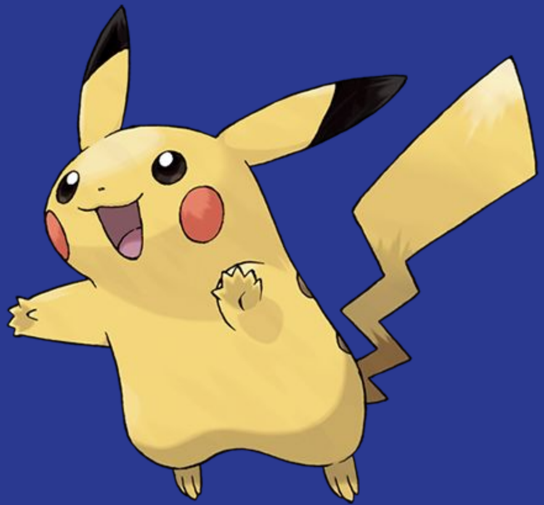
Using Supervised Learning to Win at Pokemon.

Bobby King
Jessica Gettings
Xuewen “May” Yao
Zeling “Angie” Zhang



Proposal Review

- Figure out which team of 3 pokemon yields the highest chance of winning a battle, assuming all Pokemon are at the same level
- Determine each Pokemon’s ideal set of 4 moves
- Determine which move to use in a given situation and its confidence in leading the team to a win

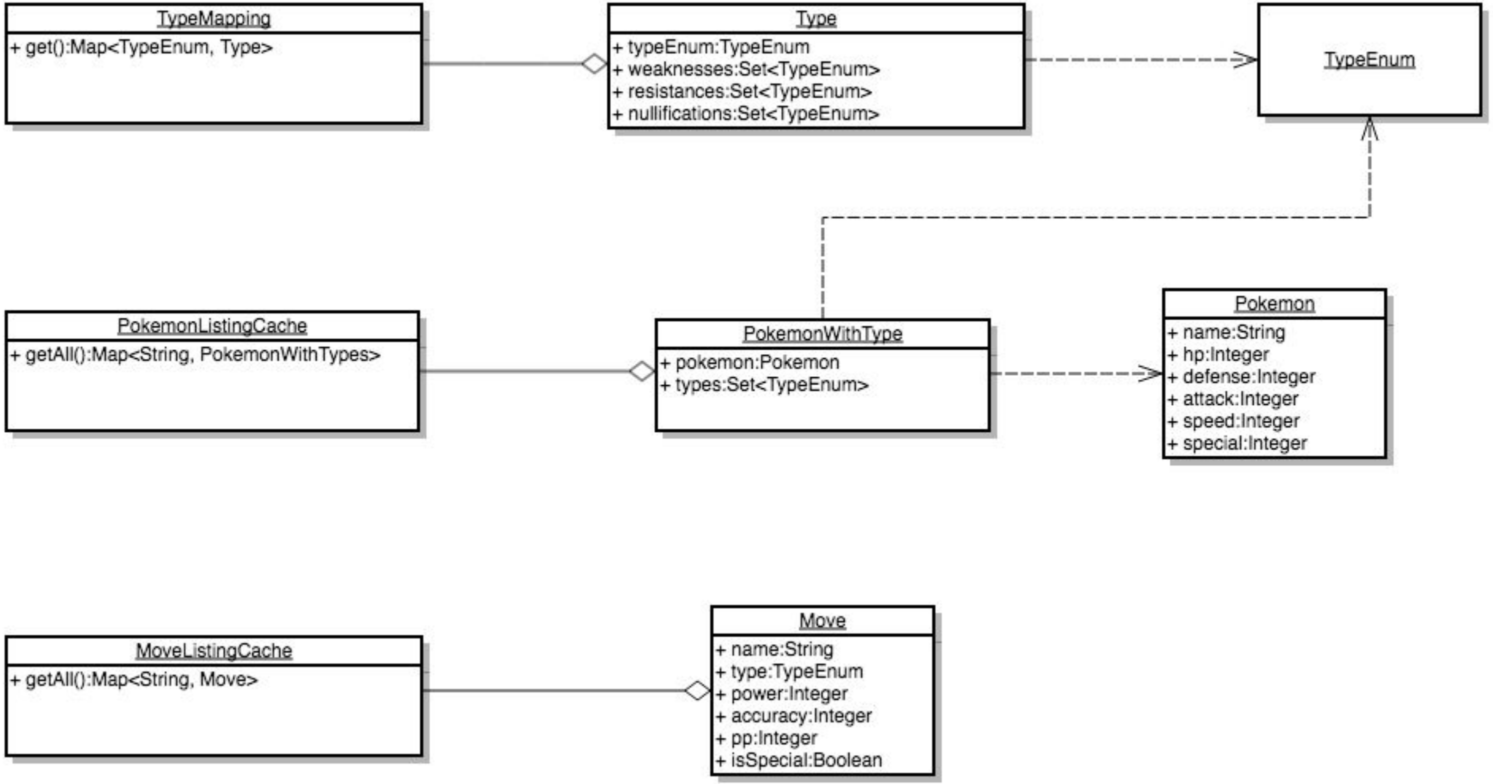
Overview of Presentation

- How we simulated and generated battle data (Bobby)
- How we determined which Pokemon led to best results (Jessica)
- How we determined which 4 moves a Pokemon should have at its disposal (Angie)
- How we determined which move a Pokemon should use in a specific situation (May)

