Learning Semantic Embeddings of *Magic: the Gathering* with a Multimodal Transformer Architecture

Matthew Bollinger
Georgia Institute of Technology
mbollinger6@gatech.edu

Jordan Reef
Georgia Institute of Technology
jreef3@gatech.edu

Abstract

Our project evaluates whether deep learning models can effectively draft cards in Magic: the Gathering. First, we draw on online draft data to generate word2vec embeddings for each card in a common dataset. Then, we fine-tune a BERT model to learn these embeddings based on the card text. With both approaches, we build an "agent" that, using the embeddings and a simple heuristic, makes draft decisions in a range of scenarios constructed with domain knowledge. Based on our tests, we conclude that our agent is only partly effective at making draft decisions. However, we highlight several ways in which both our approaches capture meaningful card information, and suggest possible directions for future research.

1. Introduction/Background/Motivation

Magic: the Gathering is a popular card game owned by Wizards of the Coast that has many interesting complexities. It is a two-player zero-sum stochastic game with imperfect information. Optimal play is highly nuanced and often difficult for even the best human players (and has formally been shown to be Turing Complete [3]).

Before playing against an opponent players are required to construct a deck from a potentially vast number of cards throughout *Magic's* history. The size of the card pool varies widely depending on the format being played, ranging from as many as 30,000 in Vintage, to as few as 300 in others. Regardless of format, constructing a functional deck relies heavily on the player's ability to evaluate how well each card complements their intended strategy.

We sought to apply tools from natural language processing to create a deck-building agent. Several aspects of the deck-building problem make it interesting from a deep learning perspective:

1. Evaluating a card with respect to some set of strategies is essentially a semantic analysis problem that depends on the natural language interpretation of a card's rules text.
2. Strategic inter-dependencies between cards are very complex.
3. Even expert human players often make errors when evaluating new cards in a zero-shot setting, though players' evaluations improve as they gain more gameplay experience with those cards.

Wizards of the Coast releases a new set of approximately 300 cards every few months which can heavily influence existing formats and which are each a novel playable format unto themselves. A system that could accurately predict which cards are effective in which strategies early in a format's life cycle, would be an invaluable tool in augmenting players' preparation for competitive events (which offer substantial monetary prizes). Further, the application and analysis of such a system may aid players in exploring the space of alternative deck building strategies, and in the best cases may reveal completely novel (yet viable) strategies (as has been shown for other games like Go and chess [11]).

Our principal data set consists of deck lists sourced from 17Lands (https://www.17lands.com/), which is an extension for Magic: The Gathering's digital client that aggregates gameplay data from its users. This data corresponds specifically to the Draft format, in which 8 players construct their decks simultaneously by opening a booster packs. Each player selects one card from the pack, then passes the remaining cards clockwise to an adjacent player. That player then selects another card from their new, smaller, pack and passes the remaining cards once again. The process continues until each players' deck is completed.

2. Approach

We attempted to develop an embedding scheme for *Magic: the Gathering* cards that captures their complex strategic interrelationships. Drawing inspiration from nat-

Set Name	Decks	Training Pairs
Wilds of Eldraine (WOE)	100,000	55.2M
Lord of the Rings (LTR)	50,000	27.6M
Neon Dynasty (NEO)	10,000	5.5M

Table 1. Training data sizes

Hyperparameter	Value
Embedding dimension, D	300
Subsampling rate, ρ	0.001
Learning rate	0.001
Batch size	128
Optimizer	Adam

