PageRank: Pokemon

March 2017

1 Introduction

The purpose behind this project is to use Markov chains to rank Pokemon in terms of their compatibility on a competitive team, using the dataset of Pokemon summaries on the competitive battling site Smogon. The idea is not to find the "strongest Pokemon", but the ones that are most synergistic with others (i.e. are useful teammates), in order to build an ideal competitive team.

2 Methods

Theory

We built our Markov chain using the concept of type synergy and the recommendations of experinced Pokemon battlers.

For type synergy, we considered three main scenarios for preferring a Pokemon as a wanted teammate. For each Pokemon, we constructed two lists describing offensive and defensive type matchups for that Pokemon.

Consider Pokemon A with up to two types t_1, t_2 (which determine its matchups against other types). For each of 18 types, we consider three possibilities for drawing an arc from A to a potential partner:

1. The offense of the partner is strong against the type, while A's defense is weak to it.
2. The defense of the partner resists the type, while A's defense is weak to it.
3. The offense of the partner is strong against the type, while A's offense is not very effective against it.

This concept of type synergy allows us to ensure that the chain is irreducible, since every Pokemon is super effective to and/or weak against another, meaning they will have at minimum one arc pointing to and away from it. In addition, we know that the loop of

grass, fire, and water forms a 3-loop, while ghost-ghost forms a 2-loop, so the graph is also aperiodic.

In addition, we can pull from online analyses of Pokemon used in competitive battling to add to our chain, since type synergy alone does not account for differences in viability stemming from differences in base stats, movesets, etc. We can pull out the names of Pokemon used in conjunction with each other as 'links' between Pokemon, similarly to how the demo used links from webpages as links between professors. This is the meat of our project and is how we intend for the chain to be weighted favorably for more-used Pokemon.

When we generated our Markov Chain based off these links, we wanted to weight them so that team recommendations would be favored over type synergy, since the experience of competitive battlers should be our main consideration. We wrote our code to allow testing of different weights to match our results more closely to competitive usage.

We scraped data from the websites Pokemondb and Smogon using the tools Scrapy and BeautifulSoup. Pokemondb was used to compile lists of Pokemon and their types for analysis, as well as a type chart that would be used for generating arcs based on type synergy. Smogon provides detailed analyses on Pokemon in the competitive battling context. Relevant to this project, the "Team Options" provided on each Pokemon's page gave us a list of Pokemon that the visited one would find useful as teammates.

Then we ran Markov chain analysis on the generated transition matrix to find the invariant distribution on the matrix. The Pokemon with the highest probability should be the 'best' teammate.

Pseudocode

See page 5.

3 Results and Analysis

We found that our model is accurate in predicting the top 200 Pokemon by comparing it against the list found at a website that records the most frequently used Pokemon in competitive matches 52% of the time.

Besides that, we also experimented with different weights in order to find the one that gives us the best accuracy. We inferred that models that give higher weights to team-mates tend to perform better.

We also did an analysis on the types of the top 100 Pokemon. When team-mates are not taken into account, we found that Ground-type Pokemon to be the ones chosen most often. On the other hand, Ice-type Pokemon were chosen least often, which aligned with our expectations because they are terrible defensively, and most water types carry an ice-type move for offensive coverage. So, Ice-types tend to be less chosen in teams.

