Final Project Report

Predicting Pokémon spawns within PokémonGo

Team Ultra

MIS 586

Yazan Abu Hijleh

Baoye Zhao

Menglu Pei

Derek Gourley

Po-Yi Du

Yunfei Wei

Abstract

Motivation
Augmented Reality and Virtual Reality are presented to be the new frontiers for the future of gaming. PokémonGo was the first game to bring Augmented Reality to cheap and commonly available consumer devices. Virtual Reality typically requires expensive equipment making it cost prohibitive to consumers. All of this helps to explain the massive popularity of PokémonGo, given that it provides a similar experience to Virtual Reality, but the game runs on a device that the consumer most likely already owns -- their phones. The accessibility and popularity of this game created a wealth of data that we want to explore and better help players, such as ourselves, enjoy the game.

Problem Statement
Given the popularity and intensity of the PokémonGo augmented reality game, players around the world have been working together since the game launched to learn as much as they can about the game. Players have been working to find ways to gain a competitive advantage and maximize the time they spend playing. The most frequent question among new and veteran players is "where can I find this Pokémon?" or, phrased differently, "What Pokémon can I expect to find near this location?"

Approach
Our team set out to use both Network Science and Machine Learning to gain new insights about the PokémonGo to help players optimize the time they spend playing the game. We utilize information about a set of observations recorded throughout a single year, and derive network metrics and co-occurrence measures to create a data set that will be fed into a machine-learning algorithm. The purpose of the algorithm is to predict what type of Pokémon is likely to spawn given some information about the desired type and the surrounding Pokémon. The Hadoop MapReduce technique is also leveraged to pre-process our data to generate useful attributes.

Results
We utilized the network analysis to explore the modularity of our dataset. We then calculate a new network metric, the Class Modularity Majority (CMM) for every observation (row) in our dataset. To build the prediction model, two machine-learning algorithms, Naive Bayes and C5.0 Decision Tree, are implemented in this project. We also include the new network metric in our models as an attribute. As a result, C5.0 holds a much better result, in term of prediction accuracy, comparing to Naive Bayes and the majority baseline of the dataset. This model tends to predict some rare Pokémon's with higher accuracy than the most common ones, which is more useful for the players. Also, latitude and longitude are the most important attributes.

Conclusions
The top attributes to affect Pokémon spawns were found to fit into 3 groups: Pokémon, Player, and Place. The findings incorporated network metrics based on co-occurrence and modularity analysis, and also relied on geographical and population data. The findings confirm certain long held beliefs in the player community, such as those about co-occurring Pokémon having related types, and the population and geographic features affecting spawn rates. More detailed data that covers a longer period of time can be used to increase the value of network measures and accuracy of the model.

Introduction

PokémonGo is a popular mobile game for iOS and Android that has 65 Million active players from around the world. This game was the first major Augmented Reality (AR) application to come to smartphones when it launched on July 6th, 2016. Soon after its launch, it reached 5 million daily active users, which spend an average of 26 minutes playing the game each day (1). Since its launch, the game has been downloaded more than 750 million times (1).

Players of this game walk around with their smartphones in real life to try and catch Pokémon in the wild or battle them in gyms. The game accomplishes this by integrating Google maps with the GPS and camera from a smartphone. By combining all of this together, an augmented reality experience is developed which mimics the popular Pokémon TV show from the 90's. Often, new players are coached by more experienced players, and players who tend to play in groups, where they work together to learn common tips and rules for finding different types of Pokémon.

Related Work

Since the game's inception, players (with technical skills) around the world have been working hard to reverse engineer the game. Given that PokémonGo was the first major game of it's type, players immediately became interested in the code and logic that powers both the server side and the client side, given that both must work together to make the game work successfully. As a result of all of the work to reverse engineer the game, two major websites have been developed for players to share information related to the inner workings of the game or just to share tips they have discover by playing the game. The most prominent website is the "The Silph Road" (2), closely followed by "The Silph Road Subreddit"(3).

Each time a new update is released to the public, players quickly reverse engineer the code and then post their findings to the Silph Road website (2) and the subreddit (3). In addition to this, players also share information about the location Pokémon and where they tend to be spawning. Every two weeks, a nest migration occurs and the Pokémon nests switch locations. A nest is a place in which a Pokémon and copies of it are likely to spawn nearby, similar to a nest of animals in the wild. Another important role of these websites that relates especially to this report, is the basic analytics performed and shared on these sites. Players will share graphs they have made, summary statistics, and results from a day's worth of playing the game.

Previous analytic work has also been completed by using Watson Analytics (4), to visually explore trends in Pokémon spawns and the frequency of Pokémon types. All of this beginning analytics work has already been helpful to both the serious and casual players. Our plans to expand upon the previous analytics by integrating the analytical techniques we have learned this term. We will perform Network Science as well as create and test several predictive algorithms using Machine Learning. The end goal will be to provide additional and new insights into the popular PokémonGo that players will be able to use to guide the way they play the game to maximize their time spent playing the game.

Project Objectives

For this project, our team is interested in trying to learn more about the spawn locations and spawn times of Pokémon within the game. Players around the world try to guess and communicate with each other through social media to share the locations of Pokémon that spawn near them. A major part of the game is communication and teamwork among the players of the PokémonGo game in order to help everyone catch all of the Pokémon in the game.