Table 2. Best (WOE) card2vec Hyperparameters

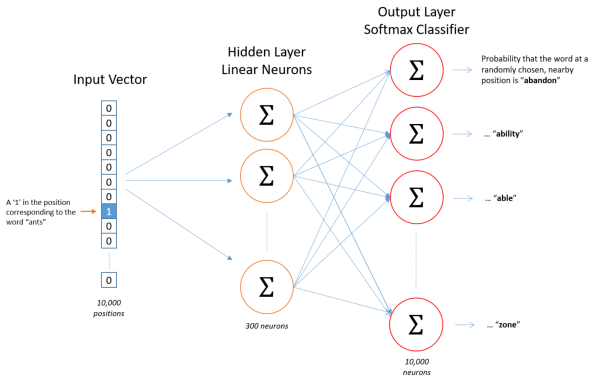

Figure 1. Example feed-forward word2vec architecture

ural language processing, we developed a procedure heavily inspired by word2vec [8] which we call card2vec, trained directly on the statistical associations between cards. We developed a multimodal training procedure that utilizes the embeddings generated by card2vec as prediction targets for fine-tuning BERT, a common baseline used for language modelling and other natural language processing tasks [4] (code at https://github.com/Tommyhillpicker/mtg_card2vec).

We gathered deck lists from 17Lands users for 3 different sets, *"Kamigawa: Neon Dynasty (NEO)"*, *"The Lord of the Rings: Tales of Middle-earth (LTR)"*, and *"Wilds of Eldraine (WOE)"*. For each of these sets, we utilized a different number of deck lists in order to test how much the quality of our embedding depended on the size of the training set (Table 1)

2.1. card2vec

Whereas word2vec aims to generate efficient vector representations of words in a vocabulary of size V, card2vec aims to generate efficient vector representations of cards in a set of size S (i.e. all 326 cards in *"Wilds of Eldraine"*).

We use the simple feed-forward neural network architecture described in [8] to achieve this. It consists of an embedding layer, fully connected linear layer, and a softmax activation. The input to the network, the *target* card, is simply a [0, S)-bounded integer which represents one card's unique index within the embedding layer (i.e. a one-hot representation of that card).

The key design consideration is D, the embedding dimension. A small choice of D may have insufficient capacity to learn the problem, while larger choices are harder to train and may overfit. In practice, we found that the "industry default" for training word embeddings, $D = 300$, yielded the best results. Configurations with $D \lessapprox 225$ were seemingly unable to learn at all, suggesting that the complexity of the interrelationships between *Magic* cards is at least as complex as words in natural language.

Another key parameter is the size of the context window, C, which controls how many words around the *target* word are considered its context. The training data for the procedure are pairs are generated based on the target word and each of its context words, which destroys the local relative ordering between words and their contexts. This is not a drawback for us since the relative ordering of cards within a deck list does not matter. We consider each of the 39 other cards in each 40-card Draft deck to be any given sample's context.

For performance considerations, we apply frequency-based subsampling to the deck lists prior to generating training pairs. This has the additional benefit of dropping cards that appear too frequently (and thus provide diminishing informational gain) in the data set [9]. The subsampling rate is controlled by its own hyperparameter, ρ. For our choice of ρ, our average deck size was reduced from 40 to approximately 24, yielding approximately 552 training pairs per deck.

card2vec's loss function is the Cross Entropy between the softmax probabilities resulting from the forward pass of the network, and a one-hot representation of the context card, *Ctx*.

$$\mathcal{L}_{card2vec} = CE(Softmax(T \times L),\ Ctx) \qquad (1)$$

The resulting (S x D)-shaped weights of the hidden layer are the learned card vectors.

2.2. BERT Finetuning

For the natural language portion of the project, we fine-tuned a BERT model, training it to reproduce the card2vec embedding given the card's "oracle" (rules) text. We hypothesized that each card's text would contain enough information on its function for the BERT model to situate it in the embedding space.

We did anticipate difficulties learning some of the more complex relationships between cards. If a card is valuable as a counter to a different card, for example, that role will be difficult to ascertain from the card text. We were also concerned with the curse of dimensionality, since the model would be producing 300-dimensional embeddings from a dataset of under 300 entries. Attempts at dimensionality reduction, though, proved ineffective, indicating that a many-dimensional space was necessary to accurately represent a card's role.

To implement the BERT finetuning, we drew on Chris McCormick's tutorial, "Bert Finetuning with Pytorch" [7]. The implementation in the tutorial consists of a BERT model followed by a single fully-connected layer. Since our problem was more complex than the tutorial's, we added a multi-headed attention layer between the model backbone and the fully-connected layer, which proved effective in testing.

