What Can We Learn About the Real-World From The Internet?

Chanyu An, Valliappa Chockalingam, Jay Soni

January 26, 2017

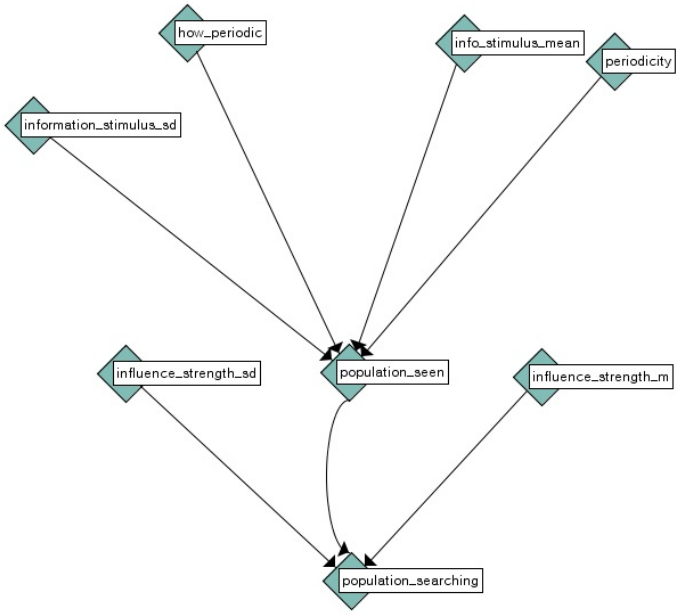
Contents

1 Introduction

On an average day, the Google search engine performs 5.9 billion different search queries. When deciding to search for a certain topic, people have several factors influencing them, including information from their personal lives as well as information from the world around them. In this paper, we discuss our model to predict peoples' searching behavior patterns by quantifying only a few different variables. Our model included using Systems Dynamics in NetLogo to figure out the relationships between real world information and the searching patterns of people over time. The searching trends that we modeled were compared to actual searching trend data found on Google Trends. Conversely, we wrote scripts in Python to take graph data from Google Trends and figure out the exact values of the variables in our model that form the data found in the Google Trends graphs.

2 The Model

To model what we want, we used Netlogo function named System Dynamics. Since there are many variables linked to each other, it is simpler to use System Dynamics than use normal agent-based model. In normal agent based model, it shows data from interaction between individual agents that has own behaviors. For example, in Wolf and Sheep Prediction, we gave rules to each Wolf, Sheep, and Grass, and the model gave us data from interaction between these agents. However, System Dynamics works different way from this. It does not give behavior to individual agent, but it shows how the whole group of agents moves. For example, System Dynamics will program how population of wolf, sheep, and grass increase or reduce as a whole.



In our model, population that searched something on the internet will change according to variables that we set. As you can see on a diagram, we made

6 variables to figure out population that searched something on the internet. Info_stimulus means how information stimulates people, if info_stimulus is very high, and then people have more chance to face this information in their normal life. For example, if a war break out somewhere on the earth, people will hear about the war from news or friends. It is very easy to be exposed by these kinds of information. In this case we can say that info_stimulus is very high. However, another snow day of Michigan, will not be reported heavily on the news, so in this case info_stimulus is very low. We made info_stimulus_mean and info_stimulus_sd(standard deviation) to give some random behavior to this variable. Also, we made variables about periodicity. Periodicity means how often the event occurs. For example, Olympic held once in 4 years. In this case, periodicity is 4 years. How_periodic means that how strictly the event follows periodicity. As I mentioned, Olympics are held once in 4 years strictly, there is no other option for it. In this case, how_periodic is 100. However, Pink Floyd might release its new album in 5 years but it also can be faster or longer. In this case, how_periodic will be pretty low. With these 4 variables, we can estimate population_seen, which means how many people have heard about the information. However, not all the people who heard about the news search about it on the internet. Some people might not have enough interest to search about the topic in the internet. We made influence_strength to adapt this idea. If influence_strength is very high people will more likely search the news on the internet. War new might attract a lot of people's attract, and it will have high influence_strength, but new Pink Floyd album will only attract people who have an interest in Pink Floyd, and in this case influence strength is going to be low. Just like what we did to info_stimulus, we made influence_strength_mean, and influence_strength_sd(standard deviation) to give some random behavior. At last, we can get population_searching from these 6 variables.