Our team saw this project as an opportunity to try to understand the factors that influence a Pokémon's spawn time and location in order to predict this in the future. In addition to this, our team wants to better understand the co-occurrence of Pokémon spawns within the game. For instance if a certain Pokémon spawns, which other Pokémon would be most likely to spawn next to it. We believe that Network Science can be used to help us better understand Pokémon co-occurrence and also produce useful derived data that could be fed into a machine-learning algorithm that predicts Pokémon spawn types.

This report will specifically focus on the Network Science techniques that our team used to better understand the co-occurrence of Pokémon spawns, and how that data was used to teach a machine-learning algorithm.

Dataset

For this report, our team used two datasets from Kaggle: "Predict'em All" (5) and "Pokémon Stats" (6). Given that the two datasets used by our team came from Kaggle, the datasets were relatively clean already and the integrity of the data was good. Our team had to do some work with both R and SQL to get the data into a usable format, in which we could conduct the analyses that we set out to complete.

For the "Predict'em All" (5) dataset, the data includes 50 fields for attributes such as the type of Pokémon, the date, the location, information about nearby gyms and PokéStops, and also weather data. Roughly 150 fields for Pokémon co-occurrence (multiple spawns) information are also included. Additional fields were added about each Pokémon's name, type and stats.

For the "Pokémon Stats" (6) dataset, the data contains some basic but crucial features of Pokémon, such as Pokémon name, type, is legendary or not, generation, etc. This dataset can be linked with our basic dataset through PokémonID, which can be used to improve our hypothesis with more features. We dropped the statistical data such as HP and attack power in this dataset because they are not relevant to the PokémonGO game.

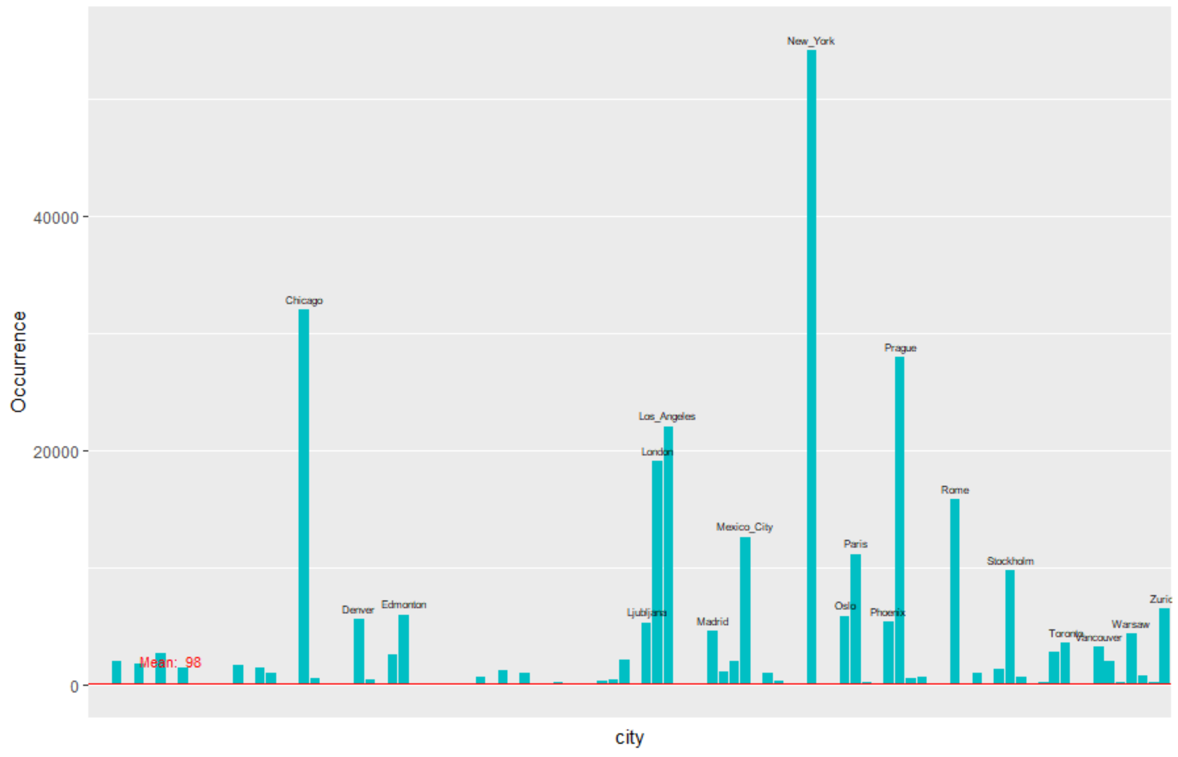
Data Processing

The two datasets from Kaggle were loaded into the R statistical software (7) and all of the attributes for each dataset were combined using the merge function within R on the PokémonID attribute. We conducted a preliminary analysis on this data to decide the scope and filtering of the data. Next, the filtered data was put into Gephi for further pre-processing: 1) we created modularity classes and other network metrics 2) conducted more preliminary analysis and 3) created co-occurrence graphs. We then passed the data through Hadoop to create a new network metric, called **co-occurrence modularity majority**. This derived data was then recombined into the original set and passed into a machine learning algorithm, as described in more detail later in this report.

Data Exploration

Since our objective is to learn more about the occurrence of Pokémon in certain location, we investigated the Pokémon occurrence within different cities, which is shown in figure 1 below.

Figure 1: Bar Chart For Pokémon Occurrences In Cities.



