Vol.2021-GI-46 No.7
2021/6/19

CGDLを用いたトリックテイキングゲーム自動生成手法の提案

牧野 貴斗[1]　濱川 礼[2]

概要：本研究ではCGDL(A Card Game Description Language)を用いてトリックテイキングゲームのルールを記述し深層生成を行いトリックテイキングゲームを自動生成する手法を提案する．トリックテイキングゲームとは，ナポレオンやコントラクトブリッジに代表される全員でカードを決められた枚数出し．出したカードの大小などで勝敗を決めるトリックを複数回繰り返すゲームのことである．人の顔を見ながらできるカードゲームは，コミュニケーション能力を養い，ゲームに勝つために考える過程で思考力を養うことができる知育能力があると考えられている．ただ同一のゲームを繰り返しているとセオリーが確立し思考機会の減少，遊び慣れからゲームに飽きてしまい，カードゲームの持つ知育能力が損なわれてしまう．そこで，常に新しいゲームを提供することができればこの問題を解決できると考え，カードゲームの自動生成を行った．ゲームの自動生成分野では遺伝的アルゴリズムを用いてチェッカーやGoなどのバランスの取れたボードゲームを設計する試みなどがあるがこれらの研究では限られたバリエーションでしかルールを表現できていない．そこで本研究ではCGDLを用いてトリックテイキングゲームのルールを記述することで様々な形式に対応させ，これを深層学習で学習させることで，ルールをより多彩に表現することを試み，生成したゲームの評価を行った．

1. 背景と目的

カードゲームは沢山の人に親しまれており，だれもが一度は遊んだことがあるものである．人の顔を見ながらできるカードゲームは，コミュニケーション能力を養い，ゲームに勝つために考える過程で思考力を養うことができると考えられている[1][2]．そんなカードゲームであるが，大きな問題が2つある．一つ目は同じゲームを遊んでいるとゲームに飽きてしまったり，つまらなくなるという点である．この理由として，ゲーム慣れているプレイヤーが一方的に勝ってしまう，すでに確立しているセオリーをなぞって遊ぶだけになってしまうなどがあり，勝てないゲームや既存の展開をなぞるゲームがつまらなくなり，ゲームに集中できなくなる．そうなってしまうとカードゲームの本来持っている思考力を養う力が十分に発揮できなくなってしまう．二つ目はカードゲームは値段が高くかさばるものであり，そうそう簡単に買えるものではないという点である．[3]ではボードゲーマーにボードゲームの発展を阻害している要因をアンケート調査しているが，値段の高さは7位に入っている．仮に興味がある人がいたとしても遊んだことがないゲームを買うのは難しく，買ったとしても遊ぶには仲間で集まる必要があり，買ったとしても余り遊べないということもある．これらの理由から興味を持ったとしてもゲームを始めるための敷居が高く，気軽に始めることが難しい．そこで常に新しいゲームを提供することで新鮮な間隔でゲームを体験を利用者に提供することでこの問題を解決できるのではないかと考え，本研究ではCGDL(A Card Game Description Language)[4]を用いてトリックテイキングゲームのルールを記述し深層学習で学習する手法を提案する．

2. トリックテイキングゲームとは

[5]によるとトリックテイキングゲームとは，各プレイヤーが1枚ずつカードを出していき，全員が出した時に最も強いカードを出したプレイヤーが勝ちとなり，これを手札がなくなるまで繰り返すゲームと記述されている．図1のように全員がカードを出して勝敗を決める一連の流れのことをトリックと呼び，トリックに勝利した時ポイントを獲得することをテイキングと呼ぶ．対面の相手が仲間になるコントラクトブリッジや自分が勝利する回数を予測するナポレオンなど非常にルールのバリエーションが豊富で現在世界でプレイされているトランプゲームの半数がトリックテイキングゲームと言われるほどであり，それにもかかわらず根幹のルールはシンプルなカードゲームである．

