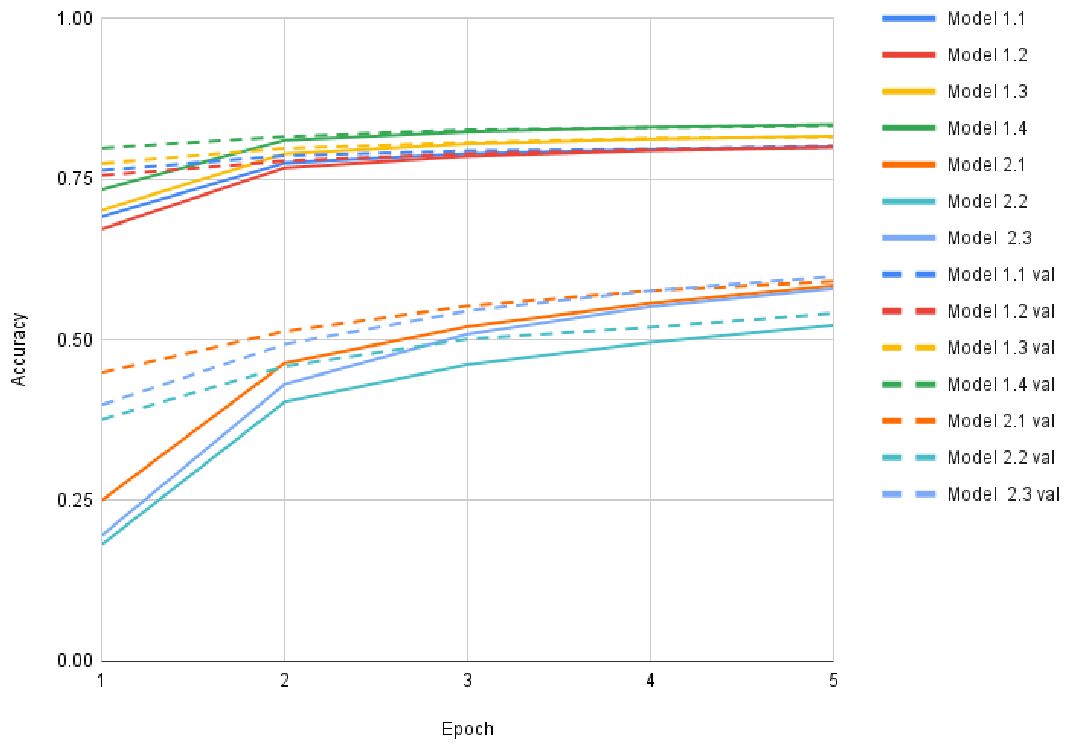

Figure 4.1: Loss and validation loss of all model iterations. These results are from 5 epochs of evaluation done after training. Actual training of models was not 5 epochs.

models, which resulted in the same card being generated each time. The poor examples (see Table 4.2) were the card generated from the seed input of just the start flag, and the good examples were a result of continued generation past that– using the previously generated card as input. Besides a slight error of a single character, these first two models were able to generate well-formed cards.

Starting with Model 1.3, we started to use weighted-probability selection when generating cards. This was beneficial because it allowed for a much wider variety of cards to be generated from a single model. This variance allowed for more examples to help judge the performance of the model. The main drawback of this is that it opens cards to the possibility of straying too far off the path, such that whole fields of the card might get skipped, nonexistent words might get created, or the sentence structure might suffer greatly

