Predicting Mass Casualty Events from 911 Data Streams

Christopher Vanhook, Eugene Kim, Selamawit Damte

University of California, San Diego

Advised by Ilya Zaslavsky

Abstract

This paper describes the full-stack development of a real-time event monitoring application with a goal to enhance emergency response and public safety. The development cycle occurred using local systems, with data derived from a Postgres server, processing and Flask server development in Python backend, and a React application front-end. Unsupervised learning was used in the form of Incremental Density-Based Spatial Clustering of Application with Noise (DBSCAN), to be able to dynamically identify and categorize anomalies in streaming 911 call data. The approach allows for real-time monitoring of calls and traffic conditions, with notifications alerting the user of events occurring in real-time.

Introduction

Mass casualty events, including both mass shootings and significant weather events, have been increasing in frequency over the past decade. Calls detailing these events result in the proper dispatch of emergency services where seconds can equal lives. However, the U.S. has a disparate 9-1-1 alert system, which is undergoing attempts at standardization. Thanks to organizations such as NENA, a non-profit with the mission to foster the technological advancement, availability, and implementation of a universal emergency telephone number system and APCO, the world's oldest and largest organization of public safety communications professionals, public safety answering points (PSAPs) are being transitioned from sometimes analog systems; to legacy 911 systems, finally, to Next Generation 911 (NG911), a system that assumes in a modern age, the caller uses modern technology (wireless devices).

The legacy system that Next Generation 911 replaces works by the 9-1-1 call, sending an automatic number identification (ANI) to be queried in an Automatic Location Identification (ALI) database for the caller's street address. ALI will list the phone numbers and the associated civic address, which is the address the 9-1-1 authority assigned. The ALI address is

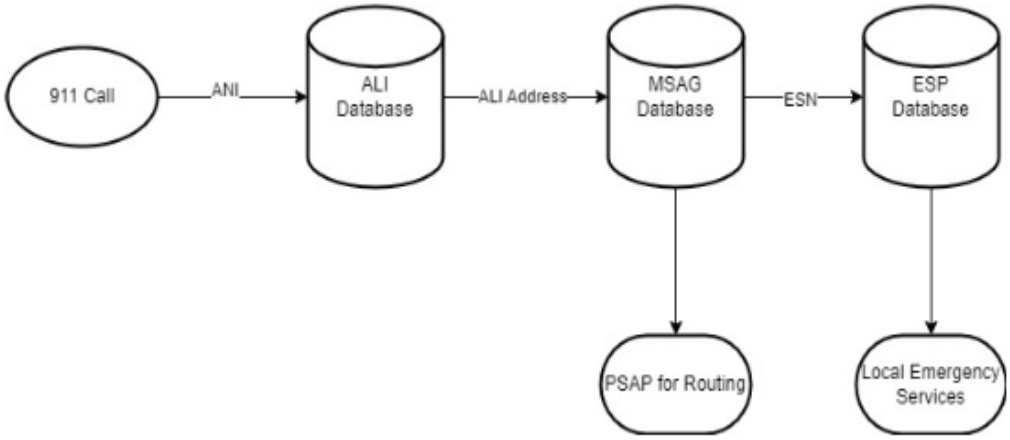

Fig I. Legacy 911 System Query

then used to query the Master Street Address Guide (MSAG) for the Emergency Service Number (ESN), which identifies the correct PSAP that the call should be routed to, as well as the associated law, fire, and EMS responders that correspond to that address. The ESN is then used to query the Emergency Service Provider lookup table for appropriate Law, Fire, and EMS associated with the ESN.

Next Generation 911 (NG911) is a digital, IP-based system that will replace analog 911 systems. It is a system developed to increase the data sharing between jurisdictions and dispatch emergency systems that are vital for the safety and security of local communities. With our increased abundance of data and communication technologies, NG911 uses these resources to facilitate connection and lead to coordinated response efforts across the United States.