1　中京大学 工学研究科　情報工学専攻
2　中京大学 工学部

図 2 のようにこのトリックに何回勝つかを競うのが，トリックテイキングの基本ルールだが，以下のようなルールを導入することでカードの大小をただ競うだけでない戦略性を持ったゲームである．

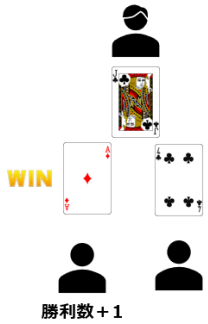

図 1　トリックの勝敗

図 2　ゲームの勝敗

2.1　切り札

最も強いマークを決めるルールである．数字の大小に加えて，指定されたマークと同じマークであるかどうかでカードの強さが変動するルールである．基本的に切り札と同じマークのカードの方が数字の大小にかかわらず強く，切り札のマークの指定方法として，「一番最初のプレイヤーが出したカードと同じマーク」という条件や「配ったカードとは別に山札から引いたカードと同じマーク」などが存在する．このルールによって，相手の持っているマークの種類などを推測する戦略性がゲームに加わる．

2.2　マストフォロー

切り札と同じマークのカードが手札にある場合に手札の同じマークのカードを必ず出さなければならないルールである．このルールがあることで，強いカードがあるのに同じマークではないため場に出すことができない，自分の手札の J や Q などの強いカードを相手が A で負けることが確定した状態で出さなければならないといったより戦略的な状況が生まれる．

3.　関連研究・関連システム

ボードゲーム自動生成の研究として，GGP(General Game Playing) の分野で異なるボードゲームを一律の言語で記述するために開発された GDL(Game Description Language) 言語を利用して，ボードゲームの自動生成を試みた研究がある [6]．この研究では，盤面のパターンや，駒の種類，勝利条件などあらかじめ用意されたパターンの中からゲームのバランスを考慮して，バランスの取れたゲームの生成を遺伝的アルゴリズムを用いて生成を行っている．

また，カードゲームの自動生成を目指す試みとして，‘A Card Game Description Language’ [4] がある．この研究では，GDL をカードゲームに改良を加えた言語を開発した研究で，カードゲーム用の言語の為，ボードゲームのようにどのような場があり，場のサイズなどの細かい設定を必要とせず，行う処理に関しても，より限定的な命令を採用している言語について述べている研究である．

4.　提案手法

カードゲームの自動生成を行うためには，カードゲームのルールを機械が理解しやすいように正確に記述する必要がある．本手法では CGDL を利用して，カードゲームのルールを記述することでカードゲームのルールを表現する．また，カードゲームはジャンルが多く，ゲームによっては全く違う目的を目指すゲームやカードを捨てるのではなく集めるなどの他のゲームと真逆の動作をするゲームも存在する．これらをまとめて学習させても望んだ出力を得るのは難しいと考え，記述するカードゲームのジャンルをトリックテイキングゲームのみにした．理由として，トリックテイキングゲームは 2 で述べたように根幹のルールはシンプルにもかかわらず豊富なルールを持っているゲームであり，シンプルなため学習を行いやすく，豊富なルールでバリエーションを作りやすいのではないかと思い採用した．トリックテイキングゲームのルールを学習する手法として LSTM[7] を用いる．LSTM は深層学習の一手法で [8] ではある特定の時間ステップにおける再帰層からの出力を次の時間ステップのへの入力の一部とすることで，時系列データを扱うことができるモデルだと記述されている．CGDL で記述したルールを単語単位で読み込ませることで次に来る単語を予測し，ルールを生成することができるのではないかと考えた．また，情報量が少ない場合でも適切な解を得られやすいため本手法に採用した．

本研究の提案手法を図 3 に示す．トリックテイキングゲームのルールを CGDL を用いて記述し，記述したデータセットを用いて LSTM モデルで学習を行う．学習後は，学習を終えた LSTM モデルに最初の単語を入れそのあとに続く単語の予測を行う．この作業を繰り返しトリックテイキングゲームのルールの生成を行う．