Our early results with the BERT finetuning were discouraging. Initially, we trained our model with a loss function defined as a ReLU of one minus the cosine similarity with the card2vec embeddings. Although the validation loss did decrease, we discovered that the model was producing very similar outputs for all cards. Rather than learning different card embeddings, the model was minimizing the loss by mapping all cards close to the centroid of the target data.

The main innovation we introduced to fix this was to apply techniques from multimodal learning to our problem. In this case, the two "modes" of the data were the cards' text embeddings and their card2vec representation. We then redefined our loss function following the methodology of Lazaridou et. al [6]. Drawing on models connecting language and vision, they recommend setting loss to a margin, minus the cosine similarity to the "right" card, plus an averaged cosine similarity to several "wrong" cards (3). This loss function encourages the model to produce outputs that are further apart from one another, rather than clustered about a centroid.

The model proved highly sensitive to hyperparameters, particularly the batch size and learning rate. Compounding the difficulty, the optimal hyperparameters varied greatly throughout the process, as we gradually improved the card embeddings. Based on Smith et al., we believe that these two parameters determine the model's ability to learn by influencing the "noise scale" [12] [5]. Self-attention layers, having great representational power, are highly sensitive to noise; too much and the model will fail to learn, but too little and the model will struggle to generalize. Since our BERT-based model effectively has 13 self-attention layers, choosing the appropriate dosage was vital. The low learning rate of 2e-5, which is required to avoid catastrophic forgetting, would have exacerbated vanishing gradient problems, making the model still more sensitive to small perturbations.

$$\mathcal{L}_{BERT} = max\Big(0,\ 1 - \Big(\frac{B \cdot C}{\|A\|\|B\|}\Big) + \frac{1}{I}\sum_{i}\Big(\frac{B \cdot \neg C_i}{\|A\|\|B\|}\Big)\Big) \tag{2}$$

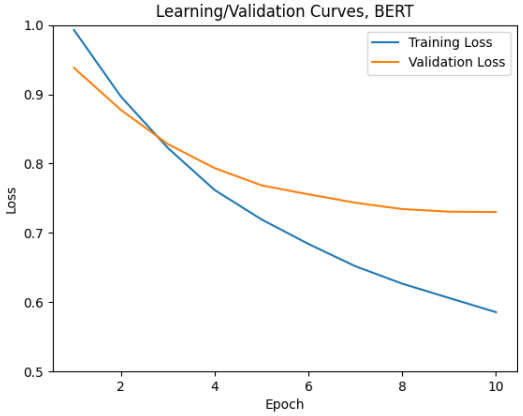

Figure 2. Learning Curve, BERT

3. Experiments and Results

3.1. card2vec and Related Evaluations

Various distance metrics between embedding vectors are often used to help evaluate the quality of the embedding. The best distance metrics to use can be domain-specific, although we focus on the cosine similarity because the magnitudes (which cosine similarity ignores) of word vectors produced by word2vec are often not semantically meaningful [1]. Here we provide details on the main evaluation tasks utilizing this metric we used to investigate the performance of our model:

- ***Pairwise Similarities.*** We record the cosine similarities between every pair of card vectors in a set after each training epoch.
- ***Curated Similarity Pairs.*** We focus on the pairwise similarities of a hand-picked subset of cards, divided broadly into one group of "similar" pairs and another group of "dissimilar" pairs. Under the hypothesis that the similarity scores between these two groups would diverge as the weights were trained to be semantically meaningful, we use the **"similarity difference"** between these two groups as a key metric in evaluating the quality of our embedding over the course of training:

$$\frac{1}{N}\sum_{n=1}^{N} Sim[n] - \frac{1}{M}\sum_{m=1}^{M} Dissim[m] \tag{3}$$

