California University of Pennsylvania

CSC 490: Senior Project

Project Vulpix

Requirements Document

Andrew Siddall

Adlene Bellaoucha

Mathew Bedillion

Christopher Crisson

October 17, 2018

Instructor Comments/Evaluation

Table of Contents

Abstract

The Pokemon Trading Card Game (PTCG) has gained in popularity as of late, with many new players discovering the game. Most of these new players want to try out the competitive aspects of the game without the high investments. In this paper we introduce a highly competitive Artificial Intelligence to help players improve their skills. Project Vulpix will use Monte Carlo search trees to teach to learn how to play the PTCG competitively. We will extensively train Project Vulpix using self-play and competition against skilled human players. Project Vulpix will develop its own strategies to accomplish its tasks. We will start with training Project Vulpix then test it's might against the simple AI supplied by the PTCGO client, then escalate it to human competition. Project Vulpix will be able to compete with skilled human player and help aid new players into joining the competitive world of the Pokemon Trading Card Game.

Introduction

Overview of Project

Project Vulpix came to mind while observing how the competitive scene of the Pokemon Trading Card Game (PTCG) is designed. The way the professional game is structured is detrimental to people in rural areas or people who do not have the means to travel. The larger, and thus more competitive, tournaments and events are only in a few select cities in the United States. If they do not live close to one of these events or have a talented group of players to practice against, then they will be at a disadvantage in the competitive scene. This is where Project Vulpix comes in. The goal of Project Vulpix is to create an Artificial Intelligence(A.I.) that can play PTCG at a competitive level. The only way to play the PTCG against a computer opponent currently is using the online client's tutorial. The computer in the tutorial has three different levels of competitiveness. The first is incredibly simple for those just starting out. The second and third are a bit more challenging, but ultimately nowhere near the level of a competitive game. When playing against the client, it almost seems like the decisions the AI takes are set up to help the player, rather than beat the player. If the player only used this method to train, they would not be prepared for a competitive event. Project Vulpix will use complex decision trees and an AI trained at being highly competitive.

Objective of Project

The object of Project Vulpix is to create an AI that can help players improve their skills at the PTCG. We want all players to be able to test their skills and strategies against a competitive player, without having to leave their house or spend hundreds of dollars. Project Vulpix will be able to make decisions about the game state as a competitive player would. We will train it extensively to determine optimal plays and develop winning strategies of its own.

Team Details

Our group is composed of four Computer Science majors. Our culture is that we must rely on each other and work well as a team, following a solid plan with plenty of communication throughout the whole period. From our background, at least one member of the group is already familiar with every feature/tool that is going to be used for the project such as Python, Databases. As far as the workflow leader, Andrew Siddall is the closest member to leading the group since Vulpix was initially his idea. He will be assigning multiple tasks to different team members and setting up deadlines for every task/duty that members are expected to complete as a function of their roles. He will also take lead in implementation. Chris Crisson will contribute in identifying training needs and mapping out development plans for team members. He is going to make sure that things are always in the right directions and will help the team to accelerate the workflow tempo by developing our skills (especially in Python) per project requirements. Adlene Bellaoucha will propose effective design solutions to meet project goals. He will be involved in aiding in project design and development activities of processes and functions that help users interact with the system. Those designs will include documentations, technical specifications and other project related documentations as needed. Matthew Bedellion will be participating by designing testing scenarios for usability testing. He will make sure how testing should be carried out in each phase. In other words, he will be assessing code and assuring the quality of software development and deployment. All team members are responsible for constant proofreading and correcting/editing whether Grammarly or on the technical level. As a team, we are fully committed to the success of the project in a timely manner.

Application Domain

Project Context

Project Vulpix will be appropriate for Pokemon TCG players who are at an intermediate level of competence with the game. This is because the software requires a lot of user input. A player with little experience certainly could use the software, but it will be most effective when a more experienced player uses it.

The user will begin the software and be provided an interface to enter the contents of each players deck. Sample decks will be provided to as well as an option to save an entered deck to the samples. This will allow the user to replay the same decks again. The user will then have to enter and/or alter the rest of the game state. This includes cards in hand, cards in play area for both players, number of cards in opponent’s hand, and cards in discard pile for each player.

Once the user confirms the information the software will process the input and provide the user with output in the form of a game action. Due to the nature of algorithm used the user will be able to stop the process to return the current best game action as the run time can be very long. The player then alters the game state again, following directions provided in the output. This is repeated until the user decides to quit or the software determines the game state is a win state or lose state.

