Methods and Frameworks for GeoSpatioTemporal Data Analytics

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachgebiet Internet-Technologien und -Systeme
des Hasso-Plattner-Instituts

eingereicht an der
Digital Engineering Fakultät
der Universität Potsdam

vorgelegt von

Aragats Amirkhanyan

Potsdam, 16. Dezember 2019

Für meine Familie

Betreuer:
Prof. Dr. Christoph Meinel

Zweitbetreuer:
Prof. Dr. Andreas Polze

Prüfungskommission:
Prof. Dr. Andreas Polze (Vorsitzender),
Prof. Dr. Christoph Meinel,
Prof. Dr. Robert Hirschfeld,
Prof. Dr. Georg Gartner

Disputation: 16.12.2019

Published online at the
Institutional Repository of the University of Potsdam:
https://doi.org/10.25932/publishup-44168
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-441685

Abstract

In the era of social networks, internet of things and location-based services, many online services produce a huge amount of data that have valuable objective information, such as geographic coordinates and date time. These characteristics (parameters) in the combination with a textual parameter bring the challenge for the discovery of geospatiotemporal knowledge. This challenge requires efficient methods for clustering and pattern mining in spatial, temporal and textual spaces.

In this thesis, we address the challenge of providing methods and frameworks for geospatiotemporal data analytics. As an initial step, we address the challenges of geospatial data processing: data gathering, normalization, geolocation, and storage. That initial step is the basement to tackle the next challenge – geospatial clustering challenge. The first step of this challenge is to design the method for online clustering of georeferenced data. This algorithm can be used as a server-side clustering algorithm for online maps that visualize massive georeferenced data. As the second step, we develop the extension of this method that considers, additionally, the temporal aspect of data. For that, we propose the density and intensity-based geospatiotemporal clustering algorithm with fixed distance and time radius. Each version of the clustering algorithm has its own use case that we show in the thesis.

In the next chapter of the thesis, we look at the spatiotemporal analytics from the perspective of the sequential rule mining challenge. We design and implement the framework that transfers data into textual geospatiotemporal data - data that contain geographic coordinates, time and textual parameters. By this way, we address the challenge of applying pattern/rule mining algorithms in geospatiotemporal space. As the applicable use case study, we propose spatiotemporal crime analytics – discovery spatiotemporal patterns of crimes in publicly available crime data.

The second part of the thesis, we dedicate to the application part and use case studies. We design and implement the application that uses the proposed clustering algorithms to discover knowledge in data. Jointly with the application,

we propose the use case studies for analysis of georeferenced data in terms of situational and public safety awareness.

Zusammenfassung

Heute ist die Zeit der sozialen Netzwerke, des Internets der Dinge und der Standortbezogenen Diensten (Location-Based services). Viele Online-Dienste erzeugen eine riesige Datenmenge, die wertvolle Informationen enthält, wie z. B. geographische Koordinaten und Datum sowie Zeit. Diese Informationen (Parameter) in Kombination mit einem Textparameter stellen die Herausforderung für die Entdeckung von geo-raumzeitlichem (geospatiotemporal) Wissen dar. Diese Herausforderung erfordert effiziente Methoden zum Clustering und Pattern-Mining in räumlichen, zeitlichen und textlichen Aspekten.

In dieser Dissertation stellen wir uns der Herausforderung, Methoden und Frameworks für geo-raumzeitliche Datenanalysen bereitzustellen. Im ersten Schritt gehen wir auf die Herausforderungen der Geodatenverarbeitung ein: Datenerfassung, -Normalisierung, -Ortung und -Speicherung. Dieser Schritt ist der Grundstein für die nächste Herausforderung – das geographische Clustering. Es erfordert das Entwerfen einer Methode für das Online-Clustering georeferenzierter Daten. Dieser Algorithmus kann als Serverseitiger Clustering-Algorithmus für Online-Karten verwendet werden, die massive georeferenzierte Daten visualisieren. Im zweiten Schritt entwickeln wir die Erweiterung dieser Methode, die zusätzlich den zeitlichen Aspekt der Daten berücksichtigt. Dazu schlagen wir den Dichte- und Intensitätsbasierten geo-raumzeitlichen Clustering-Algorithmus mit festem Abstand und Zeitradius vor. Jede Version des Clustering-Algorithmus hat einen eigenen Anwendungsfall, den wir in dieser Doktorarbeit zeigen.