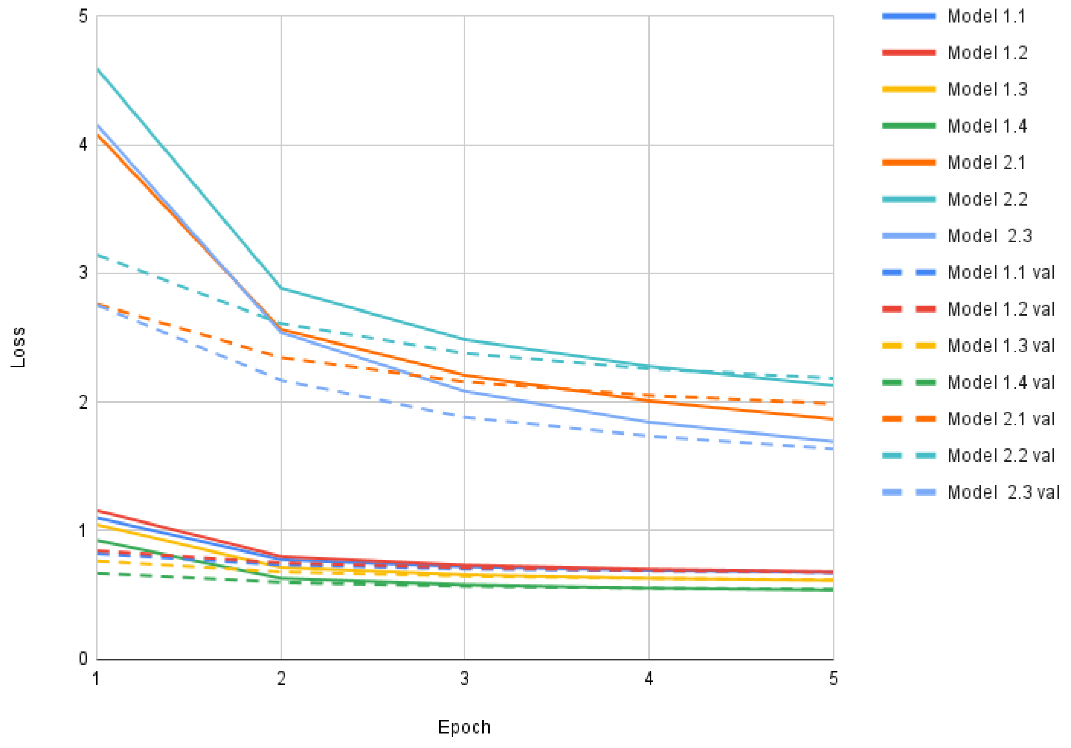

Figure 4.2: Accuracy and validation accuracy of all model iterations. These results are from 5 epochs of evaluation done after training. Actual training of models was not 5 epochs.

(see Table 4.2).

4.2 Word-Based RNN Results

The weaknesses of the word-based RNN were more severe and apparent than those of the character-based RNN (as seen in accuracy results). In Model 2.1, stopwords such as “a”, “an”, “any”, or “the”, were heavily favored at the start of the card (see Table 4.3). When the predicted probability of a certain word becomes very high, the difference in performance between maximum-probability selection and weighted-probability selection becomes smaller. With such a long sequence length (50+1) used to train Model 2.1, it’s understandable how the gap in probability between stopwords and other words could be significantly wider. Eventually, the card started to have a better vocabulary, but the phrases did not make

Model	Loss	Validation Loss	Accuracy	Validation Accuracy
1.1	0.79246	0.72472	0.77022	0.78820
1.2	0.81386	0.73498	0.76354	0.78358
1.3	0.73182	0.66728	0.78478	0.80136
1.4	0.64524	0.58756	0.80626	0.82028
2.1	2.5449	2.26008	0.47502	0.53616
2.2	2.87236	2.51478	0.41284	0.47912
2.3	2.46166	2.03486	0.45324	0.52212

Table 4.1: Mean loss and accuracy of all RNN model iterations.

sense. Decreasing the sequence length in Model 2.2 prevented stopwords from being favored too heavily, but there were still struggles with starting the card off with the right format. However, the structure of the creature type and rules text portions was improved. In some cases, such as the good example from Model 2.2, the generated rules text was identical to the existing "forestwalk" ability in the game. This may be a case where the weighted random probability selection technique chose the highest probability word a very high number of times in a row.

In Model 2.3, the format of the input data was changed (see Implementation). The card name was removed, and the rules text was moved to the end of the line. Once again, this model struggled to start cards with the right format. The mana cost and/or creature type of the card was often skipped. However, the model was able to create well-formed rules texts.

The character-based models were successful in generating interesting and unique creature names, as well as varied mana costs, and reasonable creature types. The character-based models appeared weaker when it came to generating the rules text. The ability for the character-based models to generate words that did not appear in the training text is a strength for the card name and creature types where 1) new and interesting sounding words are welcome, but not in the rules text and 2) it is especially important to form complete sentences using correct and meaningful words. Fortunately, this is where the word-based models shined. The restriction of only being able to generate words that have previously been seen, paired with the benefit of being able to look at longer sequences (a sequence of 10 words is much longer than a sequence of 10 characters), allowed the word-based models to generate well-formed sentences with a reasonable adherence to the structure of rules text.