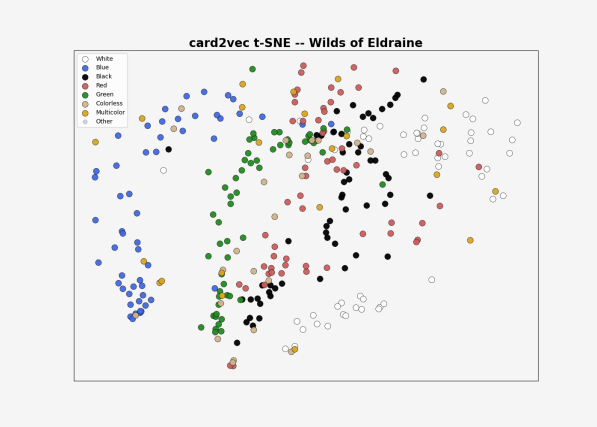

Figure 3. t-SNE visualization of the card2vec embedding trained on the set "Wilds of Eldraine (WOE)". Same-colored cards form apparent clusters

Selection Heuristic	Best	Mean
Additive	0.444	0.413
Multiplicative	0.555	0.398
No Embedding (only winrate)	0.222	0.222
No Winrate (only embedding)	0.444	0.337

Table 3. card2vec draft pick performance for different selection heuristics.

- ***Draft Picks.*** We created a procedure for making draft picks automatically. The procedure takes in a list of card choices from which it will make a selection, and a list of "context" cards, which represents an incomplete draft deck. The procedure initially performs a k-Means clustering (REF) over the vector representations of the context cards. It then takes the cluster to which the greatest number of context cards are assigned, and computes the cosine similarity between that cluster's centroid and each potential choice. The card with the highest similarity to this largest k-Means centroid is chosen.

  Separately, we gathered the win rate statistics for each card in our training sets from 17Lands and normalized them onto a range $[l, u]$, parameters of the procedure. We define four different "selection heuristics" based on how the win rates for each card are applied to it's similarity score, either by adding, multiplying, considering only win rates, or considering only similarity scores within the embedding.

Results of the card2vec procedure were somewhat difficult to interpret (discussed further in Section 4.1), but overall appear to have created embeddings that capture at least some of the semantic relationships between cards for the *WOE* set. Figure 3 contains a t-SNE visualization of the all card vectors in *WOE*. Five distinct clusters (corresponding to the 5 different color cards in *Magic: the Gathering*) can be observed, suggesting card2vec tends to places cards that share colors closer together in the embedding space. Similar color-clusters were observed across several t-SNE hyperparameter configurations.

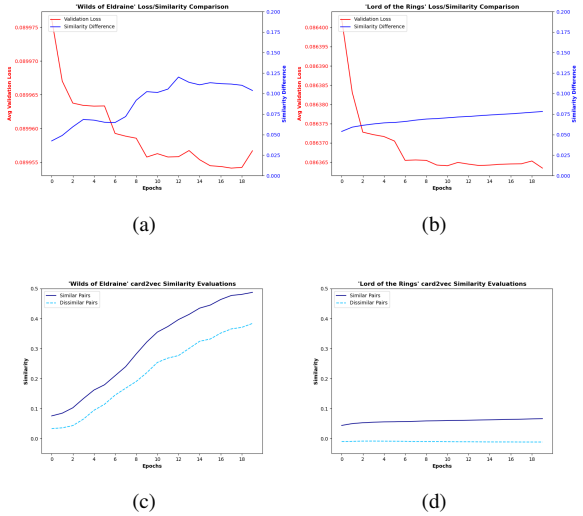

Figure 4. **(a)** card2vec Validation loss and 'similarity difference' for WOE **(b)** for LTR **(c)** Average similarity of 'similar' and 'dissimilar' card pairs for WOE **(d)** for LTR.

It is very apparent that the quality of the embedding heavily depends on the amount of input data, as demonstrated by the differences in training curves shown in Figure 4 for *WOE* and *LTR*. A persistent challenge across all hyperparameter configurations can be observed in Figures 4c and 4d: card2vec tends to assign relatively high similarities even to 'dissimilar' card pairings (though the overall "similarity difference" does tend to increase as loss decreases).

Table 3 shows the results of the draft pick selection procedure using the 'raw' card2vec embeddings. Selection strategies that incorporate the card2vec embeddings consistently outperform the "dumb" heuristic that selects cards with the highest winrate.