NG911 works by the 9-1-1 call sending a geodetic location in PIDF-LO format, which goes to the Emergency Call Routing Function (ECRF) and simultaneously queries the roads and address points as well as the PSAPs and Emergency Service Boundaries—referred to as a point in polygon query. The 9-1-1 call is then routed to the correct PSAP. NG911 has five required GIS layers associated with the data: (1) Road centerlines, (2) Site/Structure address point, (3) PSAP Boundaries, (4) Emergency Service Boundaries, and (5) Provisioning Boundaries. NG911 has additional GIS layers that aid in call routing; however, they are not required now.

Both systems ensure the call is routed to the correct PSAP. Further, this data must be maintained by a wireless operator per municipality border, responsible for maintaining and submitting the data. Both systems result in a dataset with geospatial data, query timestamps, and additional metadata like device carrier.

The challenge is using this call data to identify event hotspots for streaming time series and incorporating event notifications to allow dispatchers to quickly monitor and identify trends leading to mass casualty events for proper dispatch numbers and protocol. Producing a tool that shows a heatmap of call density anomalies along with alerts for triggered potential event thresholds based on external parameters (e.g., past events, external social media, Computer-Aided Dispatch CAD data, or weather reports) would arm the dispatchers with the ability to save time, and in turn save lives.

The data science approach can be broken down into the following steps:

1. Data collection and Preprocessing : Gathering 911 call data and ensuring its quality preprocessing, including timestamp alignment, handling missing values, and normalizing data across different regions and time periods.
2. Exploratory Data Analysis (EDA) : Creating an EDA to identify patterns and trends in the data, such as peak call times and geographic hotspots.

3. Incremental DBSCAN : Leveraging the incremental DBSCAN algorithm to dynamically adjust clusters as new data is fed in, enabling real-time anomaly detection.
4. Real-time Data Streaming : Implementing a real-time data streaming pipeline to continuously feed new data into the model for ongoing analysis and alert.
5. Visualization : Developing a React - based front-end to visualize data changes in real - time and notify user of events through alert notifications.

Formulating the Right Questions

1. How can we preprocess 911 call data to ensure its quality and relevance for real time analysis?
2. How can Incremental DBSCAN be effectively implemented to handle streaming data and adjust clusters dynamically?
3. How can we design a user interface to visualize real time data changes and alert notifications effectively?

Related work

There are a couple projects that have explored the use of Machine learning and clustering algorithms for emergency response applications.

Anomaly Detection in Call Data: The project was initially run on a small subsample, using an Isolation Forest approach, which yielded a similar result in isolating the anomaly. However, this is relative to one time window and one event.

Credit Card Fraud Detection: The idea of detecting credit card fraud is not too dissimilar from our own, we are trying to find minute occurrences amongst populations. However, the proportions vary slightly, we're still trying to find the needle in the haystack, but because of the call density aspect, the needle in our approach is the size of a shovel. The credit card fraud detection project outlines different approaches used to identify instances of fraud within a population of transactions; and, settles on a genetic algorithm (GA). This idea proved similar to ours in ideas of what to test to find the best fit for our end product.

Our project builds on these foundations by integrating incremental DBSCAN for dynamic anomaly detection, implementing real-time data streaming, and developing a comprehensive visualization and alert system customized to the needs of emergency response teams and public safety.

Team Roles and Responsibilities

Each team member has specific roles and responsibilities that contribute to the overall success of the project. The distribution of tasks ensure that all aspects of the project are covered from data modeling, cleaning and front end implementation to user experience enhancement.