Im nächsten Kapitel dieser Arbeit betrachten wir die raumzeitlich Analyse aus der Perspektive der sequentiellen Regel-Mining-Herausforderung. Wir entwerfen und implementieren ein Framework, das Daten in textliche raumzeitliche Daten umwandelt. Solche Daten enthalten geographische Koordinaten, Zeit und Textparameter. Auf diese Weise stellen wir uns der Herausforderung, Muster- / Regel-Mining-Algorithmen auf geo-raumzeitliche Daten anzuwenden. Als Anwendungsfallstudie schlagen wir raumzeitliche Verbrechensanalysen vor – Entdeckung raumzeitlicher Muster von Verbrechen in öffentlich zugänglichen Datenbanken.

Im zweiten Teil der Arbeit diskutieren wir über die Anwendung und die Fallstudien. Wir entwerfen und implementieren eine Anwendungssoftware, die

die vorgeschlagene Clustering-Algorithmen verwendet, um das Wissen in Daten zu entdecken. Gemeinsam mit der Anwendungssoftware betrachten wir Anwendungsbeispiele für die Analyse georeferenzierter Daten im Hinblick auf das Situationsbewusstsein.

Acknowledgements

First, I would like to express my sincere gratitude to Professor Dr. Christoph Meinel. His innovative attitude and huge research experience were inspiring me during all these last years of pursuing Ph.D. I have learned from him how to overcome difficulty and keep moving on the way of achieving success. He is a motivated leader, who spreads this enthusiasm and passion for work on people around him. And it is a result of success his Ph.D. students and the entire institute.

Then I would like to thank Dr. Feng Cheng, who is the senior researcher in the chair and my direct tutor. He is an experienced researcher with a talent of supervising students to help them achieve success in their research. His support and advice played important role in pursuing Ph.D. He built around him a strong team with talented and motivated people. All these years I was a member of this team and I am thankful for that.

During my stay at the Hasso Plattner Institute, I had a pleasure to meet and work with many talented people, whom I was proud to call colleagues and friends. Thank you for many interesting and productive discussions. Special thanks to Tatiana Gayvoronskaya and Andrey Sapegin.

Another special person I need to thank is Michaela Schmitz, who is our chair secretary. She is a very professional person and the most organized person I have ever met. And I have definitely learned these characteristics that helped me keep working on my research.

Moreover, I would like to express my gratitude to the HPI Research School where I had a chance to work with researchers in different disciplines that gave me new motivation and inspiration for my own research. Another good outcome from the research school is always valuable feedback and discussion about your own research that helped to improve it. I want to thank Andreas Polze, Robert Hirschfeld, and Sabine Wagner for establishing such the unique environment for young researchers.

In the end, I would like to thank my family. I would never achieve any success in my life without their upbringing.

Thank you, HPI, thank you, Potsdam, thank you, Germany.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Listings

Chapter 1

Introduction

Data is the source of information and knowledge. Nowadays, we live in the data-driven decision world model, in which our decision or assumption must be based on data analysis. For data analysis, the research world provides many machine learning and data mining techniques. We use them depending on data and actual goal, in order to find valuable patterns, predict events or explore the knowledge. Combining different types of data and different analytical methods, we try to achieve needed results. If we work with it-security data, we probably want to find anomalies or detect network attacks. If we work with financial data, we probably could be interested in detecting frauds or predicting stock prices. Data mining and machining learning approaches work differently depending on data that we use. Mostly, it is because data are subjectively unique in terms of valuable information.

If we consider data as a sequence of event items, we can consider an event as a piece of valuable information that could contain information answering on following questions: "what?", "who?", "how?", "where?", "when?" and so on. Many of these questions reflect subjective information in data. Only 2 of them reflect objective information: "where?" and "when?". These questions reflect location and time information in a data event. Therefore, those data, which have location and time information, are objectively valuable in terms of data analysis. In this thesis, we work with such data and build research methods, applications and use cases around them.

1.1 Geospatial Data

In our research, we work with data that have location and time information. Such data, we call spatiotemporal or more precisely geospatiotemporal data. The value of such data is in their objective features that can be valuable for data analysis regardless of the subjective purpose of that analysis.