3.2. Multimodal BERT Results

We examined the top five most similar pairs produced by the BERT model to determine what it was learning. Encouragingly, four of the five pairs exhibited tactical synergies evident in the oracle text. "Lord Skitter, Sewer King" and "Rat Out," for example, both create rat tokens, which can be used in a Rat Commander deck. "Scream Puff" and "Sweettooth Witch" form a more sophisticated combo, where the former creates a food token that the latter can sacrifice to deal damage. From these and other instances, we conclude that the BERT model latches onto salient phrases to identify related cards.

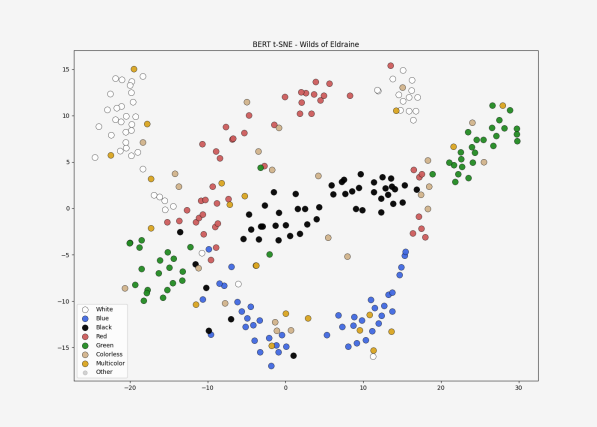

Figure 5. t-SNE visualization of embeddings from finetuned BERT model

Card 1	Card 2	Similarity
Armory Mice	Tuinvale Guide	.419
Charmed Clothier	Kellan's Lightblades	.669
Lord Skitter, Sewer King	Rat Out	.426
Mintstrosity	Scream Puff	.679
Scream Puff	Sweettooth Witch	.737

Table 4. Top 5 Similar Pairs, BERT Embeddings

We further tested the finetuned BERT model on intrinsic evaluation tasks drawn from both in-sample and out-of-sample data. The intrinsic evaluation tasks were binary "puzzles," in which the model was presented with a context card and tasked with selecting one of two cards to pair with it.

Our goals were to test whether it reproduced the card embeddings and evaluate how it generalized to out-of-sample data. As such, we created two series of puzzles, one from the Wilds of Eldraine data and another from the unseen Lord of the Rings set. For each series, we devised puzzles rated as "easy," "difficult," and "hard."

For the in-sample data, we presented the model with ten pairs that should be "similar" and four pairs that should be "dissimilar." We generated a similarity score for each card pair and took the mean over each group. The "similar" card pairs had an average similarity score of .445, with most being well above that; the "dissimilar" pairs, meanwhile, had an average of -.018, most being close to or below zero. We thus conclude that the model learned meaningful relationships between cards in the training set.

On the out-of-sample data, the model struggled somewhat, but still performed well overall. We trained the model using five different random seeds, which decided the training splits and weight initializations, and averaged its performance across runs. Across those five tests, it averaged getting six of the eight "puzzles" right on the Lord of the Rings data. Although it often missed puzzles we considered "easy," this still indicates that the model is learning patterns it can apply to out-of-sample data. Given that the model was only trained on a single set, this is an impressive result.

4. Discussion

4.1. Limitations of Evaluation

There exist many high-quality fully-annotated datasets in the domain of natural language processing (like SNLI [2], or SQuAD [10] which serve as reliable benchmarks on these tasks. Unfortunately, to our knowledge no similar dataset that could assist in the semantic analysis of *Magic: the Gathering* cards exists.

Thus, we were forced to hand-design our sets of "similar" and "dissimilar" card pairs and our draft selection puzzles based on our personal experience with the game. Doing so induces a labelling bias over these evaluation tasks that diminishes their integrity as training benchmarks. Even without resource constraints, the complexity of the game itself makes it unclear the extent to which arithmetic operations in card-embedding space or similarity metrics between different sets of cards could ever be successfully quantified and annotated by a human.