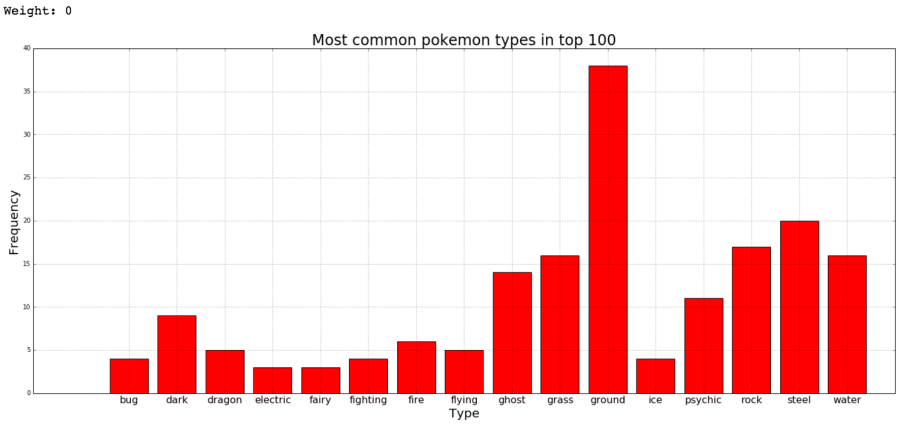

Figure 1: Weight = 0

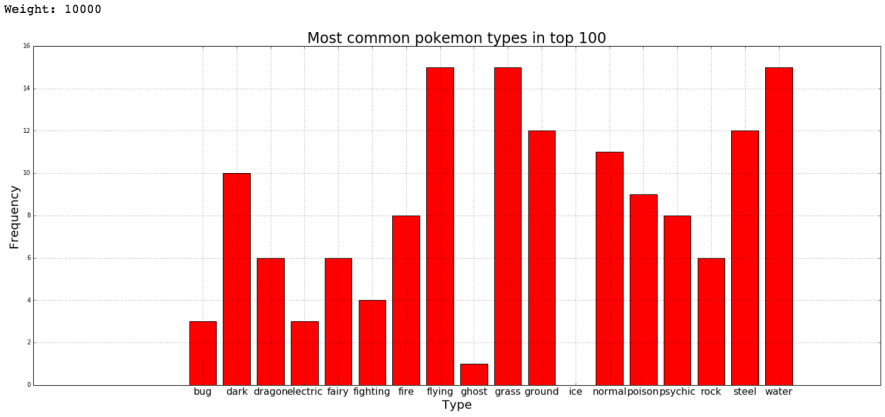

Figure 2: Weight = 10000

We expected the distribution of the types in the top 100 list to be more uniform as the weight is increased. Indeed, the results matched our intuition as shown in Figures 1 & 2.

4 Discusssion/Limitations

Because we only used two parameters in our implementation, we haven't accounted for other ways of finding teammates in generating our arcs, like moveset compatibility or overall type coverage. This limited our ability to find the best team, since these considerations are all important in actually building a competitively-viable team.

In the process of gathering the data, we encountered some difficulties as the web pages are dynamically generated by JavaScript instead of plain HTML. We circumvented this by first saving the page's source file and then parsing the Pokemon name and their types from the source file.

Furthermore, we had to condense the Pokemon to their most basic form (for instance, Alakazam and Alakazam-Mega were both considered Alakazam). This is due to the fact that the Team Description web page uses abbreviation for a Pokemon's name (like Alakazam-M) rather than the full name.

In spite of all the issues, we found this assignment to be fun and rewarding as we were able to apply Markov Chain to something interesting.

Algorithm 1 Build Markov Chain

```
procedure GENERATE RECOMMENDATION ARCS(Pokemon List P)
    d = {}
    for p in P do
        p_T = parse p webpage
        L(p_T) = parse list of pokemon from p_T
        d[p] = L(p_T)
    return d

procedure GENERATE TYPE ARCS(Dictionary[Pokemon, multipliers] P)
    d = { }
    l_P = P.keys()
    for p_1 in l_P do
        for p_2 in l_P do
            if p_1 = p_2 then
                pass
            for Type t do
                if p_1 is defensively susceptible to t and p_2 beats t then
                    d[p_1] += p_2 (arc from p_1 → p_2)
                if p_1 is defensively susceptible to t and p_2 is immune/resists t then
                    d[p_1] += p_2 (arc from p_1 → p_2)
                if p_1 is offensively weak to t and p_2 beats t then
                    d[p_1] += p_2 (arc from p_1 → p_2)
    return d

procedure GENERATE TRANSITION MATRIX(Rec. arcs A_R, Type arcs A_T, weight w)
    I_G = generate dictionary (pokemon name → index)
    P = generate zero matrix
    for each pair (p_1, p_2) in A_R do
        P[I_G(p_1)][I_G(p_2)] += 1
    P = P * w
    for each pair (p_1, p_2) in A_T do
        P[I_G(p_1)][I_G(p_2)] += 1
    return P
```