Battle Simulation and Generation

- First, determined which teams of 3 we wanted to analyze
  - Used permutations instead of combinations, since which pokemon is played first does affect the outcome of a battle
  - For all 151 pokemon, there's over 3 million different team permutations
  - So we narrowed it down to the top 50 pokemon by average stats
    - Now only 117600 team permutations to analyze.
- Round robin would be over 1 billion battle simulations
  - Would've taken many days to simulate
  - Instead, we did 5 random battles per team as team 1.
    - Reduces the number of battles to analyze to ~1 million.

UML Diagram - Caching Information



Battle Simulation - Object Design

- BattleTree
  - Contains States and Edges, in the form of a Directed Tree Graph
- State
  - Contains the current state of every pokemon on every team, and whether we are at an initial state or possibly at an end state.
- Edge
  - Contains what moves each pokemon did to cause the transition from the prevState to the nextState.

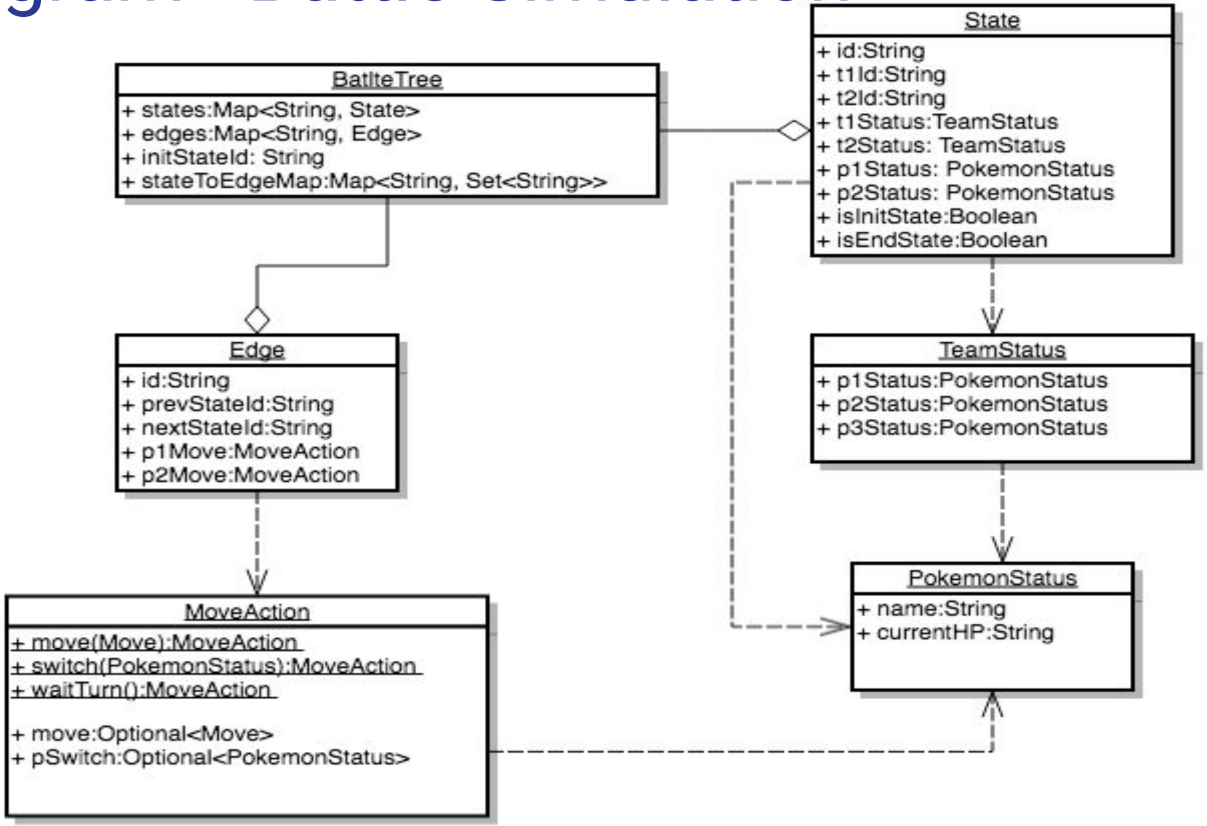
Battle Simulation - Strategy

- Looking at all possible paths from the initial state to the end states would yield a battle graph that is too large to generate and analyze.
  - Assuming a pokemon can use any one of 4 moves and an average depth of 10, we would have 16^10 possible states.
- So instead, we used a simple strategy to yield the best possible results:
  - If a pokemon can do a move that yields enough damage deemed “worth it”, it’ll do that move.
  - If a pokemon cannot do a move that meets that threshold, we instead switch to a pokemon that can take the least amount of damage from what move the opponent would’ve picked assuming they follow the first step.
  - This yields a linear battle graph of ~12-14 states deep.

Special Move cases

- Some moves require special rules that deviate from typical workflow
  - First-Turn 2-turn moves - Hyper Beam
    - Pokemon must recharge and cannot be switched for 1 turn after using this move
  - Second-Turn 2-turn moves - Dig, Fly, Sky Attack
    - Pokemon spends the first turn preparing, thus cannot be switched in the next turn and must use the move regardless of pokemon switch from opponent.
  - Ignored special cases
    - Dream Eater requires the opponent to be sleeping, so we just ignore this move.