4.2. Future Work

Although more work remains to be done, our efforts to create a drafting agent were a partial success. The card2vec embeddings were able to meaningfully inform draft decisions on the WoE set, and the BERT model was able to make educated choices on out-of-sample data. We thus believe this remains a promising frontier for deep learning research.

Making expert draft decisions, though, seems to require more than just card embeddings and a heuristic. One possible improvement would be to train an agent with deep reinforcement learning. This agent, using function approximation, could learn more sophisticated ways to analyze the embeddings for previously-selected cards. Such an agent could also, potentially, learn more strategic aspects of drafting, such as picking cards that are unlikely to be available the next round.

Though difficult, the most effective way to improve the BERT-based model would be to find a way to train it directly on the training pairs from the card2vec portion. Training the model in an end-to-end manner would remove distortions introduced in the word2vec process and allow the model to learn from many different sets of cards. The training data from a single set is likely insufficient for the model to learn the game's complex mechanics. The challenge here is finding the right way to produce a loss signal from a set's training pairs.

Student Name	Contributed Aspects	Details
Jordan Reef	Data Creation, card2vec Implementation, Evals	Processed 17Lands data into training pairs. Implementation of card2vec portion of the model–performed related testing and evaluations. MtG gameplay “expert”
Matt Bollinger	BERT Training, BERT Evaluation	Trained the BERT model to recreate card embeddings, analyzed performance, and devised methods to evaluate it on in-sample and out-of-sample data.

Table 5. Contributions of team members.

5. Appendix

Figure 6. A pair of similar cards (top), and a less similar pair (bottom)

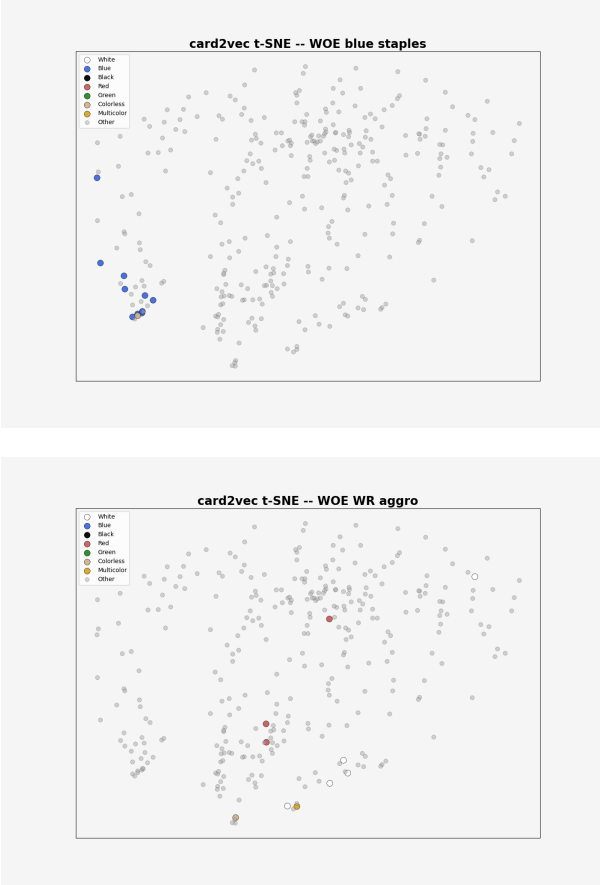

Figure 7. t-SNE visualizations highlighting small clusterings of strategically related cards

References

[1] Carl Allen and Timothy Hospedales. Analogies explained: Towards understanding word embeddings, 2019. 3

[2] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference, 2015. 5

[3] Alex Churchill, Stella Biderman, and Austin Herrick. Magic: The gathering is turing complete, 2019. 1

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 2

[5] Hang Hua, Xingjian Li, Dejing Dou, Cheng-Zhong Xu, and Jiebo Luo. Improving pretrained language model fine-tuning with noise stability regularization, 2023. 3

[6] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. Combining language and vision with a multimodal skip-gram model, 2015. 3

[7] Chris McCormick and Nick Ryan. Bert finetuning with pytorch, 2019. 3

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. 2

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013. 2

[10] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018. 5

[11] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. 1

[12] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017. 3