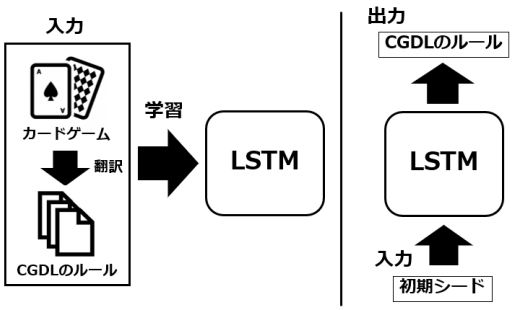

図 3　提案手法

4.1 データセット

今回使用するトリックテイキングゲームはトランプゲーム大全 [5] に記載されているゲームから内 40 個を使用している．トランプゲーム大全はトリックテイキングゲームを中心に 250 種のトランプゲームをまとめている書籍で，トリックテイキングゲームを数多く取り扱っている為採用した．

4.2 CGDL の解説

本手法で使用する，CGDL のルールについて説明する．CGDL(A Card Game Description Language) は，カードゲームのルールを RANKING と STAGES，WINNING CONDITIONS の三つの要素で記述した言語である．RANKING はゲームの役やカードの強さなどのゲームのルールを STAGES はゲーム中に行う手順を WINNING CONDITIONS は残ったトークンのポイントとゲームに残ったプレイヤーに与えられる得点を示している．本論文ではこのうちの STAGES 部分のみの生成を目指すため，以下では STAGES の説明を行う．STAGES のみの生成を目指す理由として，トリックテイキングゲームでは，カードの強さに関しては基本同じなため，RANKING を省略しても問題ないと考え，WINNING POINTS は，基本的にトークンのポイントの価値は同じなため省略しても問題ないと考えたからである．

4.2.1 CGDL のフィールドの構成

CGDL には図 4 のようなあらかじめ決まったフィールドの構成が用意されている．フィールドのそれぞれの盤面には指定するためのを変数が与えられており，変数を記述したフィールドのリストを表 1 に示す．各プレイヤーは一人一つ手札，チップ置き場，賭け金置き場を所有し，HX,KX0,KX1 の X にはそのフィールドを所有しているプレイヤーの番号 X が入る．トークンは主にポーカーのチップなどのゲーム内で賭けるものが表現されており．持ち金を KX0，現在賭けている金額を KX1 に置く．スペースの中でプレイヤーの人数と場の数は可変であるため図 4 はプレイヤー数 3，場の数 2 の時のフィールドの構成である．

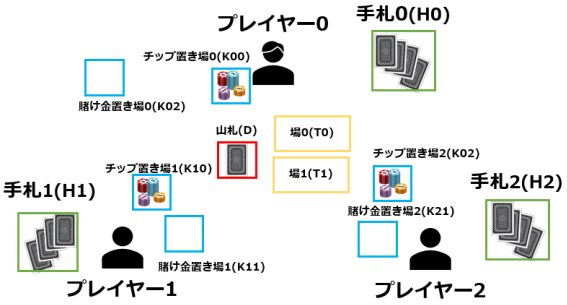

図 4 場の構成 (プレイヤー数 3, 場の数 2 の場合)

フィールド (変数名)	説明
プレイヤー (PX)	X 番目にゲームに参加するプレイヤー
手札 (HX)	X 番目のプレイヤーの手札
トークン置き場 (KX0)	X 番目のプレイヤーがゲームで利用するトークン (チップ) を置く場所
賭け金置き場 (KX1)	X 番目のプレイヤーが現在賭けているトークンを置く場所
山札 (D)	ゲームで使用する山札を置く場所
場 (TX)	X 番目の場，カードを置く場所

表 1 行動の種類のリスト

4.2.2 CGDL の STAGES

カードゲーム UNO の STAGES を Listing1 に示す．UNO の STAGES は，STAGE0 ですべてのプレイヤーにカードを配り場に最初のカードを置くという処理を行い，STAGE1 で同じマークか数字のカードを出し，出せなかったらカードを 1 枚引き，すべての手札がなくなったときに勝利するという構成になっている．