    - Earthquake deals double damage if the opponent is preparing for dig, we're ignoring this.
    - Thunder will always hit if opponent is preparing for fly, we ignore accuracy in general for the sake of simulation.
    - Self Destruct and Explosion kill the pokemon using those moves, so we ignore them.

UML Diagram - Battle Simulation



Generated Data Result

- We have ~1 million simulated battles, stored in many json files.
- “teams.json” contains a JSON object representing a key/value pair, id to team.
- “battles-id.json” is a list of simulated battles (1 file per team for indexing purposes).
- These can be read in using FasterXML library for java that can read in json files and streams into POJOs.

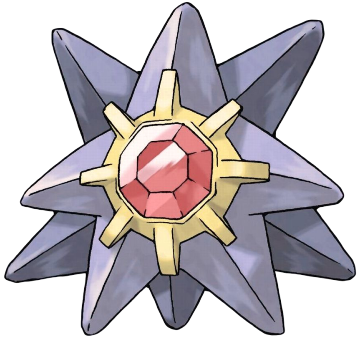
Pokemon Analysis

- We want to determine the best team of 3 Pokemon
- Generated data contains only 5 battles per team
  - This takes permutations into account, but even if it didn't, we would still only have 30 battles per team
  - Not enough to differentiate "good" from "best"
- Our Solution: *a priori* algorithm
  - Identify the individual Pokemon that most frequently appear on winning teams
  - Combine the top Pokemon into pairs and repeat the analysis to find the best-performing pairs
  - Select the team that contains the three best pairs of Pokemon

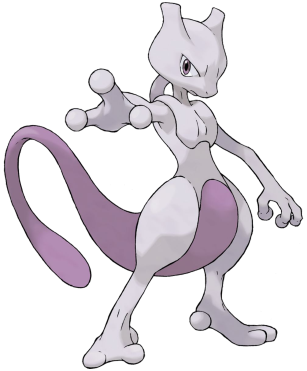
Choosing the Best Pokemon

- Each input file contains the the outcomes of all battles in which a given permutation of 3 is designated “Team 1”
- An index of Pokemon to teams was created for easy reference
- For every input file:
  - Determine which Pokemon belong to the team
  - Map battle output data (number of wins, total number of battles) to a running tally for each Pokemon on the team
  - wins/battles ratio = confidence
- Because Pokemon are equally distributed across teams:
  - Support is the same for every team
- All told, each Pokemon belongs to 7,056 teams and participates in ~35,000 battles

Preliminary Top Three



Starmie
Water/Psychic
Confidence: .7872



Mewtwo
Psychic
Confidence: .663

Lapras
Water/Ice
Confidence: .7225

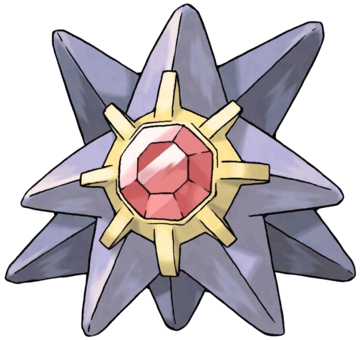
Move Analysis

- Determine the best 4 moves for the top listed Pokemon
- This is done by determining the most frequent moves for each Pokemon in all the battles they took part in.