We discovered that the Pokémon occurrence in our dataset is strongly skewed to certain cities while the overall average is 98. Since all of them appeared to be relatively big cities, we suspect the skewness resulted from each city's population size. Thus, we manually collected the population data for eight major cities (New York, Chicago, Prague, LA, London, Rome, Mexico City, and Paris), and ran a correlation analysis between the population data and city occurrences. The result indicated a 0.21 positive correlation, which meant big cities having more Pokémon occurrence might only be a general phenomenon.

As such, cities with more population do not necessarily have more Pokémon occurrences. The observations are so sparse in some of the cities (for example, there is only one observation for the entirety of Cairo, a city of around 10 million people) that it makes no sense to include the cities in our analysis, because there are not enough observations to give a reliable or useful recommendation.

Roughly 82% of the observations are in the top 15 cities compared to 72% in the top 10, which goes to say just how concentrated the data is. Table 1 below, shows the observation counts for a selection of top cities. We decided to discard data from the cities where there are fewer observations (limiting our scope to the top 15 cities), and this did not affect the reliability of our data.

Table 1: Selected Observations of Pokémon Occurrences in Top Cities.

City	Count	City	Count
New York	54,177	Stockholm	9,786
Chicago	32,046	Zurich	6,563
Prague	28,000	Edmonton	6,007
Los Angeles	22,108	Oslo	5,910
London	19,094	Denver	5,619
Rome	15,883	Phoenix	5,429
Mexico City	12,672	Ljubljana	5,361
Paris	11,210		

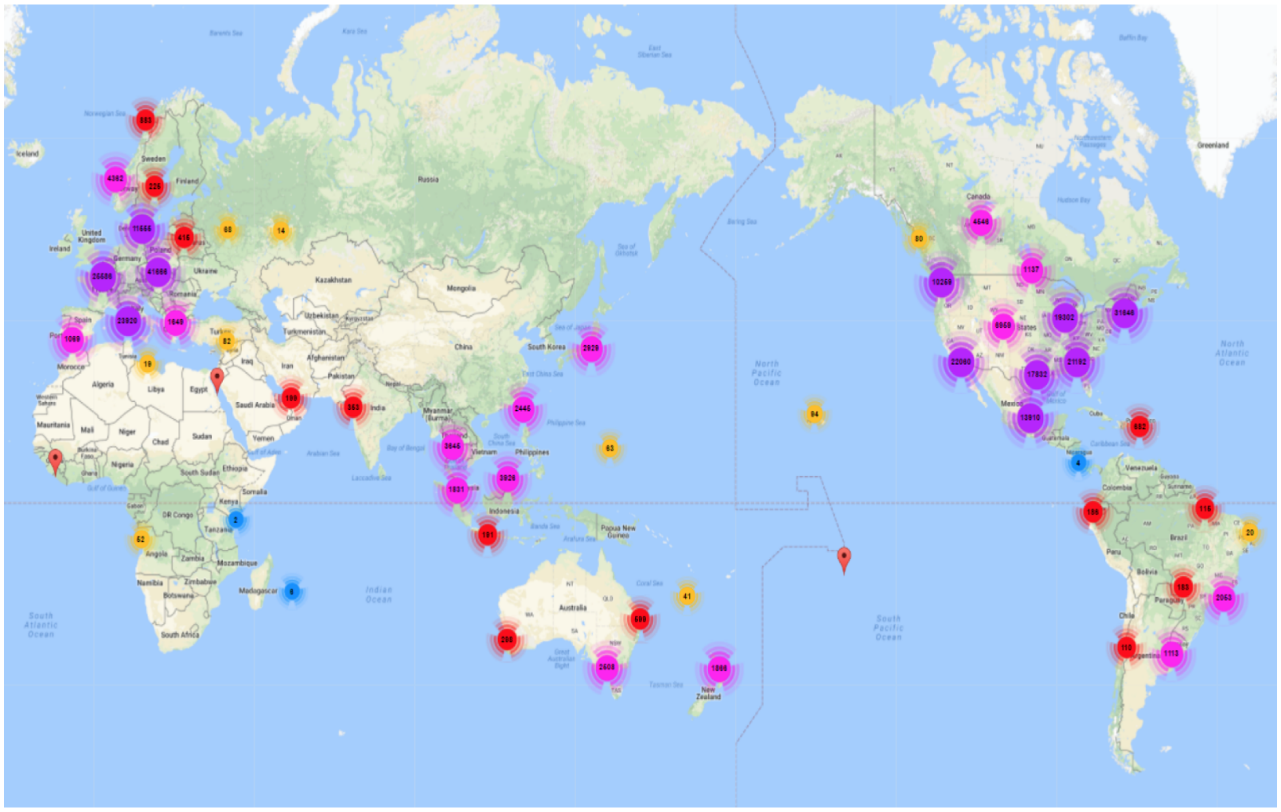
Refining The Scope of The Problem

To get a better idea of the geographic layout of Pokémon occurrences, our team used Google Earth and the location data (Latitude and Longitude) for Pokémon spawns to generate the map in figure 2 below. The map shows the overall count number of Pokémon occurrences. Different colors represent different count number on the map. We use 4 colors to label the number of occurrences. In detail, blue: 0-9; yellow: 10-99; red: 100-999; pink: 1000-9999; purple: more than 9999.

Europe and North America had the most occurrences in the dataset. Almost each country in these continents has a sizable amount of Pokémon spawns. For each continent, it seems occurrences are concentrated in the center.