Geographic information systems (GIS) is a big research topic. Such systems are designed around geographic and geospatial data and they provide methods for storage, management, display, and analysis. Traditionally, GIS systems are designed to work with specialized geodata, such as OpenStreetMap, GeoHive, Global Rural-Urban Mapping Project (GRUMP), Global Agriculture Lands and so on. But since many years data that were not considered as geospatial data became such. It happened because IT technologies change continuously processes of data generation and consumption. New technologies, IoT devices, mobile devices, social networks, and the Internet, which connects everything together, made data around us geospatial by default. When we take a photo, write a post in social networks, explore the area on online maps, it produces and consumes different types of geospatial data. Each type of data has its own subjective information regarding the use case, but it has also objective information: time and location. For example, a georeferenced message from social networks could describe something that is happening right now, and this message would contain the concrete location (where?) and time (where?) that supplement the textual information. Another example could be the photo taken by a mobile phone or camera that has a GPS (Global Positioning System) module. The visual image is supplemented by geographic information. Meanwhile, the time information, which says when data were created, is always there.

In our research, we work with data that have geographic coordinates and time information. Also, we work with data that can be normalized to data with geographic coordinates and time. Therefore, in this thesis, we address the challenges of developing and applying methods and frameworks for geospatiotemporal data analytics. Also, we address the challenge of normalization of non-spatial data and transferring them to geospatial or geospatiotemporal data that can bring

more valuable information in the analysis. Around of that, we are interested in concrete applications and use cases, as well.

In different chapters of this thesis depending on the describing research method or framework, we call data as geospatial, georeferenced or geospatiotemporal. If in the scope of the describing method, we work only with geographic information of data, we call data as geospatial or georeferenced. If we consider in our research method also the temporal aspect of data then we call data as geospatiotemporal.

1.2 Research Questions and Contributions

This thesis aims at alleviating several of the challenges associated with utilizing geospatial and geospatiotemporal data. Particularly, we focus on methods and frameworks for clustering and mining geospatiotemporal data. Also, we are interested in using these methods and frameworks in developed applications and considered use cases. In this thesis, we address a number of research questions.

Georeferenced data. How to gather, normalize and store georeferenced data that become suitable for geospatial analytics?

As we already mentioned, we work with georeferenced data but not all of them are presented in the form that is suitable for analysis. Therefore, we address the challenge of transforming data through geolocation, data normalization, entity recognition and building a storage appropriate for working with such data. We demonstrate how we tackle all these challenges in several chapters of this thesis.

Clustering of georeferenced data. How efficiently to cluster georeferenced data in online mode by keeping the center of the cluster in the densest area?

The first research challenge that we address is clustering. We are interested in clustering georeferenced data based on their geographic coordinates. There are many existing methods for clustering from unsupervised machine learning and data mining domains. We consider them as well to tackle the research question. Our focus would be to provide online clustering - clustering that can perform without having the entire dataset at the beginning and that can update its results depending on incoming new data. Also, we try to keep the center of a cluster in the densest area that better reflects cluster information.

Geospatiotemporal Clustering. How efficiently to cluster geospatiotemporal data in online mode by keeping the center in the densest and intensest areas?

Here we try to extend the previous research question for temporal aspect. And we try to meet the same requirements: online clustering, the center in the densest area, but meanwhile accounting the time feature of data and consider it in clustering and detecting the cluster's center. We design and implement the algorithm that afterwards, we apply to the collected dataset, in order to detect anomalies in spatial and temporal perspectives.

Spatiotemporal Sequential Rule Mining. How to design and implement the framework that will allow applying sequential rule/pattern mining algorithms to spatiotemporal data?

Not all data are georeferenced, but they have such information that can be extracted and normalized. In the scope of this research question, we address the challenge, in order to normalize data and make them spatiotemporal. On top of them, we design and implement the framework that allows applying data mining algorithms to spatiotemporal data. The framework is aimed to detect patterns in the combination of their features: location, time and type of the event. Additionally to the framework, we provide the use case for that. For the use case, we work with publicly available crime data and design spatiotemporal crime analytics, in the scope of which we try to detect crime patterns - types of crimes happen close to each other in spatial and temporal perspectives.

Application and Use Cases. How can we design and implement an application that can be used for analysis and visualization of geospatiotemporal data?

We combine previous research contributions into one application that utilize the research methods to provide data analytics and visualization in terms of providing geospatiotemporal situational awareness. Additionally, we address the challenge of making data more valuable for that use case by applying data cleaning and filtering approaches.

Another contribution is one more valuable use case. In the scope of that use case, we collect publicly available data about crime incidents in Berlin, normalize

them and provide public safety awareness information. We address this challenge by applying methods developed in earlier research contributions.