STAGES は複数の STAGE で成り立っており，全ての STAGE が終わったらゲームが終了する．1 つの STAGE は複数の命令で成り立っており，ゲーム中に行う行動をこの命令で表現している．各 STAGE 毎に上から順に命令を行い，全ての STAGE の手順が終わった時にゲームは終了する．記述されている命令は最初のプレイヤーから順番に行い，UNO のカードを出す行動のような順番が回れば何度でもできる命令に関しては，他のプレイヤーの行動が終わり次第再度順番に命令の処理を行う．

Listing 1
UNO

```
1 STAGE0
2 com _ deal, ALLPLAYERS, 7
3 com _ deal, T0, 1
4 STAGE1
5 show, same suit, T0, _ playit
6 show, same number, T0 _ playit
7 draw _ next
8 mandatory _ have, λ _ win
```

4.2.3 CGDL の命令

CGDL の命令は以下の 3 つで構成されている．

- 行動の種類
  いつ行動するのか
- 先行詞
  この条件を満たしたときにアクションを行う
- アクション
  どういう行動をするのか

例えば UNO の STAGES の 8 行目の‘mandatory _ have, λ _ win’は各手番の最初に手札がないプレイヤーは勝利するという手札がなくなったプレイヤーが勝利するという

命令だが図 5 のような構成になっており，各手番の最初にという行動の種類，手札がないプレイヤーという先行詞を満たしたプレイヤーは勝利という行動をするといった形で表現されている．

図 5　命令：手札がなくなったプレイヤーは勝利する

行動の種類で行動のタイミングを決定し，先行詞の条件を満たした場合にアクションを行う．また，UNO のカードを出す命令のような何回でも行ってよい命令の場合は，行動の種類がない．
行動の種類，先行詞，アクションの命令のリストをそれぞれ表 2,3,4 に示す．命令によっては後ろにスペースや＞，＜，同じマーク，同じ番号を要求することもある．

行動の種類	効果
once(1 度きり)	STAGE 中 1 回のみ行える命令
com(computer)	STAGE の開始時に行われる行動
mandatory(必須)	STAGE の開始時に必ず行う命令

表 2　行動の種類のリスト

4.3　CGDL の命令の追加

トリックテイキングゲームを表記する際に既存の命令だけでは表現しきれない命令がある為，表 5 のコマンドを追加する．行動の種類‘else’は直前の命令を実行できなかった際に以下の命令を行うというコマンドで，2.2 のマストフォローのような状況で切り札と同じマークのカードが出せない時に他のカードを出すという命令を表現するために使用されている．アクション‘loop’は指定した STAGE から現在までを指定した回数分繰り返す命令である．トリックテイキングゲームは 5 回トリックを行うなど決められた回数トリックを繰り返す必要があるが，その回数を既存の言語使用では表現できないため追加した．

5.　システム構成

本手法の構成図を図 6 に示す．CGDL を用いてルールを記述し，記述したルールを一定のルールに従ってデータセット化，生成したデータセットを DataAugmentation を行いデータ数を増やし，LSTM モデルで学習を行う．学習後学習したモデルを使用して，トリックテイキングゲームの自動生成を行う．

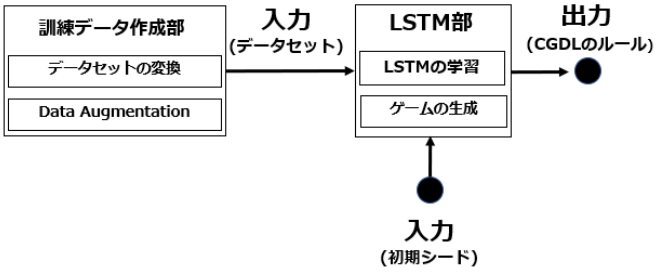

図 6　システム構成図

5.1　訓練データ作成部

5.1.1　データセットの変換

ここでは，CGDL で記述したトリックテイキングゲームのルールを，どのような形で学習させたかを述べる．本手法では以下の条件でルールを記述している．

- 文頭に＜ BOS ＞を文末に＜ EOS ＞を付ける
- stage の区切りには＜ ST ＞をつける
- データ中のカンマはスペースに置き換えている．
- 先行詞，アクションに付随する対象は_で繋ぎ 1 つの単語としてまとめている．