Second-most Pokémon occurrences are in the cities of Southern Asia. Most of northern Asia has no Pokémon because the app service is not provided or allowed there. However, most of southern areas like Hong Kong, Singapore, Thailand, Philippines, Japan and some related islands have many Pokémon occurrences.
The Pokémon distribution in Latin America is somewhat spread out. The Pokémon are located mostly in Brazil, Argentina and Peru. Large city (like capitals) seem to have the largest number of Pokémon occurrences. Africa, Central Asia and Oceania are almost empty when it comes to Pokémon occurrences, at least when looking at our dataset.

Figure 2: Map of Pokémon spawns.



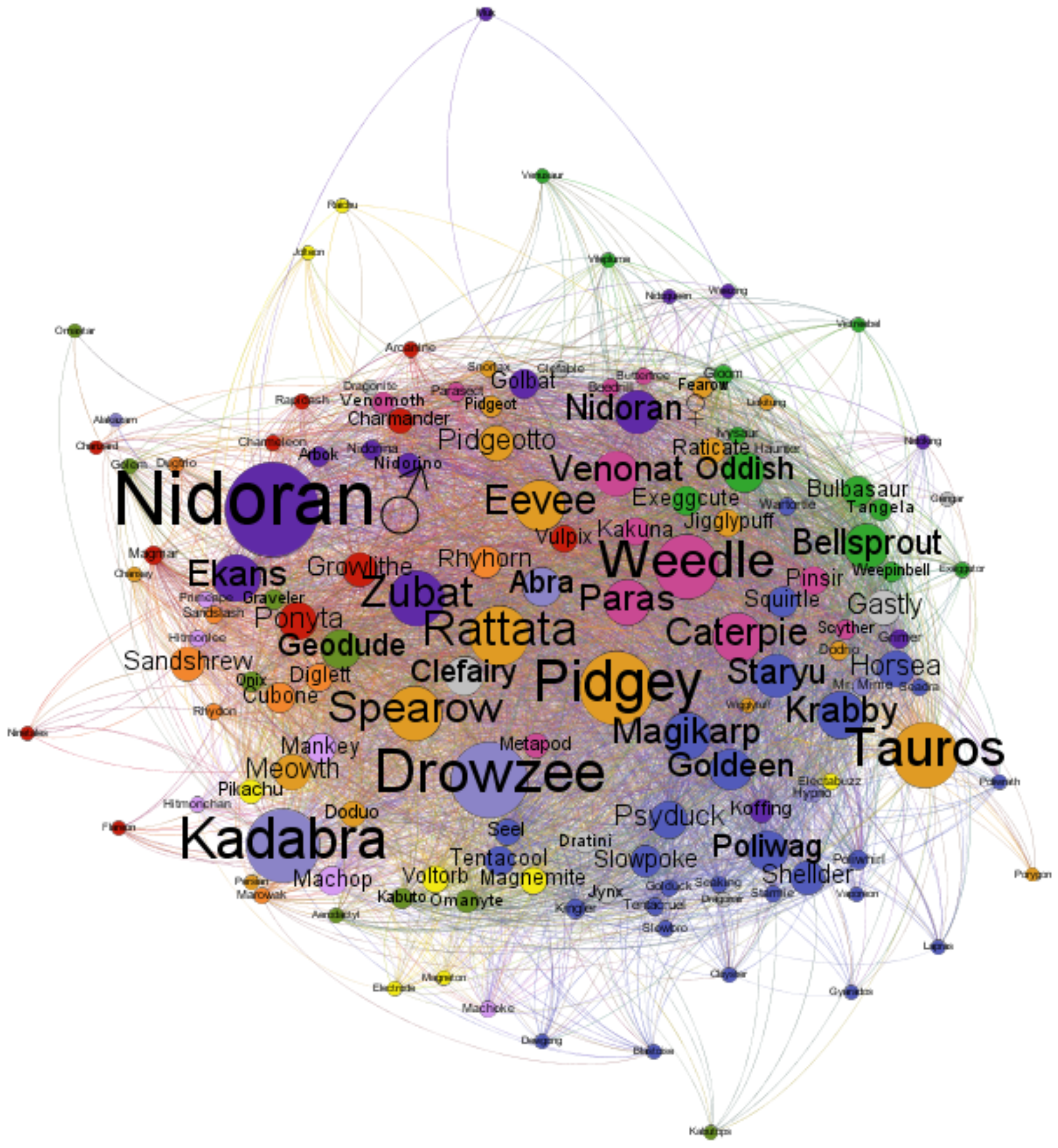
Network Analysis

Our team had access to two datasets: one containing the occurrences of Pokémon around several major cities, and the other containing metadata about each of the 150 Pokémon. The first step was to combine these data sets on the Pokémon ID, which denotes one of the 150 Pokémon. This resulted in a labeled set of Pokémon observations, each with 150 possible connections to other co-occurring Pokémon.

These co-occurrences were transformed into an edge list between the Pokémon, with each co-occurrence contributing one edge weight. As such, the total edge weight represents the number of co-occurrences between the two Pokémon for that edge. We also refined some values for time and location, to make the data more consistent and in a usable format for the Gephi Network Science Software (10).

The results of our co-occurrence network graphs can be seen in Figures 3, 4, and 5 below. In these graphs, the size of a node is the degree of co-occurrence (how many different Pokémon co-occur with that of the node), while the edge weight corresponds to the frequency of co-occurrence between any two Pokémon. The layout algorithm puts Pokémon that occurred frequently closer to each other, and ones that don't occur frequently farther apart. The color represents the primary type of the Pokémon (e.g. fire, water, grass).

Figure 3: Overall Network Graph for Pokémon Co-occurrence.