Initial Business Model

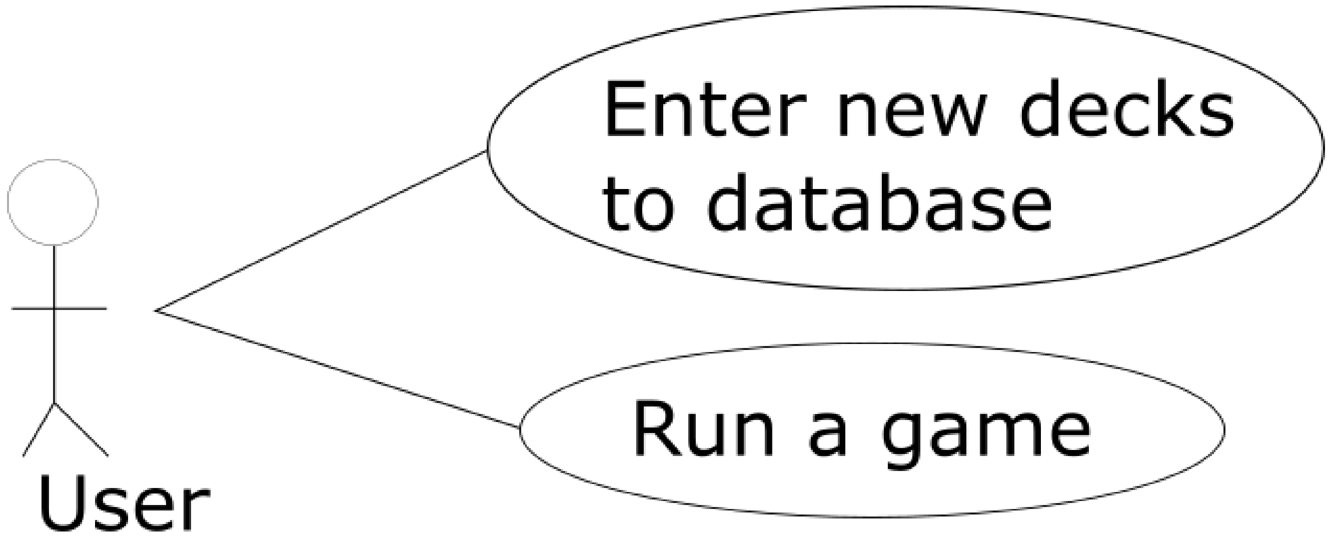
Operational Environment

The software will be written in Python 3 using multiple required libraries. The user will need to have these installed to run the script. Python 3 is supported on Microsoft Windows, Mac OS, and Linux. While system requirements are not known yet, the software is expected to need a relatively fast processor to run in a reasonable amount of time.

Description of Data Sources

The program will require input from the user about the game state. The software will analyze the input and provide output to the user as a game action. It will then wait for an updated game state to analyze. This will continue until the user stops entering data.

Use Case



User will enter new decks into Project Vulpix and can run Project Vulpix.

Initial Requirements

Functional

Project Vulpix will provide the user with a simple user interface. It will consist of many text fields, each accepting either a list of card names for game information known to the user, or a number representing the number of cards in the game zone for cards the face of which is hidden from the user. Once this data is received it will begin to determine the best game action for the player. It will do this by searching a tree. The tree will be created using a modified Monte

Carlo Tree Search. The traditional Monte-Carlo tree search has become the standard for general game playing (Monte Carlo Tree Search – About- web).

The user will be able to choose from a list of decks for the AI to use. The decks will each have their own strategies. When the user selects a deck, we will display the contents of that deck. The user will be able to edit the currently selected deck to add or change the cards in the deck. The user will be prompted with a list of cards that can replace the current cards in the deck, and the current contents of the deck. When a user selects a card to add to the deck, the program will ask the user what card it is replacing and validate that card to make sure if fits within the rules of the game. If the card passes validation, then the card is replaced/added to the deck.

The software will run on any Window's PC that has python 3. It should run on any machine with Python 3 installed, but it will be designed for Windows users. The user will need to have an account with the PTCGO client to use our AI effectively. The user could choose to use the software to play with a physical deck.

Nonfunctional

A Monte Carlo search tree is a way to teach a computer how to play an imperfect game, a game that has some information obscured from the player. In MCTS each node of the tree is a game state. MCTS uses a four-step approach to help solve the problem of playing an imperfect game with an AI. The four steps are Selection, Expansion, Simulation, and backpropagation. The MCTS will repeat the four steps multiple times, this is called sampling or playout. Sampling allows the AI to go down may different paths of the tree and get estimated win/loss ratios for each path. The more samples you do the more paths of the tree is gone down. As the AI goes down more paths it gets better estimates of which paths are good. In the selection step the algorithm starts with the root and selects child nodes until there are no more child nodes in that path. We then add a node to the last node that we explored, this is the expansion step. In the simulation step we randomly select moves to advance the game state until a winner, either the player or his opponent, is found. We then update all the nodes that have led to this node and back propagate the win/loss ratios to all the nodes above us. One of the drawbacks of MCTS is that it can be slow, we intend to use the Upper Confidence Bound 1(UCB1) algorithm to improve efficiency. UCB1 is a way to balance exploration, how many new nodes we test, and exploitation, choosing an action we think is optimal. We use it to guess what action will give us the best reward, winning the game.