Move Analysis - cont'd

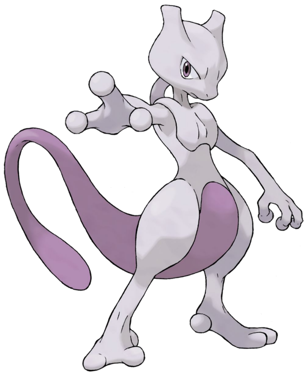
- Looked at each battle state and determined which move was used along its corresponding edge
  - Since each move made is the optimal move, we just count up how often each Pokemon used each move.
- Chose the 4 most frequent moves as that Pokemon's moveset

Move Analysis - Some Results



Starmie
Hydro Pump - Water
Thunder - Electric
Psychic - Psychic
Blizzard - Ice

Lapras
Hydro Pump - Water
Thunder - Electric
Blizzard - Ice
Waterfall - Water



Mewtwo
Psychic - Psychic
Earthquake - Ground
Blizzard - Ice
Hyper Beam - Normal

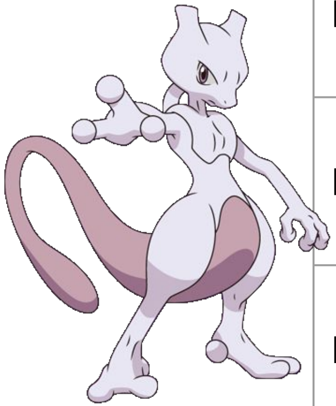
Situation Analysis

- Given a Pokemon doing battle and its current opponent, which move would yield the best results? What's the confidence that the move leads to a win for that Pokemon's team?
  - Either the move that deals the most damage against its opponent, or a switch
  - Want to find out the effects of the current move on the end result of the battle

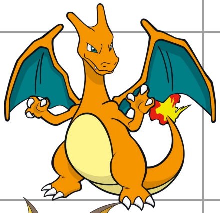
Situation Analysis

- There are 50 Pokemon in total
- Not counting “mirror matches,” (Pokemon fighting themselves), there are 2450 pairs of attacker and defender in total
- For each attacker-defender pair, find all their battles
- For each battle, find the move the attacker uses, walk down the tree, and obtain the result of the whole game
- Calculate the confidence of the move--how often this move leads to a win against the defender

Situation Analysis - Results



Attacker	Defender	Move	Confidence
Mewtwo	Charizard	Rock Slide	0.798153
Mewtwo	Raichu	Earthquake	0.998177
Mewtwo	Arcanine	Earthquake	0.796018