In the center of the graph we see Pokémon such as Pidgeotto, Raticate, and Arbok, which are Pokémon that have the highest co-occurrence. In other words, these are the Pokémon that are most likely to spawn with other different Pokémon. When playing the PokémonGo game, our team members have noticed that in fact these Pokémon tend to co-occur together. Looking at Figure 3 we see that there is a large number of water Pokémon that spawn together. This is what players would expect given that water Pokémon tend to spawn next to bodies of water. We also notice looking at Figure 3 above that Pokémon of opposite types don't seem to co-occur together. According to the makers of the PokémonGO game, Pokémon have nests and tend to spawn nearby them. In Figure 3 above, it seems to show that Pokémon of similar types tend to have nests nearby each other given their co-occurrence.

Figure 4: “Water” Type and “Normal” Type.

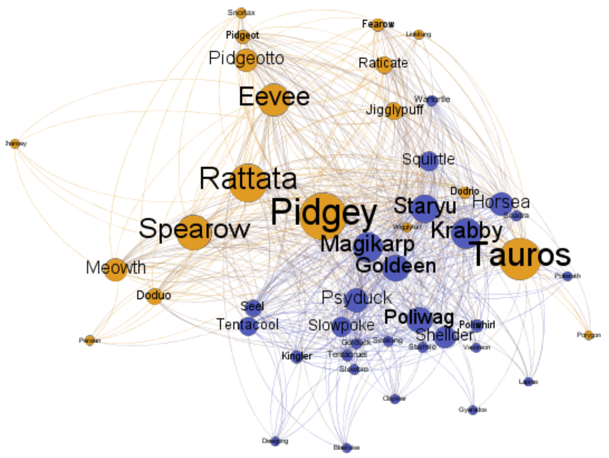
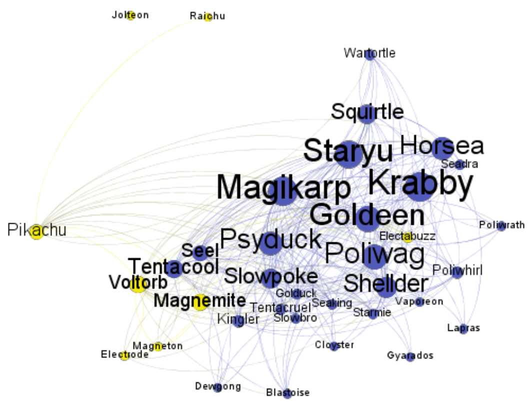


Figure 5: “Water” Type and “Electric” Type.



In Figure 4 above, we see that Pokémon that are Normal type and Water type are the most likely to co-occur together. This finding is very important to players given that they can expect Normal type Pokémon to be the ones most likely to spawn next to water features in the game. If a player were interested in finding Pokémon with a type other than Normal or Water they should not be next to water features in the game. In Figure 5, we see that Electric and Water type Pokémon are least likely to co-occur, which means if a player wants to find an Electric Pokémon they should not be next to a body of water. This lines up with player expectations that Pokémon of opposing types will not spawn together.

Network Measures And Network Analysis Summary

In addition to the above analyses, our team used Gephi (10) to conduct a Weighted Degree Analysis and ran a Community Detection Algorithm.

For the Weighted Degree Analysis, our team ran the average weighted degree subroutine in Gephi, and found that the highest values were for basic Pokémon (e.g. Pidgey and Weedle) that also highly co-occurred together. This initial finding supports the commonly held notion that basic Pokémon are plentiful and easier to find, when compared with more evolved forms. Since these Pokémon are very common, it would follow that they also co-occur with others.

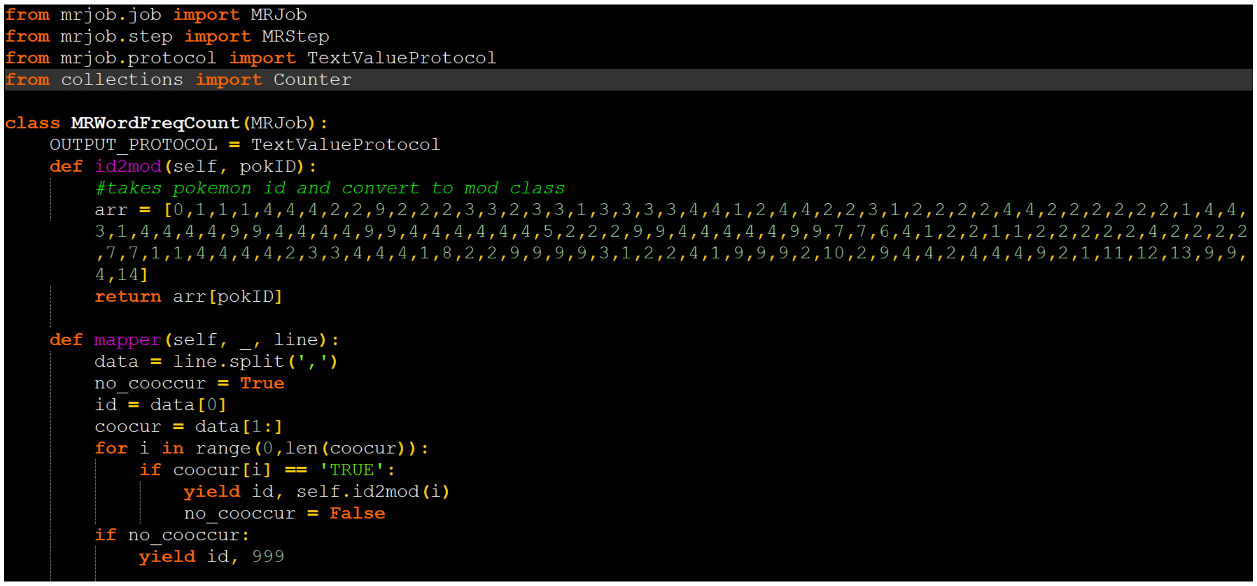
For the Community Detection, our team ran a community detection algorithm several times in Gephi (10), but could not find a common attribute between the nodes in each cluster. We repeated this with weighted edges and without weighted edges, and used the resulting communities in our prediction model.