5.1.2　Data Augmentation

本論文ではデータセットをを十分な数用意できず，今回のデータ数 40 ではサンプル数が足りないためデータ数を増やすために Easy Data Augmentation(EDA)[9] を使用して，データの拡張を行った．EDA では以下の 4 つの手法を持ってデータ数を増やしている．

- 同義語置換
- ランダムな同義語挿入
- ランダムなワード移動
- ランダムなワード削除

CGDL には同義語が存在しないため，本手法ではランダムなワード移動とランダムなワード削除を使用している．また，EDA では，ワード移動を行う回数を n，ワードを削除する確率を p，としたとき，

$$n = al$$

$$p = a$$

を満たすこととする．l は文章の長さ，a はパラメーターであり，a の値を自分で設定し，確率を決めている．本手法では，論文内で述べられている最適なパラメーターの中で最も訓練データ数が近いパラメーターである．

$$a = 0.05$$

表 3 先行詞のリスト

先行詞	効果	使用例
tokens, KA, 制限, KB	トークンを置く場所 KA と KB のトークンの量を比較する． 制限を満たしている場合，アクションを行う	tokens, K01, ＞, K11
play, LA, 制限, LB	カードを置く場所 LA と LB のカードの強さ (役) を比較する． 制限を満たしている場合，アクションを行う	play, H0, ＜, H1
sum, LA, 制限, LB	カードを置く場所 LA と LB のカードの合計値を比較する． 制限を満たしている場合，アクションを行う	sum, H0,=,H1
have , 組み合わせ	プレイヤーが手札にカード，マーク，特定の数字の組み合わせ (連番など) を 持っているか確認する．持っている場合アクションを行う．	have, 2 of diamonds
draw	デッキからカードを 1 枚引き，それを現在のプレイヤーの手札に加える． 加えたのちアクションを行う．	draw
show, 制限, LA	カードの場所 LA の一番上に置かれているカードと制限を 満たすカードを指定したのちアクションを行う．	show, card with same suit, TO
λ (LAMBDA)	条件なしでアクションを行う	unconditional

表 4 アクションのリスト

アクション	効果	使用例
pifr, LA * 枚数, 向き	カードを置く場所 LA から枚数分指定 した向き (表，裏向き) にカードを引く	pifr, D*2, down
puin, LA * 枚数, 制限, 向き	カードを置く場所 LA に制限を満たす カードを指定した枚数分置く	puin, T0 +1, ＞, up
bet, 制限, KX	トークンを置く場所 KX のトークンの量と比較 したときに制限を満たすトークンの量を賭ける．	bet, ＞, K01
gain , KX	トークンを置く場所 KX のトークンを取得する．	gain, K21
playit	Antecedent で指定 (show) したカードを指定した場所に置く	-
next	プレイヤーの状態を next(行動済み) に変更する． next になったら次の順番が来るまで待機する．	-
done	プレイヤーの状態を done(行動済み) に変更する． done になったらそのステージ中行動できない．	-
out	プレイヤーの状態を out(脱落) に変更する． out になったらゲームから脱落する．	-
win	プレイヤーが勝利し，ゲームが終了する．	-
end	ゲームを終了する．	-

表 5 追加する命令

種類	命令名	説明
行動の種類	else	直前の命令が実行できない 場合以下の行動を行う
アクション	loop, STAGEX, Y	STAGEX からここまで を Y 回繰り返す

$$n_{aug} = 16$$

の Data Augmentation を行っている．

5.2 LSTM 部

本手法では，[8] に記されているソースコードを利用し実装を行う．ソースコードは github に公開されている [10]．パラメーターは，epoch を 1000，batch_size を 32，temperature を 0.2 に設定している．temperature は値が小さくなればなるほど決定論的なサンプリングを行うため，CGDL というプログラムに近い性質をもつ言語には低い値の方がより精度の高い出力が出ると考え，0.2 に設定した．