Appendix: Technical Glossary

Ability: An Ability is an effect on a Pokémon that is not an attack. Some will be active all of the time, while some you will need to choose to use. Read each Ability to make sure you understand exactly how and when it works.

Active Pokémon: The player's in-play Pokémon that is not on the Bench. Only the Active Pokémon can attack.

Attach: When you take a card from your hand and put it on one of your Pokémon in play.

Attack: 1) When your Active Pokémon fights your opponent's Pokémon. 2) The text written on each Pokémon card that shows what it does when it attacks (a Pokémon can have several attacks on it).

Attacking Pokémon: The Active Pokémon, as it performs an attack.

Basic Energy Card: A Grass, Fire, Water, Lightning, Psychic, Fighting, Darkness, or Metal Energy card.

Basic Pokémon Card: A card you can play directly from your hand on your turn. *See Evolution card.*

Bench: The place for your Pokémon that are in play but are not actively fighting. They come out and fight if the Active Pokémon retreats or is Knocked Out. When Benched Pokémon take damage, do not apply Weakness or Resistance.

Burn marker: What you put on a Pokémon to remind you it is Burned. Remove the marker if the Pokémon is Benched or Evolved. *See counter, damage counter.*

Counter: Object put on a Pokémon as a reminder (for example, a Char counter). A counter does not go away when you Bench the Pokémon, but it does go away if the Pokémon evolves (damage counters are a special exception to this rule). *See damage counter, Poison marker, Burn marker.*

Damage: What usually happens when one Pokémon attacks another. If a Pokémon has total damage greater than or equal to its Hit Points, it is Knocked Out.

Damage Counter: A counter put on your Pokémon to show it has taken damage. It stays on your Pokémon even if the Pokémon is Benched or evolved. Each damage counter counts as much damage as it has printed on it. *See counter, Poison marker.*

Defending Pokémon: Your Active Pokémon when your opponent is attacking.

Devolve: Certain cards can devolve an Evolved Pokémon, which is the opposite of evolving your Pokémon. When a Pokémon is devolved, it also loses Special Conditions and any other effects.

Discard Pile: The cards you have discarded. These cards are always face up. Anyone can look at these cards at any time.

Energy Card: Cards that power your Pokémon so they can attack. *See basic Energy card.*

Evolution Card: A card you play on top of a Basic Pokémon card (or on top of another Evolution card) to make it stronger.

Fossil Trainer cards: A special kind of Trainer card that acts like a Basic Pokémon when put into play. When a Fossil Trainer card is in your hand, deck, or discard pile, it is not considered a Basic Pokémon. However, these Trainer cards always count as Basic Pokémon during set-up.

Hit Points (HP): A number every Pokémon has, telling you how much damage it can take before it is Knocked Out.

In-Between Turns: The part of each turn when the game shifts from one player to the other. Check Poison, Burn, Asleep, and Paralysis at this step, and see whether any Pokémon are Knocked Out.

In Play: Your cards are in play when they are on the table. Basic Pokémon cards, Evolution cards, and Energy cards cannot be used unless they are in play. (The cards in your deck, your discard pile, and your Prizes are not in play, but your Benched Pokémon are.)

Knocked Out: A Pokémon is Knocked Out if it has damage greater than or equal to its Hit Points. That Pokémon goes to the discard pile along with all cards attached to it. When one of your opponent's Pokémon is Knocked Out, take one of your Prizes.

Item card: A type of Trainer card. Follow the instructions on the card and then discard it.

Lost Zone: Cards sent to the Lost Zone are no longer playable during that match. Put them face up anywhere out of play.

Owner: A Pokémon with a Trainer's name in its title, such as Brock's Sandshrew or Team Rocket's Meowth.

Poison Marker: Object put on a Pokémon to remind you it is Poisoned. Remove the marker if the Pokémon is Benched or evolved. *See counter, damage counter.*

Poké-Body: An effect that is active as soon as that Pokémon card is in play and lasts until the Pokémon leaves play.

Poké-Power: A once-per-turn power on Active and Benched Pokémon you must choose to use. Most Poké-Powers are turned off if the Pokémon has a Special Condition.