Hadoop MapReduce

Hadoop MapReduce was used to compute a new network-derived metric, which was the **co-occurrence modularity majority (CMM)**. This metric is calculated for each observation in the following way: 1) for each observation, emit the modularity class associated with each positive or present co-occurrence. Then, 2) sum up the totals of each modularity class for each observation, and 3) pick the modularity class with the largest representation in co-occurrence, or the majority.

For example, if an observation has a Pokémon that co-occurred with 3 others (with modularity classes 2, 10 and 2 respectively), then the co-occurrence modularity majority for that observation is 2, because it represents 66.7% of the co-occurring Pokémon.

Figure 6: MapReduce code for calculating CMM. The mapper emits the matching modularity class for each co-occurring class for an observation.

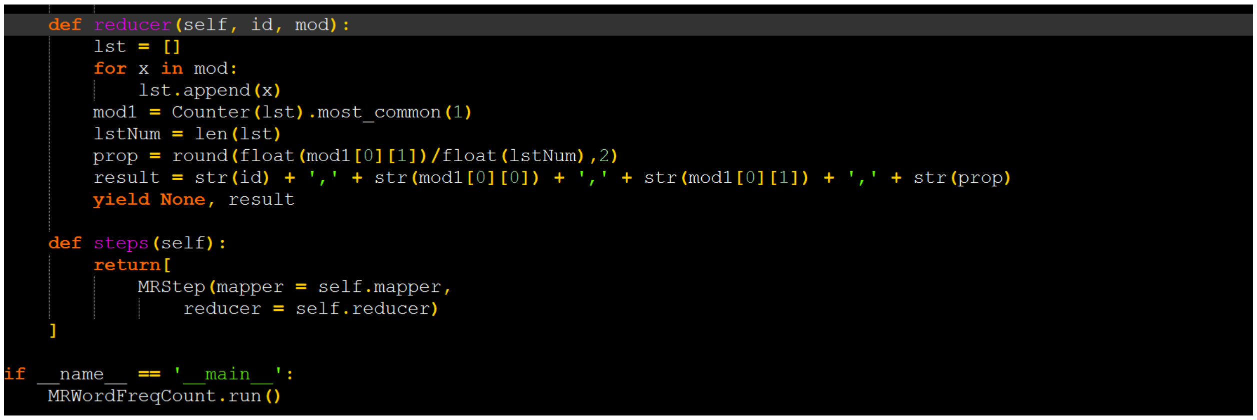


```
from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.protocol import TextValueProtocol
from collections import Counter

class MRWordFreqCount(MRJob):
    OUTPUT_PROTOCOL = TextValueProtocol
    def id2mod(self, pokID):
        #takes pokemon id and convert to mod class
        arr = [0,1,1,1,4,4,4,2,2,9,2,2,2,3,3,2,3,3,1,3,3,3,3,4,4,1,2,4,4,2,2,3,1,2,2,2,2,4,4,2,2,2,2,2,2,1,4,4,
        3,1,4,4,4,4,9,9,4,4,4,4,9,9,4,4,4,4,4,4,5,2,2,2,9,9,4,4,4,4,4,9,9,7,7,6,4,1,2,2,1,1,2,2,2,2,2,4,2,2,2,2
        ,7,7,1,1,4,4,4,4,2,3,3,4,4,4,1,8,2,2,9,9,9,9,3,1,2,2,4,1,9,9,9,2,10,2,9,4,4,2,4,4,4,9,2,1,11,12,13,9,9,
        4,14]
        return arr[pokID]

    def mapper(self, _, line):
        data = line.split(',')
        no_cooccur = True
        id = data[0]
        coocur = data[1:]
        for i in range(0,len(coocur)):
            if coocur[i] == 'TRUE':
                yield id, self.id2mod(i)
                no_cooccur = False
        if no_cooccur:
            yield id, 999
```

The first function defined in Figure 6, id2mod(), contains an array that contains the modularity classes for each PokémonID: The Pokémon with id 1 would have the index 0, and have the modularity class 0. This function is used to encapsulate the modularity class data in a simple to use way. It takes in the PokémonID - 1, and outputs the correct modularity class. The array values themselves were produced through Gephi network analysis.

The mapper loads in the co-occurrence data, where each line is an observation, and emits pairs with the observation identifier as the key and a modularity class as the value. A special value is emitted if the observation had zero co-occurrence.