5.3 生成結果

以下に生成した CGDL のゲームの一例を記す．このゲームではまず STAGE0 ですべてのプレイヤーにカードを 5 枚配り，T1 にカードを 1 枚置いている．STAGE1 同じマー

クのカードがあったら何回でも置くことができ，一度だけ順番をスキップすることができる．STAGE2 では，最も強い手札を持っているプレイヤーが賭けたトークンを得ている．STAGE3 では最もトークンを持っているプレイヤーが勝利し，その後すべてのプレイヤーにカードを 1 枚配ってこのゲームは終わる．

```
1  STAGE0
2  com _ deal, ALLPLAYERS, 5
3  com _ deal, T1, 1
4  STAGE1
5  show, same suit, T1 _ playit
6  once _ unconditional _ next
7  STAGE2
8  mandatory _ play, HX, >, HA _ gain,KX
9  STAGE3
10 mandatory _ tokens, KX0, >, KA0 _ win
11 com _ deal, allplayers, 1
```

5.4 考察

9 行目でトークンを賭けていないにもかかわらず gain が出てしまっている点を除けば行動の種類の後には先行詞，先行詞の後にはアクションといった文法的な部分に問題は生じておらずプログラムとしては成立している．一方で，勝者が決まった後に最後にカードを配る，手札が強さで勝利が決まるゲームながら何回でもカードを捨てられるルールなど，ゲーム性としては不可解な部分も存在している．これは，訓練データが足りないため，前後の単語の関係性は学習できても，単語レベルの関係性を学習しきれていなかったのではないかと考える．今後はデータを増やして，さらに高い精度を目指す．

6. 今後の課題

本手法では CGDL を用いたカードゲームの自動生成を試みた．結果，完全にルール的に正しいゲームを作ることはできなかった．以下に精度を上げるための今後の課題を示す．

- データ数の増加
  今のデータ数では，ルールに不備が生じる点や，様々なルールを表現するのが難しいため，トランプゲーム大全全てのゲームをデータセットにしたい．
- 単語の細分化
  本手法では，各命令の対象を命令に含み学習を行ったが，今後は対象を分割してゲームを生成できるようにしたい．
- 生成したゲームの評価
  ルールとして問題がなくなるまで精度が上がった際に作成したゲームを遊んでもらい評価を行いたい．

参考文献

[1] 有田 隆也 (2011) 「ドイツボードゲームの教育利用の試み—考える喜びを知り生きる力に結びつける—」, コンピュータ＆エデュケーション 31(0), 34-39, 2011
[2] 松本太一 (2018) アナログゲーム療育ーコミュニケーション力を育てるー (幼児期から学齢期まで) ぶどう社
[3] 日本にボードゲームが広まらない理由”https://tgiw.info/2011/08/post_1064.html”
[4] Jose M. Font , Tobias Mahlmann, Daniel Manrique1, Julian Togelius(2013) ”A Card Game Description Language”
[5] 赤桐 裕二 (2014) トランプゲーム大全　スモール出版
[6] Vincent Hom, Joe Marks . ”Automatic design of balanced board games.” Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital EntertainmentJune 2007 Pages 25–30
[7] Hochreiter, Sepp, and Jürgen Schmidhuber. ”Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.
16th European conference on Applications of Evolutionary Computation
[8] David Foster(2020) 生成 Deep Learning　——絵を描き，物語や音楽を作り，ゲームをプレイする　オライリージャパン
[9] Jason W. Wei, Kai Zou(2019) ”EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks” EMNLP-IJCNLP 2019 short paper
[10] github GDL Code
”https://github.com/davidADSP/GDL_code/blob/master/06_01_lstm_text_train.ipynb”