3 An Extension: Modeling Real World Trends

The model by itself allows for setting the few variables mentioned above (information stimulus mean, information stimulus standard deviation, periodicity, how_periodic, influence strength mean, and influence strength standard deviation). To make the model more useful and to really understand real world trends, we had to go on to collect data from our model by varying the variables. NetLogo's BehaviorSpace is made exactly for this. Using NetLogo BehaviorSpace, we were able to collect the "population_searching" and the "population_seen" at regularly sampled intervals of time. NetLogo's measure of time is "ticks." There doesn't really seem to be a correlation between "ticks" and any standard unit of time. So, we decided to plot the ratio "population_searching/population_seen" about every 100 ticks (using modular arithmetic and a global counter. We then used these intervals to signify a month, as this is what Google® Trends "csv" (comma separated values) files use. The code for a major portion of this code (which appears in the 'go' procedure) is given below:

```
to go
  set global_counter global_counter + 1
  if (global_counter mod 12 = 0)
  [
```

```
    tick plot population_searching / total_population
    set total total + population_who_search set average total / ticks
  ]
  find_population_seen
  find_population_search
  if (global_counter mod (180 * 12 ) = 0 )
  [
    set average total / ticks
    stop
  ]
end
```

Now, with a "time unit" chosen, we went on to run the BehaviorSpace. There was still the question of what variable values to use. Initially, we took the following values:

Variable	Starting Value	Increment	Ending Value
Information Stimulus Mean	0	10	100
Information Stimulus Standard Deviation	0	5	20
Influence Strength Mean	0	10	100
Influence Strength Standard Deviation	0	5	20
Periodicity	0	6	12
How Periodic?	0	10	100

Now, with these configurations, the BehaviorSpace had a negative number of runs. This is likely due to the fact that there was an overflow caused by the computer being unable to express the number of runs within a standard integer/double.

So, we reduced the variable "Sampling Space" to be more narrow:

Variable	Starting Value	Increment	Ending Value
Information Stimulus Mean	0	20	100
Information Stimulus Standard Deviation	0	5	20
Influence Strength Mean	0	20	100
Influence Strength Standard Deviation	0	5	20
Periodicity	0	6	12
How Periodic?	0	20	100

Once the simulations completed in about five hours, we started trying to model "real world trends." What we tried to do was get some graph from Google Trends® and then use data collected from the model to approximate what variables would need to be set at for that behavior.

For this, we ended up using Python. Initially, using a simple iterative solution whereby the script would go through the file with the data we generated from BehaviorSpace and simultaneously over the file from Google Trends® and use the squared differences of the Google Trends® data of proportion of people searching (compared to the point where most people searched for a topic) and the "Population Searching" as a fraction of the total population was found. So, using the least squares method, we were able to find the set of values with the least deviation from the values obtained from Google® Trends. This method works quite well, however, it was very time consuming (took nearly 15 minutes for every graoh we tried) and had some pitfalls (like the need to align the time's and random graph were just recorded mostly as noise).

To make the analysis faster, we made a few optimizations like analyzing the number of peaks in the graph and comparing it to the total number of data points we have (in the Google® Trends data). Another optimization was to pick sets of data that would likely contain the variable configurations we are looking for (start from the first data set obtained from BehaviorSpace, then check the least squares difference, store the least squares difference (the average of the least squares at each point in time), look at random data set or jump to a data set that likely contains more accurate variable configurations and then repeat the process.

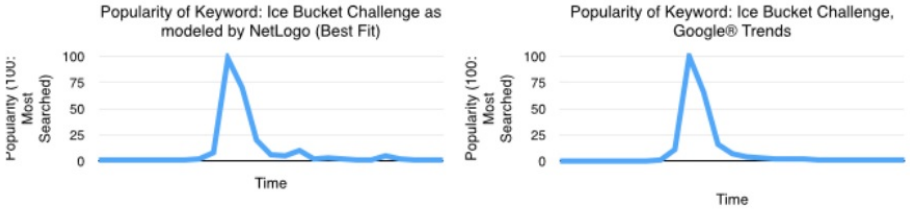
The above optimizations did certainly help, but what really made the run time decrease and the algorithm "smarter" was to use multivariate regression and machine learning. Using NumPy (Python's numerical analysis package), we used a multivariate regression tool given by them with 7 independent variables (time, Information Stimulus Mean and Stadard deviation, Influence Strength Mean and Standard Deviation, and Periodicity and How Periodic, along with one dependent variable, the proportion of people who search to the total population. Following this, we also tried using machine learning to train the computer to be able to recognize graphs; this was considerably more challenging, but, we did get quite accurate results, and with more time and expertise or help from people more familiar with machine learning, the algorithm we used could be made more efficient and accurate.