Salem - Data Engineer

1. Data Collecting and preprocessing:
    a. Gather and clean 911 call data
    b. Address timestamp inconsistencies, missing values, and data normalization
2. Back End
    a. Develop and maintain the server application using Django
    b. Implement the real - time data streaming pipeline to handle incoming data.
3. Data Management
    a. Design and manage the database scheme for efficient storage and retrieval of call data
    b. Ensure data integrity and security

Christopher - Machine learning Engineer

1. Back-end Development
    a. Write Python backend for data preprocessing and serving with Flask
2. Algorithm
    a. Implement and fine-tune the incremental DBSCAN algorithm for real time, streaming anomaly detection in Python
3. Front-end Development
    a. Develop the React-based front-end application
    b. Implement interactive dashboards to visualize real time data changes and alert notifications for user

Eugene - Machine learning and User Experience Designer

1. User Experience (UX) Design
    a. Design a user friendly interface that allows stakeholder to easily interact with the data
    b. Ensure that the visualizations are clear, concise, and informative.
2. Documentation
    a. Prepare technical reports and presentations to communicate findings and insights.
    b. Document the entire data pipeline, from data collection to model development and visualization
3. Scalability and Efficiency
    a. Implement parallel and distributed computing techniques to handle large volumes of data.
    b. Optimize the model for real - time processing and quick response times.

Collaboration

Regular Meetings

- Schedule weekly meetings to discuss progress, challenges and next steps
- Use of collaborative tools, including: Github, Discord, Google Drive and online suite, and email to keep in regular contact regarding project updates and next steps.

Data Acquisition

Data Sources

- 911 Call Data
  - This dataset contains Michigan records of 911 emergency calls, including timestamps, geographic locations and various phone company metadata required by state and local regulatory agencies.
- Gun Violence Events
  - This dataset contains data on mass shootings, defined as incidents where four or more people are shot or killed, excluding the shooter. The source its data from a variety of publicly available reports, including media outlets, law enforcement, and governments.
- Mass Casualty Wikipedia Database
  - The data on Wikipedia is compiled from various sources, including historical records, new reports, official government reports, scholarly research and other reputable publications. It contains Event Name, Date, Death Toll, and Description.

Solve identified questions

Accessing data from a variety of sources and using different technologies provides an advanced view of the emergency response.

Looking at all the data together helps us make better decisions because it shows us what affects how quickly help arrives in emergencies and how we distribute resources.

Data Collection

1. Database Queries: For accessing historical 911 call data stored in relational databases.

2. Cloud Storage solutions: Using services like AWS(Amazon Web Services) for scalable and reliable data storage.

Data Sizes

1. 911 Call Data: It has around 3 millions rows
2. Gun Violence events : 2000 Rows
3. Mass Casualty events : 400 Rows

Data Pipelines

1. Cloud Vs Local
    a. We used local storage for our end product, with the potential expansion to hosting on AWS. However, due to the sensitive nature of our data, we opted to not utilize cloud storage.
2. Database Vs. Flat files:
    a. We query a Postgres database; however, rather than calling this connection each time, we have opted to download to our local systems to speed up processing times.

Local systems are used across the board for backend data processing, serving the data through a localhost via Flask, and hosting the frontend React Application on a localhost. Our system is entirely scalable and able to be deployed on a cloud system, given setting up connections to a live database, deploying our backend processing, and hosting our frontend; however, this approach was not required due to the static nature of our data sources.

Data Preparation

Data Issues

During the data preparation phase, two primary were identified in the datasets

1. Timestamp : The problem with the timestamp is inconsistencies. Variations in timestamp formats and missing timestamp in some records.
2. Incomplete Data : Missing value for geographic location and skew in distribution of data across year and state

Further, the team hoped to acquire additional CAD data, which would give a downstream analysis of calls placed, resulting in potential transcripts and aid in event labeling.

Pre-processing Methods

1. Timestamp Standardization
    a. Converting all timestamps to a consistent formation and filling in missing timestamps.
2. Handling Missing Value
    a. Missing values were filled using the most frequent category or dummy data.
3. Removing Duplicates
    a. Remove duplicate records based on unique identifier and timestamps
4. Cleaning categorical columns
    a. Involved text cleaning for geographical labels, required to bucket data for accurate unsupervised model application

The preparation of data allowed for both cleaning of data to allow for bucketing with the application of the model *and* allowed the usage of additional features in the front-end application. Bucketing by both timestamp and geographic area allows for better performance without the dependency on population, and scales the application upwards by allowing for multiple geographic areas to have monitoring based on its unique population and features.