Pokémon: The colorful characters that fight for you in the Pokémon Trading Card Game. They are represented in the game by Basic Pokémon and Evolution cards.

Pokémon-ex: Pokémon-ex are a stronger form of Pokémon with a special drawback: when your Pokémon-ex is Knocked Out, your opponent draws two Prize cards instead of one.

Pokémon LEGEND: Special double cards that showcase powerful Legendary Pokémon. Both cards must be played together at the same time.

Pokémon LV.X: Stronger versions of a regular Pokémon, put on top of the regular Pokémon of the same name and adding extra abilities to the original Pokémon.

Pokémon Power: A special ability some Pokémon have. Pokémon Powers are divided into two categories: Poké-Power and Poké-Body. They always include the words "Poké-Power" or "Poké-Body" so you can tell they are not attacks.

Pokémon SP: A special Pokémon trained by a particular Trainer, with a symbol in its name to show its owner.

Pokémon Tool: A special kind of Trainer card (an Item) you can attach to your Pokémon to help you. Each Pokémon can have only 1 Pokémon Tool attached at any time.

Prize Cards: The 6 cards you put face down at the start of the game. Every time one of your opponent's Pokémon is Knocked Out, you take 1 of your Prizes into your hand (or 2 Prizes, for a Pokémon-ex). When you take your last Prize card, you win!

Resistance: A Pokémon with Resistance takes less damage when attacked by Pokémon of a certain type. The amount of Resistance is printed next to the type of Resistance(s) a Pokémon has, if any.

Retreat: When you switch your Active Pokémon with one of your Benched Pokémon. To retreat, you must discard Energy from the retreating Pokémon equal to the Retreat Cost of the Pokémon. This cost appears in the lower right-hand corner of the card. You can only retreat once per turn.

Special Conditions: Asleep, Burned, Confused, Paralyzed, and Poisoned are called Special Conditions.

Stadium Card: A type of Trainer card similar to an Item card, but stays in play after you play it. Only one Stadium card can be in play at a time—if a new one comes into play, discard the old one and end its effects. You can play only one Stadium card each turn.

Sudden Death: Sometimes both players win at the same time. In this case, you play a short game called “Sudden Death” (use only 1 Prize card each instead of 6).

Supporter Card: A Trainer card similar to an Item card. You can play only one Supporter card each turn.

Technical Machine: A kind of Trainer card (an Item) you can attach to your Pokémon. When attached, your Pokémon can use the Technical Machine attack as its own. Technical Machine cards remain attached unless the card text says otherwise.

Trainer Card: Special cards you play to gain advantages in the game. *See Item card, Stadium card, Supporter card.*

Trainers’ Pokémon: Pokémon with Trainers’ names in their titles, like Brock’s Sandshrew. You cannot evolve a regular Sandshrew into Brock’s Sandslash, and you cannot evolve a Brock’s Sandshrew into a regular Sandslash. This is because “Brock’s” is part of the name.

Weakness: A Pokémon with Weakness takes more damage when attacked by Pokémon of a certain type. The effect of the Weakness is indicated next to the type(s) of Weakness a Pokémon has, if any.

Appendix II: Team Details

Andrew Siddall served as the workflow leader for this portion of the project. Andrew understood the goal and did extensive research on the topic. Adlene, Chris, and Matt all contributed to this document as well. We all worked together by adding content, editing each other's work, and just bouncing ideas off each other. Because of this there was no one place that only one of us contributed. We all worked cooperatively and did our fair share of the work.

Appendix III: Workflow Authentication

I, Andrew Siddall, hereby declare and confirm that I have performed the work as documented herein.

Signature: ____

Date: ___________________

I, Adlene Bellaoucha, hereby declare and confirm that I have performed the work as documented herein.

Signature: ____

Date: ___________________

I, Mathew Bedillion, hereby declare and confirm that I have performed the work as documented herein.

Signature: ____

Date: ___________________

I, Christopher Crisson, hereby declare and confirm that I have performed the work as documented herein.

Signature: ____

Date: ___________________

Citations

Monte Carlo Tree Search - About, mcts.ai/about/index.html.

j2kun. “Optimism in the Face of Uncertainty: the UCB1 Algorithm.” Math ∩ Programming, 9 May 2014, jeremykun.com/2013/10/28/optimism-in-the-face-of-uncertainty-the-ucb1-algorithm/.

Kordík, Pavel. “Reinforcement Learning: Artificial Intelligence in Game Playing.” Medium, Medium, 17 Nov. 2016, medium.com/@pavelkordik/reinforcement-learning-the-hardest-part-of-machine-learning-b667a22995ca.