Figure 7: MapReduce code for calculating CMM. The reducer combines the counts and aggregates them to find the most common modularity.



```
    def reducer(self, id, mod):
        lst = []
        for x in mod:
            lst.append(x)
        mod1 = Counter(lst).most_common(1)
        lstNum = len(lst)
        prop = round(float(mod1[0][1])/float(lstNum),2)
        result = str(id) + ',' + str(mod1[0][0]) + ',' + str(mod1[0][1]) + ',' + str(prop)
        yield None, result

    def steps(self):
        return[
            MRStep(mapper = self.mapper,
                reducer = self.reducer)
    ]

if __name__ == '__main__':
    MRWordFreqCount.run()
```

The reducer takes in each observation id as a key, and a list of a list of modularity classes (with duplicates) that describe the co-occurring Pokémon's modularity classes. We then compute the majority modularity class after adding the items to the same list. The output of this reducer is a comma-separated file that includes the majority modularity class and the proportion of that modularity class for that observation. This data will be fed into the machine-learning algorithm described in the coming section.

Prediction Models

The main prediction models we used in this project are **Decision Tree** and **Naive Bayes**.

We used C5.0 (9) and the Naive Bayes (8) classifiers in R Statistical Software (7) and set the predictor as latitude, longitude, appearedDayOfWeek, terrainType, closeToWater, weather, temperature, windSpeed, windBearing, pressure, population_density, urban, suburban, midurban, rural, gymDistanceKm, Type_1, group, count and prop. Overall, we have 19 variables and three of them are networking features: group, count and prop. Our training data makes up a random sample of 70% of the entire dataset and testing data makes up the remaining random 30%.

Since we would like to predict the occurrence of specific Pokémon (the feature 'Name' in our dataset, the baseline should be the proportion of the most common Pokémon. The Pokémon with the highest frequency is Pidgey, which makes up 17.84% of all of the Pokémon that appeared in our dataset, so the baseline of our prediction models to beat is 17.84%.

We tried Naive Bayes (8) and Decision Tree (C5.0) (9), as they had straightforward implementations that we could readily integrate into our project. The accuracy of Naive Bayes is around only 17%, which is lower than our baseline, so we decide not doing any deep analysis of Naive Bayes. The accuracy of the C50 decision tree is about 43.75% without using features generated by using network analysis, and is improved to 45.59% after applying the network science feature representing modularities. The possible reason is that the many of our observations share similar values for co-occurrence modularity majority, which is due to the data set being relatively constrained in terms of size. This facet of the analysis could be improved with additional data in the future, because that could balance out the distribution of the co-occurrence modularity majority.

Table 2: Top 20 Pokémon with highest prediction accuracy (with co-occurrence modularity majority).

Name	Cases	False Pos	False Neg	Error Rate	Accuracy
Clefairy	2110	40	0	0.00%	100.00%
Gastly	1305	77	0	0.00%	100.00%
Jynx	373	0	0	0.00%	100.00%
Dratini	321	43	0	0.00%	100.00%
Geodude	1451	407	7	0.48%	99.52%
Drowzee	5517	618	67	1.21%	98.79%
Mankey	1174	276	26	2.21%	97.79%
Weedle	15766	9057	1090	6.91%	93.09%
Sandshrew	1169	546	88	7.53%	92.47%
Pikachu	304	50	28	9.21%	90.79%
Growlithe	1046	464	135	12.91%	87.09%
Zubat	5447	2629	726	13.33%	86.67%
Magikarp	4350	2007	650	14.94%	85.06%
Pidgey	29880	13891	5459	18.27%	81.73%
Dewgong	15	16	3	20.00%	80.00%
Ekans	2393	968	492	20.56%	79.44%
Charmander	431	52	99	22.97%	77.03%
Krabby	2457	1155	577	23.48%	76.52%
Voltorb	601	176	150	24.96%	75.04%
Bellsprout	2026	780	520	25.67%	74.33%

The Decision tree shows different results for different Pokémon, which means each Pokémon have different accuracy and error rate based on those inputting variables. In fact, in real-life conditions, each Pokémon may have different possibilities to be caught. It is realized that our accuracy showed that not just the common one like Pidgey, Ekans, Zubat, Mankey have the high accuracy, but many rare one like Clefairy, Dratini, Pikachu also have high accuracy by using the model. Thus, the players could use that to find some of their dream Pokémon accurately.

Table 3: Top 20 Pokémon with lowest prediction accuracy (with co-occurrence modularity majority).

Name	Cases	False Pos	False Neg	Error Rate	Accuracy
Haunter	71	0	71	100.00%	0.00%
Snorlax	41	0	41	100.00%	0.00%
Clefable	40	0	40	100.00%	0.00%
Primeape	39	0	39	100.00%	0.00%
Sandslash	32	0	32	100.00%	0.00%
Dragonair	27	0	27	100.00%	0.00%
Hitmonchan	21	0	21	100.00%	0.00%
Vaporeon	18	0	18	100.00%	0.00%
Wigglytuff	18	0	18	100.00%	0.00%
Dragonite	16	0	16	100.00%	0.00%
Golem	15	0	15	100.00%	0.00%
Marowak	14	0	14	100.00%	0.00%
Poliwrath	8	0	8	100.00%	0.00%
Gengar	6	0	6	100.00%	0.00%
Ninetales	6	0	6	100.00%	0.00%
Venusaur	6	0	6	100.00%	0.00%
Weezing	6	0	6	100.00%	0.00%
Kabutops	4	0	4	100.00%	0.00%
Vileplume	4	0	4	100.00%	0.00%
Alakazam	3	0	3	100.00%	0.00%
Muk	1	0	1	100.00%	0.00%

We also list the top 20 Pokémon with lowest prediction accuracy. The common thing for those Pokémon is they all have few individual cases according to current dataset, so they could not be predicted accurately like those Pokémon, which have many cases. Also, the accuracy could be improved if the Pokémon could be clustered once preprocessing the datasets by the variables like apperedTimeOfDay because some of those Pokémon just appeared in night or morning of the day or they are set rarely appeared in the world because they are supposed to appear once evolvement or hatch executed by players originally.