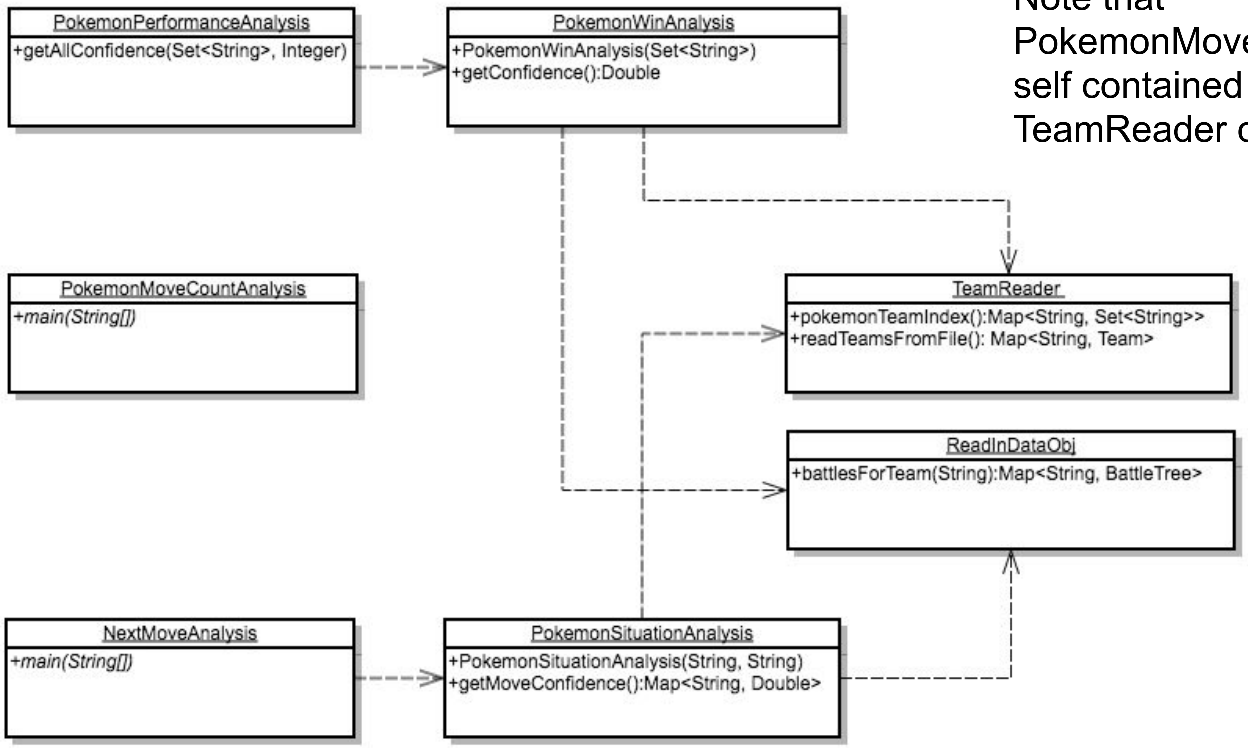
Next Steps for future work on this

- Phase 2 of Data Generation
  - Come up with other strategies to compare against our current strategy.
  - See if we can have more branching paths if we move this to a computing cluster for analysis.
- Phase 2 of Pokemon Analysis:
  - Select teams that consist only of top-performing pairs of Pokemon, and generate more battles for those teams in order to confirm which one is the best
- Phase 2 of Pokemon next move analysis
  - Create a way to update move analysis with more simulated battles without restarting the algorithm.

Future Work needed for modern day Pokemon

- There would be a few modifications needed to adapt our algorithms to the most recent versions of the game
  - New Pokemon with new abilities would need to be hard-coded in
  - Update the damage dealing algorithm to account for changes in engine

System Architecture

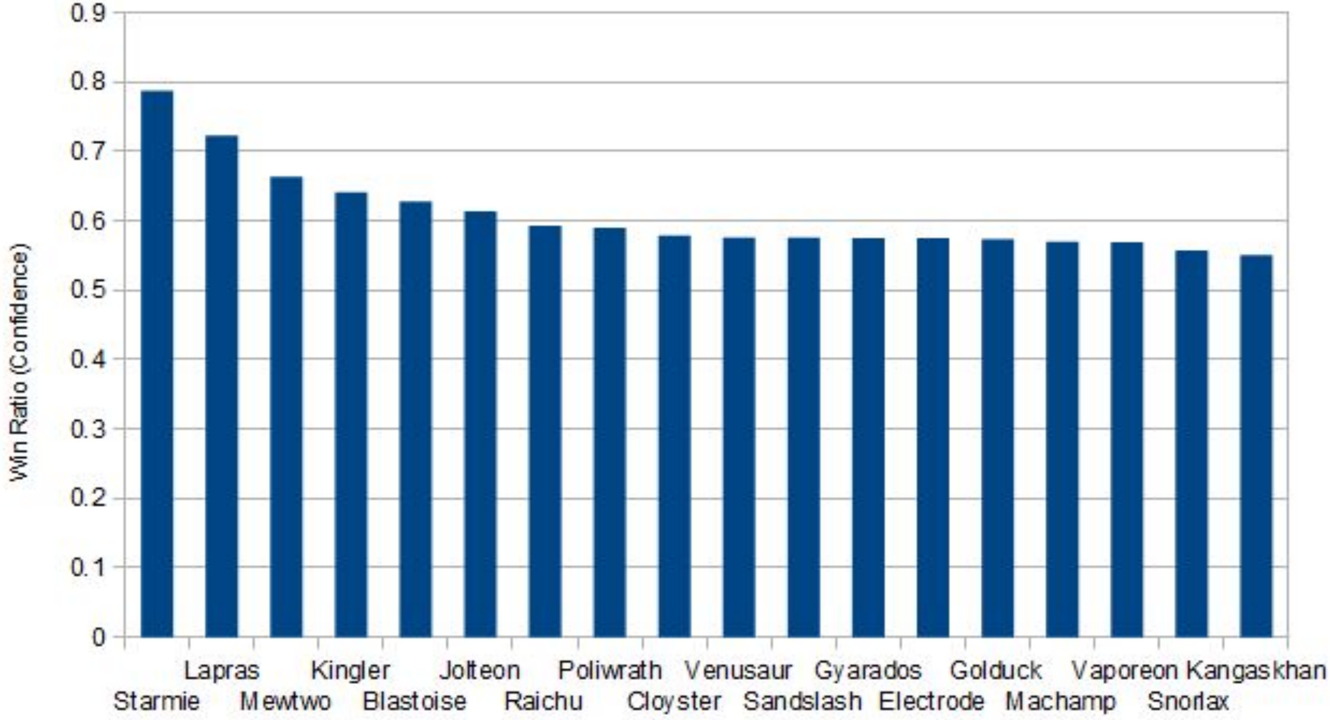


Note that PokemonMoveCountAnalysis is self contained and doesn't use TeamReader or ReadInDataObj.