This led to the idea of using both models in tandem. By reordering the fields so that the rules text appears last, we can use the character model to generate the card up until the rules text, and then use the word model to generate the rest of the card. This resulted in cards that had consistently strong names, mana costs, and creature types, as well as strong rules texts (see Table 4.3). These cards were almost always in correct format and grammatically correct. However, there are still cases where the sentences don't break grammar rules, but fail to make sense or adhere to game rules, such as the "poor" example in Table 4.4.

Model	Poor Example	Score	Good Example	Score
1.1*	S|| Legot Spirit | flying when @ enters the battlefield, you may put a creature token with flying.) | 2/2 ||E	0.419	S|| Storm Sage | {2}{B} | Creature Human Wizard | flying when @ enters the battlefield, you may put a creature token with flying.) | 2/2 ||E	0.524
1.2*	S|| {t}: target creature gets +1/+1 until end of turn. | 1/1 ||E	0.443	S|| Shaman Spirit | {2}{R}{R} | Creature Giant | flying whenever @ attacks, it gets +1/+1 until end of turn. | 1/1 ||E	0.390
1.3	S|| /lan may Creature Hentawk | @ can't be blocked except by artifact creature. if you do to tap enters the battlefield, target creatures. | 6/6 ||E	0.476	S|| Limin | {3}{W} | Creature Human Wizard | {t}: target creature gains flying until end of turn. | 1/1 ||E	0.494
1.4	S|| Chopline Bird | escape—{w}: target knight permanent. | 1/1 ||E	0.050	S|| of the Seaper | {1}{W} | Creature Human Vampire | whenever a creature you control attacks, it gets +1/+1 until end of turn. | 5/5 ||E	0.607

Table 4.2: Selection of example outputs from character-based models. Scored using cosine similarity between the generated card an a single document containing all existing cards.
* Output from these models only used maximum-probability selection.

Model	Poor Example	Score	Good Example	Score
2.1	s|| a an - a any the an a an - exile the exile 2 2 cards 1 2 {c} 2 2 2 2 white creature token you mana dealt phase combat . | 2 / 0 ||e	0.302	s|| any the a an - any a an combat end the an anywhere a any combat the and goblin warrior : this creature deals 1 2 + 1 {r} : regenerate @ deals 1 damage + 6 until end of combat . | 1 / 1 ||e	0.316
2.2	s|| other . - : the giant , put up to one cards an opponent owns . @ becomes a , it can't be blocked . % whenever a player casts an instant or sorcery spell , @ gains trample until end of turn . (to manifest the next 1 damage that would be dealt this turn . (exile the spell with fewer permanents from anywhere until you lose 1 life . | 3 / 4 ||e	0.332	s|| target . - {u} {u} | creature elemental monk | forestwalk (this creature can't be blocked as long as defending player controls a forest .) | 2 / 1 ||e	0.330
2.3	s|| / / / 2 / 5 | (an opponent wins a nontoken humans get + 1 / + 1 . % whenever you cast a wall more or more more cards are less . you may put that card on the bottom of your library face exile a land and this way card tails {g} . x is more . ||e	0.261	s|| artifact creature elf wizard | 1 / 1 | g g , {t} : target creature gets + 2 / + 0 until end of turn . ||e	0.502

Table 4.3: Selection of example outputs from word-based models. Scored using cosine similarity between the generated card an a single document containing all existing cards.

Model	Poor Example	Score	Good Example	Score
1.4/2.3	s|| {2} {b} {g} | creature elf cleric | 2 / 2 | {1} {r} , {t} , discard a card : @ deals x damage to target opponent destroy that creature . ||e	0.424	s|| {3} {w} {u} {b} {r} | legendary creature human pirate | 6 / 5 | trample % {r} : @ gets + 1 / + 1 until end of turn .) % {b} {r} : @ gains deathtouch until end of turn . ||e	0.349

Table 4.4: Selection of example outputs from the combined char-based and word-based model. Scored using cosine similarity between the generated card an a single document containing all existing cards.