Analysis Methods

The analysis methods used within the project were unsupervised techniques, with both DBSCAN and Neural Networks being used to identify anomalies within the call populations. The team attempted to utilize supervised learning approaches by incorporating labels acquired through boolean search; however, this proved unsuccessful. The approach used for the end application was incremental DBSCAN, which levies the idea of DBSCAN and allows for a cluster to incorporate new streaming data in a way that adds to the previously existing clusters or noise.

Analysis techniques

1. Using Incremental DBSCAN to identify natural grouping in the data, particularly focusing on high density clusters indicative of significant events .
2. Analyzing temporal patterns in the data to understand peak times for emergency calls and the occurrence of anomalies over time.
3. Mapping call data to geographic locations to identify spatial anomalies.

Incremental DBSCAN was chosen for its ability to handle noise and find clusters of varying densities, crucial for detecting significant events in emergency call data. Time series analysis is essential for understanding temporal trends and patterns, aiding in developing models that consider time base variations. Furthermore, geospatial analysis provide insights into the geographic distribution of call, assisting in spatial anomaly detection

Basic Analysis Techniques Used

1. Heatmaps
    a. Streaming the call heatmaps allows for the user to see the population of calls in case of false identification. Without labeled data, the model is at mercy of the dispatcher to be able to identify trends and patterns. For example, if an event is triggered on a highway, the dispatcher is able to use contextual information to indicate that this is likely a traffic accident and will probably require immediate emergency services pending additional information.
2. Alerts
    a. The use of colors (dense populations) and notification alerts allow for the user to pick up on visual cues of events.
3. Toggling for streaming and live traffic feeds
    a. Allowing the user to be able to interact with the map locations, timestamps, and button options increases the usefulness of the application.

The application and analytics were built entirely off of basic techniques, bucketing spatial and temporal densities, cleaning text, and understanding the data serves as the cornerstone of the application.

Analytical workflow

1) Data Ingestion - Importing data from various sources
2) Data Preprocessing - Cleaning, normalizing, and transforming data to ensure quality and consistency.
3) EDA - Identify potential patterns and anomalies
4) Feature Engineering - Creating new features based on insights from EDA
5) Model Selection - Choosing appropriate algorithms for clustering and anomaly detection, focusing on incremental DBSCAN.
6) Visualization - Developing a React-based front end to visualize data changes and provide alert notifications for detected anomalies.

Fig II. Backend to Frontend Development Used: Python, Flask, and React

Processing Environment

1. Local File Systems
2. Python Script for Data Loading and Preprocessing
3. Python Flask for Data Serving
4. React for front-end visualization

Findings and Reporting

Our project focused on developing a real-time anomaly detection system for emergency call data using Incremental DBSCAN. This algorithm allows for dynamic clustering where the clusters adjust as new data is fed in batches. The user can select a timestamp within a date-range, and have the option to stream the data in 10-second batches. The incoming data will then be either added to an existing cluster, create a new cluster, or fall into noise. As the data streams, the additional data is treated the same, therefore if a group is to arise, it will happen as soon as a set of data points become anomolous. The primary goal was to enhance the accuracy and responsiveness of the emergency response system by identifying significant anomalies and high density clusters indicating potential emergencies.

DBSCAN was able to identify event occurrences in a 24 hour period within a city region. Therefore, the incremental DBSCAN in the live application was set to use the same bucketing parameters.

Techniques and Tools Used to Communicate Results

1. Data Visualization : Used charts, graphs, and heat maps to illustrate call density patterns, anomaly detection results, and model performance metrics.
2. Reporting Dashboard : Developed interactive dashboards using React, Connected Django to present real - time data analysis model outputs.
3. Presentation : Created PowerPoints presentation to communicate key insights effectively, combining text and visuals.