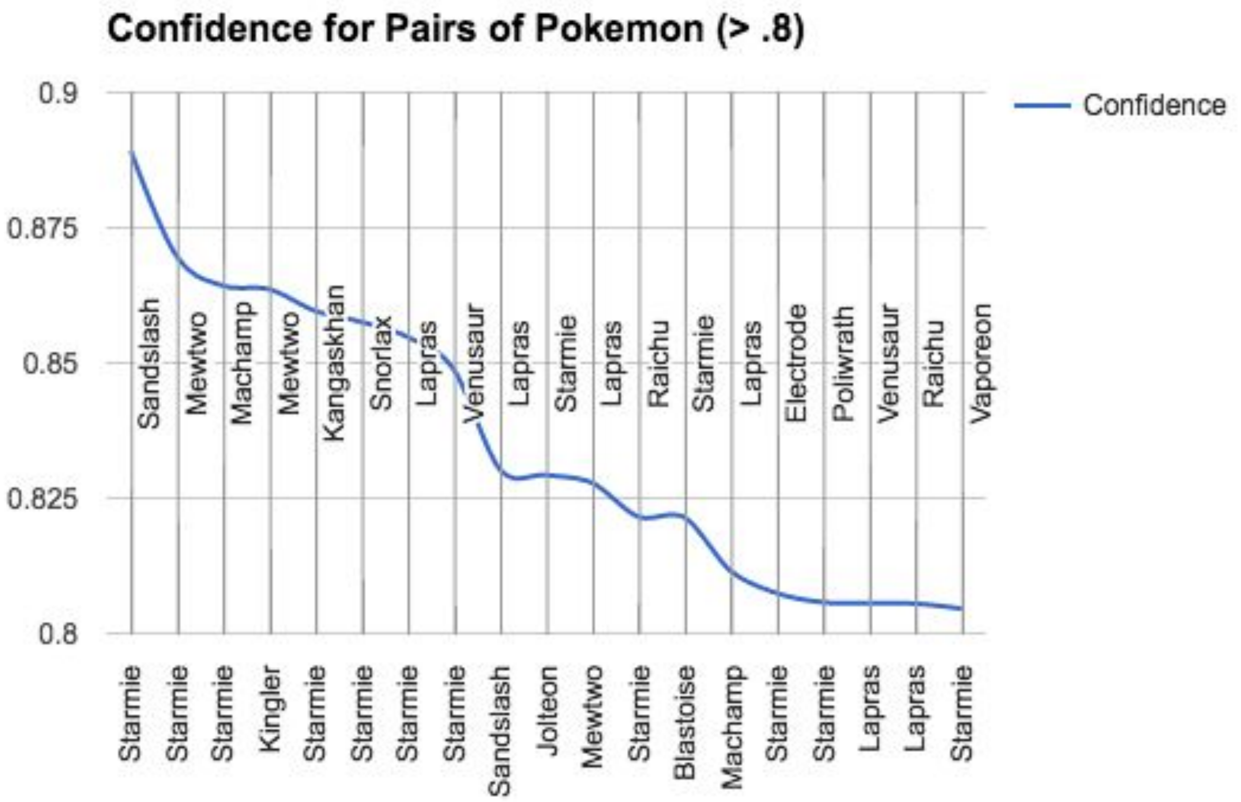
Descriptions of Main Functional Components

- Pokemon, Team, Move, and BattleTree objects are used to simulate battles and generate data for each battle
- Additional Analysis classes are used to analyze the data
    - Pokemon Analysis (find the Pokemon with the best track records)
    - Move Analysis (find the moves with the best track records)
    - Situation Analysis (find the best move in context)