Table 4: Model attributes, descriptions, and the attribute importance for C5.0 Decision Tree.

Attribute	**Description**	**Attribute Importance for C5.0**
Latitude	Coordinates of a sighting (numeric)	92.23%
Longitude	Coordinates of a sighting (numeric)	88.47%
Appeared Day of Week	Day of the week that the sighting occurred (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)	44.35%
Terrain Type	Terrain where Pokémon appeared described with help of GLCF Modis Land Cover (numeric)	69.35%
Close to water	Did the Pokémon appear close (100m or less) to water (Boolean, same source as above)	43.95%
Weather	Weather type during a sighting (Foggy Clear, PartlyCloudy, MostlyCloudy, Overcast, Rain, BreezyandOvercast, LightRain, Drizzle, BreezyandPartlyCloudy, HeavyRain, BreezyandMostlyCloudy, Breezy, Windy, WindyandFoggy, Humid, Dry, WindyandPartlyCloudy, DryandMostlyCloudy, DryandPartlyCloudy, DrizzleandBreezy, LightRainandBreezy, HumidandPartlyCloudy, HumidandOvercast, RainandWindy)	68.43%
Temperature	Temperature in Celsius at the location of a sighting (numeric)	40.24%
Wind Speed	Speed of the wind in km/h at the location of a sighting (numeric)	35.62%
Wind Bearing	Wind direction (numeric)	32.58%
Pressure	Atmospheric pressure in bar at the location of a sighting (numeric)	57.21%
Population Density	What is the population density per square km of a sighting (numeric)	70.28%
Urban	How urban is the location where the Pokémon appeared (Boolean, built on Population density, >800 for urban)	16.20%
Suburban	How urban is the location where the Pokémon appeared (Boolean, built on Population density, >=400 and <800 for subUrban)	14.37%
MidUrban	How urban is the location where the Pokémon appeared (Boolean, built on Population density, >=200 and <400 for MidUrban)	15.25%
Gym Distance Km	How far is the nearest gym/PokéStop in km from a sighting	47.62%
Type 1	Each Pokémon has a type, this determines weakness/resistance to attacks	100%
Group	Modularity Group	66.68%
Count	Observations of Pokémon Occurrences in the Group	66.62%
Prop	Proportion of the most common Pokémon	51.58%

Results and Evaluation

Our C5.0 Decision Tree model has significantly better performance than the Naive Bayes model. Compared to the majority baseline (around 17%), the C5.0 model has an overall accuracy of 42% without network metrics. Including the network metrics (CMM) allowed us to boost our model accuracy by 2%. Interestingly, this model does a better job predicting some of the rare Pokémon than it does for the common Pokémon. This finding makes sense, the common Pokémon are more likely to contain inconsistent features, since they spawn in more varied locations. All of these factors result in our models having less specific decision rules.

In this model, latitude and longitude appear to be among the most important attributes of the ones we tested. These two attributes might somewhat reduce the importance of other geographical attributes, such as "Urban", "Gym Distance Km", and etc., since each latitude and longitude pair contains its unique geographical information which involve other geographical feature presented as other attributes.

Conclusions (Conclusions and Future Directions)

The top 10 attributes impact the ability to catch a Pokémon can be categorized into 3 groups: Pokémon, Player, and Place. For the Pokémon, the most important attributes are the type, modularity group and proportion. It is worth noting that modularity group and proportion were the network metrics that we derived ourselves. For the player, the population of the city seems to be the most important feature. The bigger a city's population, the greater the opportunity for the player to catch the Pokémon they are looking for. For place, in the current stage, the key attributes are latitude and longitude, which are very high-resolution features that also incorporate some of the other features we used, such as distance to water.

These findings confirm certain long held beliefs in the Pokémon community, such as co-occurring Pokémon often have types that are related and that the population of a city and the geographic features are important variables that impact the spawn rates of Pokémon.

For the future, we hope to have more detailed data such as distance from landmarks or the presence or absence of a special in game event which tend to occur from time to time, which are factors that are known to influence the chance a player has to catch a Pokémon. However, the current dataset we used contains 300,000 rows of data collected over a single week. These constraints limit the amount of information we have about the time dimension of spawns and also events.

References

1 80 Amazing PokémonGo Statistics
Craig Smith - https://expandedramblings.com/index.php/pokemon-go-statistics/

2 Grassroots Pokémon GO Network
https://thesilphroad.com/

3 Pokémon GO's Largest Grassroots Network: The Silph Road • r/TheSilphRoad
https://www.reddit.com/r/TheSilphRoad/

4 Pokémon Go explained with Watson Analytics
https://www.ibm.com/blogs/business-analytics/pokemon-go-explained-with-watson-analytics/

5 Predict'em All.
SemionKorchevskiy - https://www.kaggle.com/semioniy/predictemall

6 Pokémon with stats.
Alberto Barradas - https://www.kaggle.com/abcsds/pokemon/data

7 R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

8 David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2017). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.6-8. https://CRAN.R-project.org/package=e1071

9 Max Kuhn, Steve Weston, Nathan Coulter and Mark Culp. C code for C5.0 by R. Quinlan (2015). C50: C5.0 Decision Trees and Rule-Based Models. R package version 0.1.0-24. https://CRAN.R-project.org/package=C50

10 The Open Graph Viz Platform
https://gephi.org/