Visualization

1. Visual Reports : Included charts and graphs in technical reports
2. User Interface : A user-friendly interface for people to explore findings in real - time, better decision-making during emergency response.

These tools and techniques ensured that our findings were communicated effectively, making them actionable and impactful for improving emergency response and public safety.

Solution Architecture, Performance and Evaluation

Performance

Our models are evaluated for performance based on their ability to accurately identify and categorize anomalies in 911 call data. The Neural network showed results in detecting spatial-temporal anomalies during its initial training phases. The model currently classified all data points as potential events, indicating an overly sensitive detection setting that must be recalibrated to reduce false positives. The recalibration will adjust the threshold for anomaly detection and potential redesigning the networks' final layers to refine output.

The incremental DBSCAN algorithm demonstrates strong performance in identifying high-density clusters corresponding to actual mass casualty events confirmed by external data sources. This validation underscores dbscans' utility in unsupervised settings, where the precise identification of anomaly can direct emergency service more efficiently. Future performance enhancement will focus on optimizing DBSAN's parameter settings through automated tuning techniques, ensuring the model maintains high performance as new data is incorporated. Due to this outperformance, incremental DBSCAN was chosen as the model for the application.

Evaluation

Unfortunately, unsupervised methods rely on varied evaluation methods compared to their supervised counterparts. Due to our structure of detection of event occurrence, and not event subtype, the normal evaluation method, *silhouette score calculation*, was not feasible as a performance measure.

Budget

There was not much of a budget to manage, as we did not employ the use of cloud services for any part of our project. The sensitive nature of our data and no need for real streaming discouraged the use of a cloud environment. Instead, a local storage and local processing environment was utilized throughout.

Conclusion

Through the processing of data, selection of model, and working with data POCs, we were able to develop a full-stack application that allows for the user to stream call data, monitor traffic feeds, and receive alerts at event notification. Our application could be improved with the introduction of labeled data and user feedback on the applications strengths and weaknesses. However, in conclusion, DBSCAN produced the highest quality of isolation for high-density events in populations with additional filtering by location. Incremental DBSCAN was able to be implemented to allow the model to be used in real-time feedback with a streaming data architecture, and provide the user with real-time feedback as calls occur and events arise.

The next steps for this project include the potential to improve model performance by moving toward a supervised methodology, or incorporating additional downstream Computer Aided Dispatch (CAD) data.

References

Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. *Journal of Big Data*, 9(24). https://doi.org/10.1186/s40537-022-00573-8

Bakr, A. M., Ghanem, N. M., & Ismail, M. A. (2015). Efficient incremental density-based algorithm for clustering large datasets. *Ain Shams Engineering Journal*, 6(2), 209-216. https://doi.org/10.1016/j.asej.2015.01.005

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 226-231). https://doi.org/10.5555/3001460.3001507

National 911 Program. (n.d.). *911.gov*. Retrieved June 6, 2024, from https://www.911.gov/

National Emergency Number Association (NENA). (n.d.). *NENA.org*. Retrieved June 6, 2024, from https://www.nena.org/

Federal Communications Commission (FCC). (n.d.). 9-1-1 and E9-1-1 services. Retrieved June 6, 2024, from https://www.fcc.gov/general/9-1-1-and-e9-1-1-services

Appendices

A. DSE MAS Knowledge Applied to the Project

- Data Preprocessing and EDA techniques from Course I
- Machine Learning Techniques (choosing models) from Machine Learning and Statistics for Machine Learning
- Big Data Analytics (cloud systems, live ML models) from Scalable Data Analytics Course
- Postgres processing from Data Management Systems and Beyond DMS

- Project Design and Data Integration from DSE 203
- Visualizations (frontend) from DSE240
- Project Implementation from DSE260

B. Link to the Library Archive for Reproducibility: https://doi.org/10.6075/J0MC907C