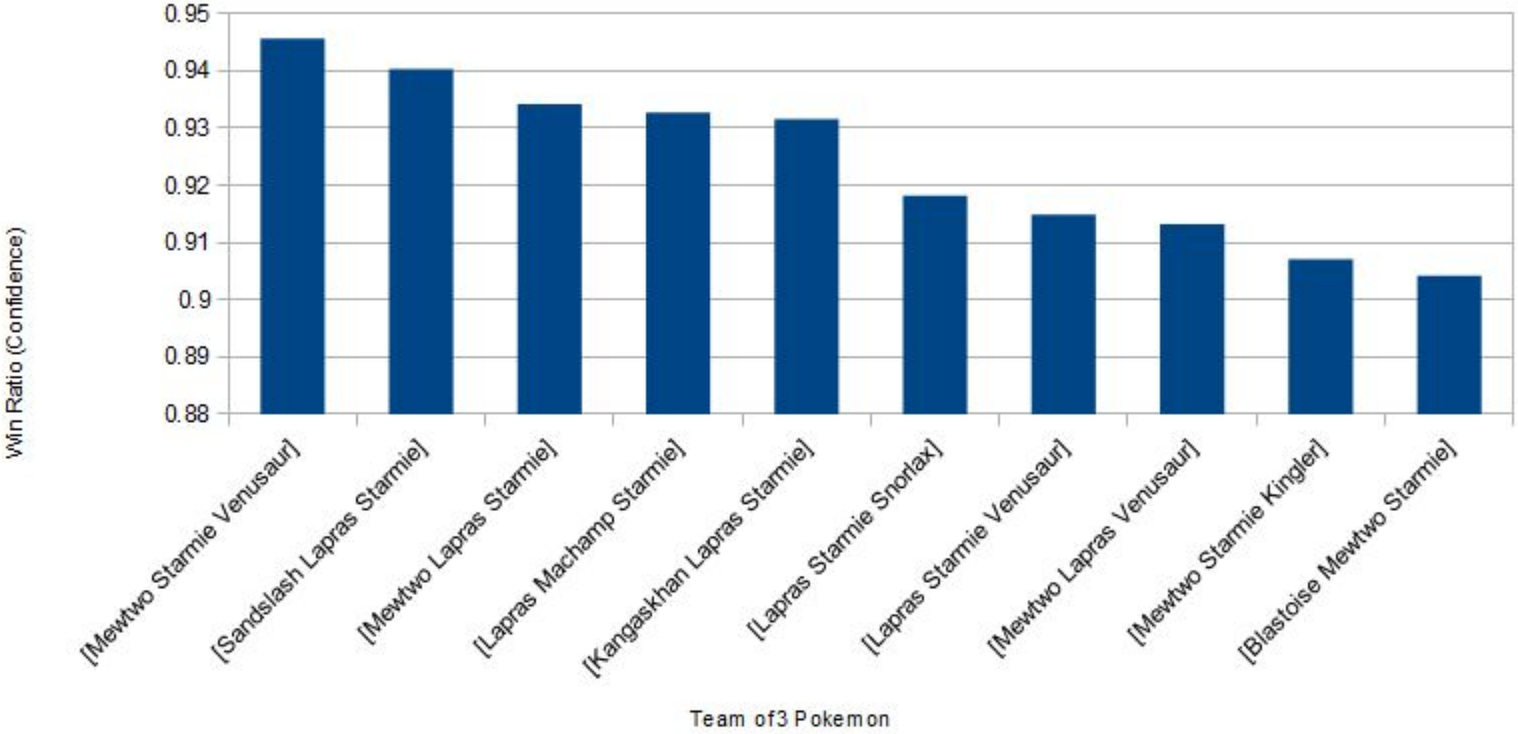
Experimental Results #1: Top Pokemon



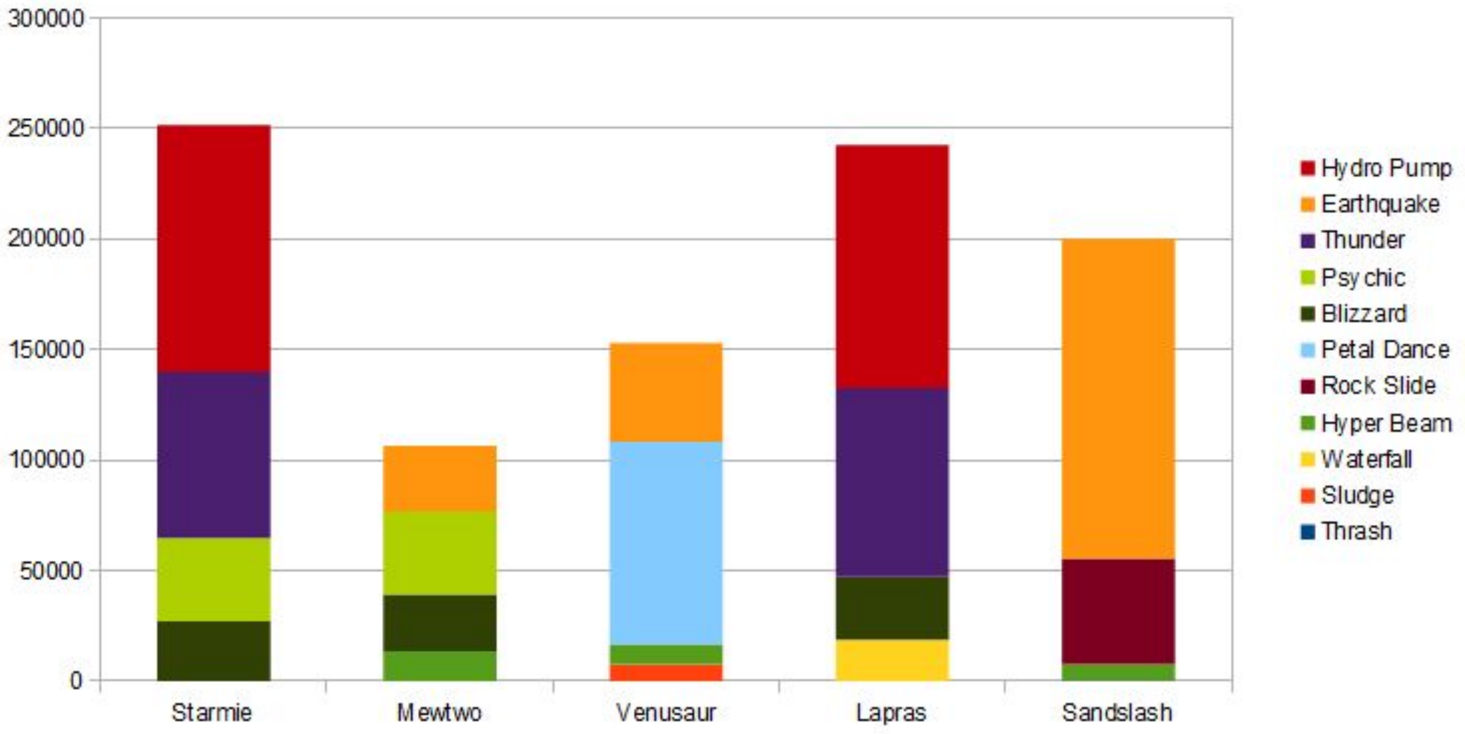
Experimental Results #2: Top Pairs



Experimental Results #3 - Top Teams



Experimental Results #4: Top 4 Moves Each



System API (see README.md in Source Code)

- `./gradlew runDataGenerator` - Simulates battles for teams of 3 found in "PokemonToAnalyze" file. Takes in arguments for determining which teams to select on and how many battles to simulate.
- `./gradlew runPokemonAnalysis 1` Looks at the battle trees for every pokemon in the "PokemonToAnalyze" file and determine the confidence for each pokemon, pair of pokemon, or team based on number argument.
- `./gradlew runMoveAnalysis` Looks at the battle trees for each pokemon to determine how often each move is used to determine the four most common.
- `./gradlew runNextMoveAnalysis` Looks at the battle trees for each pokemon to determine what move to use against the opponent and the confidence that it leads to a win

Data Structures

- To help with quick indexing, almost everything is stored in memory as a hash map, generated from a JSON file.
  - O(1) fetching since it's by ID, and is predetermined when data was generated.
- Also used Multi-Threaded runnable tasks for all the analysis code so that we can eventually port this to work on a cluster instead of a single computer.
  - Data Generation is not as thread safe due to the complexity of that code.

Sample Outputs - Pokemon Analysis Output

```
$ ./gradlew runPokemonAnalysis -Dexec.args=3
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:runPokemonAnalysis
[Blastoise Sandslash Starmie], 0.8890356671070013
[Blastoise Lapras Machamp], 0.8841544607190412
[Blastoise Kangaskhan Kingler], 0.6849087893864013
