Home Safe: An Automated Directions Service for Finding the Safest Walking Path

Michael Cao
Department of Computer Science
Stanford University
Stanford, United States
michaelcao@stanford.edu

Isaac Cheruiyot
Department of Computer Science
Stanford University
Stanford, United States
icykip@stanford.edu

Atem Aguer
Department of Computer Science
Stanford University
Stanford, United States
atemjohn@stanford.edu

***Abstract*—In this paper, we attempt to develop a program that outputs the safest walking path from point A to point B, using San Francisco as a model. We use KNN to classify the risk level of a certain street based on surrounding crime, and we use UCS for our pathfinding algorithm. We have developed multiple iterations of our algorithm that successfully avoids crime "hot spots" while still outputting a relatively quick route. This paper discusses the process, the dataset used, the algorithms, and the results.**

I. Introduction

San Francisco is one of the largest metropolitan areas in the United States, as it is home to over 800,000 residents and is a major tourist destination. At the same time, with a crime rate of 64 per one thousand residents, San Francisco has one of the highest crime rates per capita. Especially for tourists, knowing how to navigate the city in the safest manner is a daunting task. Thus, we aim to develop a simple routing service that provides the user the best walking path from point A to point B, factoring in walking time and the surrounding crime.

II. Literature Review

A slew of research has been dedicated to applying artificial intelligence techniques to analyze crime data. Most studies are focused on classifying crime itself, however our project is more focused on the search problem of finding the safest route based on already classified crime data. Regarding the research performed on classifying crime, a study performed by researchers in British Columbia, Canada took two approaches with two different classification algorithms to analyze crime in Vancouver [1]. In the first approach, they used the K-nearest-neighbors algorithm (KNN), assigning a unique number to each neighborhood and crime category. Each new input is classified by searching for its nearest neighbor within the set of previous inputs. This approach achieved a 40.1 percent accuracy after training for 2209 seconds. Their second approach utilized decision trees, choosing a feature that best splits the data according to the Gini impurity and information gain functions. They also performed 5-fold cross-validation to help prevent overfitting the data. This approach achieved a 39.9 percent accuracy after training for 101.73 seconds. For both approaches, they applied the boosted decision tree algorithm, specifically AdaBoost, and saw marginal improvements of 2-3 percent. Similarly, a group of students from Shrimati Kashibai Navale College of Engineering in Pune India proposed a solution to our same problem using an older version of the same data set (from 2016). Their approach theorized utilizing an ID3 decision tree and a street safety weight assignment that was based off of the punishment prescribed for the crime [2].

As seen with the above examples, Clustering algorithms are very useful when working with multi-attribute data; in the case of crimes, for a single crime, there are many attributes, such as neighborhood, time of day, entry method, etc. This can also be applied to geospatial data to identify hotspots. Spatiotemporal analysis (tracking proximity to other events such as sporting events or concerts) can also improve the accuracy of predictions. In our project, we are simplifying the classification of crime greatly, since we are more focused on search, but our classifications, as described later, are reasonable in terms of gauging the danger a certain type of crime poses to walkers.

III. Dataset

A. Crime Data

For our crime data, we are using the incident report from the San Francisco government website, consisting of over two million cases over nearly two decades. Because recent data is much more relevant and accurate than older data, as crime patterns shift over time, we decided to only utilize a small amount of data for our risk classification, using crime data from January 1, 2017 to May 25, 2018. Our dataset includes relevant variables such as: crime category, date, time, coordinate location, and neighborhood. With this information, we can construct a numeric value for each crime incident that rates the risk that the incident contributes to the area. In classifying the risk level for each crime, we needed to manually assign crime types to a certain category based on the crime's impact on walking safety:

- Group 5: Assault, Robbery, Arson, Vehicle theft, Kidnapping, Sexual Offense-Forceable
- Group 4: Theft/Larceny, Drugs, Sexual Offense Non-Forcible, Weapons Laws, Stolen Property, Burglary
- Group 3: Suspicious Occurrence, Missing Person, Secondary Code, Disorderly Conduct, DUI, Drunkenness
- Group 2: Vandalism, Pornography, Trespassing, Other Offenses

- Group 1: Non-Criminal, Liquor Laws, Prostitution, Gambling, Fraud

Note: some categories are excluded from the risk level assessment as they occurred at such a low rate and were deemed "not high risk" and were thus not included.

Raw Crime Node Example:

Location: $(-122.454363730782, 37.7347598143379)$

RiskLevel: 4

B. Failed Approach: Google Maps API

For our map data, we originally intended to leverage Google's powerful Directions, Distance Matrix, and Roads APIs to implement our navigation algorithm, however we ran into issues with their limited API calls, which could not fully serve our needs. For example, the Directions API could only output 1 to 3 of the best routes and could not output all possible routes. Other APIs from large companies such as Waze and Yahoo resulted in a similar story.

C. Open Source Map Data and Libraries

After trying to work with the Google Maps API, we came across an open source map data site, OpenStreetMap, which hosts a plethora of user-validated map data. Seeing that we could extract easy to work with data that were relevant to our purposes, such as street nodes and edges, we decided to use OpenStreetMap for our routing service implementation.

OpenStreetMap's feature-rich data makes it a great dataset for street network analysis, and it prompted the design and release of OSMnx, a Python library built on NetworkX, geopandas, matplotlib, and R-tree, which allows for easy projection, visualization, and analysis of complex street networks. OSMnx is very useful for conversion and preprocessing of raw OSM data and is an important tool in our pipeline.

IV. Methods

We needed to first obtain a graph representing all streets in San Francisco. To do this, we used OSMnx to pull the OSM map data from San Francisco and generate a multigraph with nodes that store coordinates and edges that store the length of the path. Additionally, NetworkX allows us to view the exact street graph structure, which is very helpful for visualization of the paths.