Chapter 5
CONCLUSION

We began with the intention of comparing the effectiveness of character-based and word-based RNN models for generating MTG cards. We implemented 4 different iterations of the character-based RNN and 3 different iterations of the word-based RNN, as we learned more about what kind of input and parameters resulted in better output. It's important to minimize vocabulary, by converting all text to lowercase (except for unique symbols that use alphabet characters, such as mana costs), and removing card features that are not important to the learning.

We saw that the character-based RNN models generally performed better when it came to loss and accuracy evaluations. However, with the same input sequence length, the word-based model sees a wider scope of the card compared to the character-based model. This made the word-based model more valuable for forming whole, complex sentences in the rules text. Because of this, the word-based model was more effective at generating grammatically correct phrases.

The character-based RNN models were better at producing varied card features in the correct format, outside of the rules text. Because the strengths and weaknesses of the two models complemented each other, the input data was reformatted to support using a combination of the models to generate cards; the character-based model would generate everything before the rules text, and the word-based model would generate the rules text last. This resulted in the most consistent and novel generated cards so far.

Chapter 6
FUTURE WORK

The models described in this thesis have applications for CCGs beyond just MTG. Almost all CCGs (Pokemon, Digimon, Yu-Gi-Oh, etc.) share similar features on their card, such as a name, some sort of cost(s), some values to indicate "strength", and often some kind of type and/or subtype. These features, like MTG, are usually portrayed using a combination of natural language and symbols. The models may need to be refined if the data for a different type of CCG card is significantly different, but if the card information can be represented in a data format similar to what was described above, the models will have success with those CCGs as well.

We are currently refining the format of the data to consider more edge cases that weren't considered before, as well as investigating new selection methods for generating cards that strike a balance between the accuracy of maximum-probability selection and the variability of weighted random choice selection.

Our cosine similarity function compared the entire output of a generated card (start flag, end flag, and all card features) to a compilation of all existing cards from the data set. This was so the score could also reflect successes and failures in replicating the data format, as well as all features of the card. However, since the format is something that can easily be checked, it may be more important to only look at the rules text, which uniquely has free-form natural language, unlike other card features.

While our cosine similarity function could be improved, natural language is much more nuanced than this scoring method, and should be supplemented by more sophisticated scoring techniques and/or human observation. The next step is to design a more accurate scoring function, so that potential card designers using these models won't have to sift through low-quality cards. But, cards generated by these models should ideally also be evaluated by MTG players to determine their correctness, uniqueness, and gameplay value.

Chapter 7
APPENDIX - GENERATED CARDS

Generated using model 1.2:

```
Shaman Spirit {2}{R}{R}
Creature – Giant


Flying
Whenever Shaman Spirit attacks, it gets +1/+1 until end of turn.
1/1
```

Generated using model 1.3:

Lums {2}{B}

Enchantment Creature – Hugau

Mutale — Lums enters the battlefield.

{2}{r}, discard this card: search your library for a plate.

1/1

Limin {3}{W}

Creature – Human Wizard

{t}: target creature gains flying until end of turn.

1/1

Jactal {1}{U}

Creature – Leech

3/2

Hartadi's Tereed {1}{W/U}{W/W}

Creature – Elk

When Hartadi's Tereed enters the battlefield, create a 2/2 black baidskirge.)

4/4

Generated using model 1.4:

Crasher {2}{B}

Creature – Human Cleric

Lifelink

When Crasher enters the battlefield, copy it ontowasw.

4/4

Ascelland Kranka {1}{U}

Creature – Faerie Wizron

Ascelland Kranka enters the battlefield, each opponent loses 1 life.

2/3

Us, the Racketh {2}{G}

Creature – Elf Druid

When Us, the Racketh enters the battlefield, if it enters the battlefield, denture that was next turn. Encole a creature card from a graveyard, sacrifice
Us, the Racketh from a graveyard.

2/2

---

fiend {3}{U}

Creature – Human Warrior

Whenever you cast a spirit or arcane spell, create a 1/1 white human creature token at heter tokens.

Delirium — at the beginning of your end step, exile two cards.

{1}{B}: fiend gets +1/+1 until end of turn.