[Blastoise Mewtwo Kingler], 0.8546875
[Blastoise Jolteon Kangaskhan], 0.8378839590443686
[Raichu Snorlax Venusaur], 0.7496087636932708
...
BUILD SUCCESSFUL
```

Sample Outputs - move-analysis.json


```
…
"Mewtwo" : {
    "Psychic" : 666,
    "Thunder" : 73,
    "Earthquake" : 521,
    "Rock Slide" : 222,
    "Hyper Beam" : 259,
    "Blizzard" : 525,
    "Double-Edge" : 215,
    "Fire Blast" : 88
  },
…
```

Sample Outputs - next-move-analysis.json

…
,{"attacker":"Sandslash","defender":"Electrode","move":"Earthquake","confidence":0.86427598
72900591}
,{"attacker":"Mewtwo","defender":"Muk","move":"Psychic","confidence":0.9982911825017088}
,{"attacker":"Sandslash","defender":"Jolteon","move":"Thrash","confidence":1.0}
,{"attacker":"Blastoise","defender":"Articuno","move":"Rock
Slide","confidence":0.9086193745232647}
,{"attacker":"Mewtwo","defender":"Electrode","move":"Thrash","confidence":1.0}
,{"attacker":"Sandslash","defender":"Dewgong","move":"Rock
Slide","confidence":0.5081912957140493}
,{"attacker":"Blastoise","defender":"Muk","move":"Earthquake","confidence":0.78641180137128
61}
,{"attacker":"Mewtwo","defender":"Jolteon","move":"Earthquake","confidence":0.995249406175772}
...

Good Points and Lessons Learned

- We were continually surprised by the sheer quantity of data generated, and had to keep finding ways to prune it down in order to keep processing times under control
- We investigated the free AWS service as a way to process data, but the virtual machines available to us had less capacity than our personal machines.
- We have tried some AWS services, but still we need integrate more components of AWS services to implement the entire process.
- We learned how complicated Pokemon battles can get!