With all the above said, we then went on to use actual Google® Trends data and see how well the model did. After much trial and error and modifications to the algorithm, we think we have arrived at an accurate model with an accurate algorithm for modelling real world data. These results are given in the next section.

4 Results

To model the searching trends for the Ice Bucket Challenge, we took into account that the periodicity for this keyword was low due to the fact that this was a single event.

4.1 Single Event

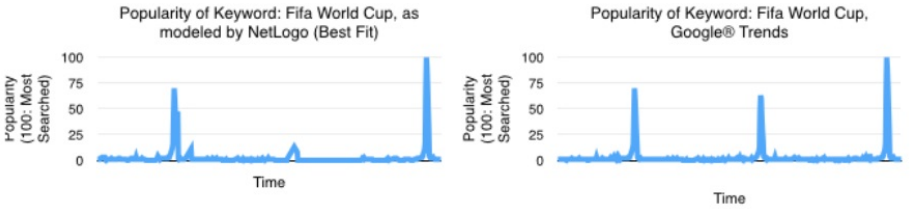


A search trend for a single event involved people searching for something that has a popularity that generally rises once, peaks, then steadily declines. This behavior represents an event that happens one time with rising popularity leading up to it, or a fad. An example of this that we used for our model was the "Ice Bucket Challenge". Over the summer of 2014, the ALS Ice Bucket Challenge was a challenge that people took on social media websites where they posted videos of them getting a bucket of ice poured on their heads to raise awareness for Amyotrophic Lateral Sclerosis (ALS). Prior to the summer of 2014, the data on Google Trends showed that there were very few searches for the keyword "Ice Bucket Challenge". Once popularity for the challenge increased, the number of searches for the keyword went up dramatically.

However, as people stopped doing the ice bucket challenge, the number of searches started going back down. Since this was not a repeating event, we only see this behavior one time on the Google Trends data for this keyword.

To model the searching trends for the Ice Bucket Challenge, we took into account that the searching trends would not be very periodic due to the fact that this was a single event. However, our model model's this behavior by having a rather large period with a very large "How Periodic", so that there would be a peak sometime in the timeline, but not anywhere after that.

4.2 Periodic Event



Search trends for periodic events consist of peoples' searching behaviors for a certain topic repeatedly rising and falling over a certain time period. In our model, we decided to use the example of the FIFA World Cup, a very popular event worldwide that occurs every four years. Due to this event gaining massive worldwide popularity every four years, the search behavior graphs on Google Trends undoubtedly show peaks in the number of searches in 2006, 2010, and 2014, which are all years in which the FIFA World Cup occurred.

When we modeled the searching trends for the FIFA World Cup in NetLogo, we made this keyword very periodic since the event repeats every four years.Therefore, this model will have periodicity of 4 years and periodic effect of 100, since there is no other case that the FIFA World Cup is not held in every 4 years. Also, World Cup is the most popular sports event in the world, therefore, it will have very high information stimulus mean and low information stimulus standard deviation.

4.3 Random Behavior

Random search behavior involves trends for queries for "unimportant" keywords to the general population. One example of this is a search for the keyword "chair". While a search for a chair may be important to some person at any given moment in time, searches for a chair are not periodic, nor are they related to a event, which will therefore make the search trends graph exhibit "random" behavior. If there was to be a high profile murder involving a chair used as the murder weapon, the graph would no longer exhibit random behavior, and look more like the Single Event search graph.

To illustrate the concept as closer to being truly random behavior, we used the search keyword "QAZWSX" in our model. The various peaks in the Google Trends data for this keyword shown above do not have any notable "information stimulus" relating to them. Since it is random, it also does not have notable periodic effect.

5 Conclusion

Our model allowed us to generally predict peoples' searching behaviors based on only the characteristics of information, influence, and periodicity. By knowing this information, we could apply our results to various fields, such as marketing and social psychology.