1/1

Generated using model 2.1:

```
a an - a any the an a an  exile the exile 2 2 cards 1 2 {c} 2 2 2 2 white creature
token you mana dealt phase
combat .
2/0
```

Generated using model 2.2:

```
target . - {U}{U}
Creature – Elemental Cleric


({U/B} can be paid with either {R} or {B}.)
{1}: Exile target creature.
4/3
```

```
target . - {U}{B}
Creature – Elemental Wizard


{T}: Another target blue creature gets +0/+2 until end of turn.
1/1
```

Generated using model 1.4 combined with model 2.3:

@ {7} Artifact Creature – Demon Flying Whenever @ enters the battlefield or attacks, create two 2/2 black zombie creature tokens. 1/4
@ {2}{B}{R} Creature – Lizard {B}{G}: @ gets +3/-3 until end of turn. 2/2
@ {3}{W}{U}{B}{R} Legendary Creature – Human Pirate Trample {R}: @ gets +1/+1 until end of turn.) {B}{R}: @ gains deathtouch until end of turn . 6/5

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] et al. Andrej Karpathy. Cs231n: Convolutional neural networks for visual recognition. https://cs231n.github.io/neural-networks-3/, 2022.

[3] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. *Poker as a testbed for AI research.* Springer, Berlin, Heidelberg, 1998.

[4] billzorn. Mtg encode. https://github.com/billzorn/mtgencode.

[5] Jason Brownlee. A gentle introduction to the rectified linear unit (relu). 2019.

[6] Francois Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[7] Peter I. Cowling, Colin D. Ward, and Edward J. Powley. *Ensemble determinization in monte carlo tree search for the imperfect information card game magic: The gathering. IEEE Transactions on Computational Intelligence and AI in Games.* 2012.

[8] Dauchen Zha et al. *RLCard: A Platform for Reinforcement Learning in Card Games.* IJCAI, 2020.

[9] Dauchen Zha et al. *RLCard: A toolkit for reinforcement learning in card games.* 2029.

[10] V. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[11] Magic: The gathering. http://mtg.fandom.com/wiki/Magic:_The_Gathering.

[12] Elizabeth Gibney. Google ai algorithm masters ancient game of go. *Nature News*, 529(7587):445, 2016.

[13] Ni Guo and Zehui Qu. The application of imperfect information game theory in social games. *Journal of Physics: Conference Series.*, 1883(1), 2021.

[14] Zach H. Mtgjson. https://mtgjson.com/downloads/all-files/#allprintings, 2021.

[15] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. Pcgrl: Procedural content generation via reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.*, 16(1), 2020.

[16] Jieh-Sheng Lee. *Measuring and controlling text generation by semantic search.* 2020.

[17] Zachary C. Lipton, John Berkowitz, and Charles Elkan. *A Critical Review of Recurrent Neural Networks for Sequence Learning.* 2015.

[18] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780, 2006.

[19] Reed Milewicz. Generating magic cards using deep, recurrent neural networks - custom card creation - magic fundamentals - mtg salvation forums. http://www.mtgsalvation.com/forums/magic-fundamentals/custom-card-creation/612057-generating-magic-cards-using-deep-recurrent-neural, 2015.

[20] Wizards of the Coast. *Magic: The Gathering Comprehensive Rules.* Wizards of the Coast, Renton, WA, USA, 2022.

[21] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1, 2012.

[22] Mark Rosewater. Mechanical color pie 2021. *Making Magic*, 2021.

[23] J. Rubin and I. Watson. Computer poker: A review. *Artificial Intelligence*, 2011.

[24] Robert Shaffer and Zachary Elkins. On measuring textual similarity. 2019.

[25] David Silver, Julian Schrittwieser, and Karen Simonyan et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[26] Ronaldo Vieira, Luiz Chaimowicz, and Anderson Rocha Tavares. *Reinforcement Learning in Collectible Card Games: Preliminary Results on Legends of Code and Magic.* SBGames, 2019.

[27] Kevin Webb. With more than 35 million players worldwide, magic the gathering is giving back to its community with a brand new game and $10 million in esports prize money. *Business Insider*, 2018.

[28] How to play. https://magic.wizards.com/en/how-to-play.