A. Speed Comparison

To compare how fast our safety-oriented paths are compared to a traditional navigation service such as Google Maps, we implemented a simple mapping that finds the shortest path, based on distance alone, between two coordinate points using a UCS search implemented in the NetworkX library. This serves as our baseline with the rationale that the shortest or fastest path somewhat reduces one's exposure to crime compared to a longer path. Given the current location and destination as coordinate pairs, we find the nearest nodes on the street graph to these two points, plug them into the OSMnx path algorithm, and as an output, get a path, which is formatted as list of node tuples that indicate the edges of the path from the current location to the destination. This gives us a rough estimate at what the fastest path is, which helps us gauge what our tradeoff between safety and speed will be.

Sample Path:

$[(-122.454263337194, 37.7341744099267),$
$(-122.454363730782, 37.7347598143379),$
$(-122.454363730782, 37.7347598143379)]$

B. Assessing Risk Levels

In order to make predict the safety of individual paths we had to formulate a method for assigning a weight to each edge based on our crime data. In order to do this, we initially incorporated K-nearest-neighbors (KNN) with varying k values:

$$P(y = j|X = x) = \frac{1}{K}\sum_{i \in A} I(y^{(i)} = j) \tag{1}$$

Given the location of the starting node of an edge, KNN finds the 20 closest neighbors and classifies the edge based on the risk levels of the neighbors – the risk level being an integer from 1-5 with 1 being the safest and 5 being the least safe. However, after testing this approach, we found that it classified streets that were close or connected to each other very similarly, even if one street clearly had less crime along it than the other.

To improve upon KNN, we developed a new weighting algorithm that calculates radial risk based on crime within a ellipse-shaped border, which is drawn using the start and end nodes of a path as the major axis:

$$\frac{((y - y_c)cos(\theta) - (x - x_c)sin(\theta))^2}{a} + \tag{2}$$

$$\frac{((y - y_c)sin(\theta) + (x - x_c)cos(\theta))^2}{b} \tag{3}$$

$$\leq 1 \tag{4}$$

C. Safest Path

In order to find the safest path, we must update the weights of each node in the street graph with the attribute for every edge that is equivalent to the risk level of the origin node. We then ran the NetworkX UCS search once again on a given current location and destination, both as coordinates, using the risk level as the weight instead of the length of the route. This gives us an output as a series of nodes that optimizes safety and distance.

D. Examples

V. Evaluation Metric

To evaluate the speed of our routing model, we compared the routes created by our baseline model with those created by Google Maps. And from a visual standpoint, the routes

generated by our baseline model have a strong correlation to those recommended by Google Maps for the same pair of origin and destination addresses. Below we ran our model on sample origin, destination pairs for locations in San Francisco Metropolitan Area and compared our results against those created by Google Maps. There may be slight differences in some of routes since google maps is factoring in other constraints like traffic, shortest time, etc while our model simply generates the shortest route between the two endpoints. Note that for this examples only consider the blue lines on our map since those indicate the shortest routes generated without any constraints applied. The red lines will be discussed in the results analysis section below. The images are at the end of the paper. The reason for applying qualitative metrics in our evaluation is attributed to the fact that there is no specific quantitative way available at the moment that can inform us on the accuracy of our results. But visually we can still see the differences in the routes generated by our model in presence and absence of constraints and we are able to correlate the streets avoided in the route to those that are riskier to move through. This therefore acts as an informative metric for evaluating our model.

VI. RESULTS ANALYSIS.

We chose certain routes specifically for a reason. They pass along streets in San Francisco with some of the highest levels of crime reported i.e Geary Street and Byrant Street. First we generated the shortest route from the origin to the destination without taking into account any risk level constraints. It was purely based on the shortest distance between the endpoints. This generated the blue lines you can see on figures $Fig.1.$ and $Fig.3$. In comparison to those generated by Google Maps, they are nearly the same. The red lines represent the routes generated by our model after applying crime risk level constraints in San Francisco neighborhoods to the routes being generated and as you can it was able to avoid both streets in recommending a safe route.

VII. FUTURE WORK

For our next steps, we will fine tune our algorithm to improve the tradeoff between time and risk, and we will adjust the hyperparameters and other parameters of our model (such as the K value, edge weights, etc.). Our biggest challenges are improving the tradeoffs and preventing overfitting. We will also overlay our map on top of a heat map to clearly show that our routes are avoiding crime "hot spots."

REFERENCES

[1] S. Kim, P. Joshi, P. S. Kalsi and P. Taheri, "Crime Analysis Through Machine Learning," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, 2018, pp. 415-420, doi: 10.1109/IEMCON.2018.8614828.

[2] Sarang Tarlekar et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (3) , 2016, 1536-1540

VIII. RESULTS

A. 501 California Street to 899-869 Geary Street

Fig. 1. Route generated by our model. Note that the blue path is the path generated based on only distance, while the red path is generated based on risk level.

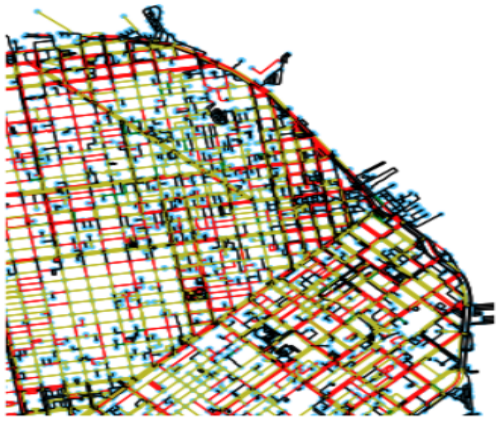

Fig. 2. Route generated by Google Maps.