Our model could be used by a marketing group to figure out what will make a company's desired audience search for various keywords, in order to eventually drive more sales to the company. Moreover, a marketing group could use this model to figure out the searching behavior of their desired audience to improve the company's search engine optimization techniques.

From a social psychological standpoint, this model could be used to describe, predict, and explain the spreading of information. Since Google search trends are a general representation of the information that the general population knows, this model could help to learn about why certain information spread the way that it did, as well as to predict how certain information, such as a tabloid rumor, will spread in the future.

In public policy, if an elected official can know the 6 variables of a current event from our model, that information will be useful when making a policy decision. When we want to boost up or slow down the spread of information, the model can help to find the variable that we have to change to get the result that we want. And then, the elected official can make policy depending on the variable that we need to change.

Our model currently only uses Google Trends data, but our quantification of the factors that influence searching behavior is an example that can be applied to many other types of behavior in the real world other than only searching behavior. So, we could certainly expand this model to be used in areas described above as well as some more which we might not have even thought about.

6 Python Code

6.1 Simple Iterative Solution

```
import csv
f = open('/Users/Valli/Proj3SystemsDynamicsFinal.csv') ''' Path to NetLogo BehaviorSpace File'''
g = open('/Users/Valli/Downloads/report-3.csv') ''' Path to Google Trends File'''

reader_f = csv.reader(f)
reader_g  = csv.reader(g)

counter = 1
loop_counter = 0

curr_dictionary = {}

dictionary = {}

peaks_list = {}

def mean_func(dictionary):
    total = 0.0
    for i in range(0, len(dictionary)):
        total = total + dictionary[i]
    return total / len(dictionary)

def sd_func(dictionary, mean):
    total_sd = 0
    for i in range(0, len(dictionary)):
        total_sd = total_sd + abs(dictionary[i] - mean)
    return pow(total_sd / len(dictionary), 0.5)

def peaks_func(dictionary, mean, peaks_list):
    numofpeaks = 0
    peaks_list = []
    for i in range(1, len(dictionary) - 1):
        if(dictionary[i] > mean + 0.25 * mean and
        dictionary[i + 1] < dictionary[i] and dictionary[i - 1] < dictionary[i]):
            numofpeaks += 1
            peaks_list.append(i)

    if(len(peaks_list) < 2):
        return [numofpeaks, 0]

    peak_distance_total = 0
    for i in range(0, len(peaks_list) - 1):
        peak_distance_total += peaks_list[i + 1] - peaks_list[i]
    return [numofpeaks, (peak_distance_total / numofpeaks) ]

g = {}

```

```
for row in reader_g:
    if(len(row) == 0 and loop_counter > 3):
        break
    if(len(row) > 0):
        if(row[0][0].isdigit()):
            g[loop_counter] = float(row[1]) / 100.0
            loop_counter += 1

loop_counter = 0

for row in reader_f:
    if row[0].isdigit():
        if int(row[0]) == counter and loop_counter < len(g):
            curr_dictionary[loop_counter] =
            (float(row[8]) - float(g[loop_counter])) *
            (float(row[8]) - float(g[loop_counter]))
            loop_counter += 1
        if int(row[0]) != counter or loop_counter >= len(g):
            curr_mean = mean_func(curr_dictionary)
            dictionary[counter] = [curr_mean, row]
            counter += 1
            loop_counter = 0

best = float("inf")
index = 0

for i in range(1, len(dictionary)):
    if dictionary[i][0] < best:
        best = dictionary[i]
        index = i


print best[1]
```

6.2 Multivariate Regression

```
import csv
f = open('/Users/Valli/Proj3SystemsDynamicsFinal.csv')

import numpy as np

reader_f = csv.reader(f)

y = []
x = []

counter = 0
loop_counter = 0

for row in reader_f:
    if row[0].isdigit():
        if int(row[0]) == counter:
            y.append(float(row[8]))
            x.append([row[1], row[2], row[3], row[4], row[5], row[6], row[7]])
            loop_counter += 1
        if int(row[0]) != counter:
            counter += 1
            loop_counter = 0

X = np.column_stack(x + [[1]*len(x[0])])
beta_hat = np.linalg.lstsq(X,y)[0]
print beta_hat
```