1.3 Outline

The thesis is structured as follows: In Chapter 2, we introduce different concepts, terms, and techniques related to geospatial data. In particular, we detail:

- *Geocoding* - the computational process of transforming an address description to a location - geographic coordinates (the pair of the longitude and latitude coordinates).
- *Geolocation* - the technique for the identification or estimation of the real world geographic location of an object.
- *Data Gathering* - collecting/gathering/fetching needed data from external providers for research purpose.
- *Data Normalization* - the process of transferring data into more structure form suitable for storage and analytics.
- *Storage* - the process of storing normalized data in a suitable database that can be utilized for geospatial analytics.

Based on information and knowledge gained from Chapter 2, we tackle the first research challenge and introduce the method for online clustering georeferenced data in Chapter 3. In that chapter, we provide the pseudo-code of the algorithm, method description and we demonstrate the work on real data with a discussion around the results. Chapter 4 extends the proposed approach from Chapter 3 for temporal aspect. And there, we show the method for clustering geospatiotemporal data. Further, we discuss the benefits and implications of applying these approaches and propose the idea for an additional extension and improvement depending on use cases.

We address the fourth research question in Chapter 5 and illustrate how to transfer data to spatiotemporal data and how to apply sequential rule mining to such data. Such data normalization jointly with adjusting a sequential rule

mining algorithm for spatiotemporal data can bring interesting results that we discuss in that chapter, as well. Also, we present the outcome of that research as the framework for spatiotemporal sequential rule mining.

As the next step of our research and this thesis, we develop a software application that combines the research contributions from previous chapters. We introduce this application and describe its software architecture in Chapter 6. In that chapter, we also give details about the technical implementation and we evaluate the application on a use case and discuss the results of the analysis. Also, we give details about approaches that we use to filter data, in order to make them more valuable for the considering use case.

Chapter 7 provides another use case. We address the challenge of providing public safety awareness based on publicly available data. We show how to achieve this goal by applying methods for data normalization and geolocation. We show explored knowledge and discuss results.

Finally, we summarize this thesis and present an outlook on research issues in Chapter 8. Additionally, we provide directions for future work.

Several of the ideas and findings contained in different parts of this thesis have been published previously:

Chapter 3: Aragats Amirkhanyan, Feng Cheng, and Christoph Meinel. **Real-Time Clustering of Massive Geodata for Online Maps to Improve Visual Analysis**. In *Proceedings of the IEEE 11th International Conference on Innovations in Information Technology (IIT'15, IEEE)* [1].

Chapter 4: Aragats Amirkhanyan and Christoph Meinel. **Density and intensity-based spatiotemporal clustering with fixed distance and time radius**. In *Proceedings of the 16th IFIP Conference on e-Business, e-Services and e-Society (I3E2017, Springer)* [2].

Chapter 5: Aragats Amirkhanyan and Christoph Meinel. **The Framework for Spatiotemporal Sequential Rule Mining: Crime Data Case Study**. In *Proceedings of the 2017 International Conference on Knowledge Engineering and Applications (ICKEA2017, IEEE)* [3].

Chapter 6: Aragats Amirkhanyan and Christoph Meinel. **Visualization and Analysis of Public Social Geodata to Provide Situational Aware-**

ness. In *Proceedings of the 8th International Conference on Advanced Computational Intelligence (ICACI2016, IEEE)* [4].

Aragats Amirkhanyan and Christoph Meinel. **Analysis of the Value of Public Geotagged Data from Twitter from the Perspective of Providing Situational Awareness**. In *Proceedings of the 15th IFIP Conference on e-Business, e-Services and e-Society (I3E2016, Springer)* [5].

Chapter 7: Aragats Amirkhanyan and Christoph Meinel. **Analysis of Data from the Twitter Account of the Berlin Police for Public Safety Awareness**. In *Proceedings of the 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD, IEEE)* [6].

Additionally to that, intermediate results were presented and published as technical reports in the scope of the research group at Hasso Plattner Institute:

Aragats Amirkhanyan. **Towards Analysis of Public Social Data to Improve Situational Awareness**. In *Proceedings of the 9th Ph.D. Retreat of the HPI Research School on Service-Oriented Systems Engineering (Universitätsverlag Potsdam 2016)* [7].

Aragats Amirkhanyan. **One Working Day of the Berlin Police Analysis of Data From the #24hPolizei Twitter Marathon**. In *Proceedings of the 10th Ph.D. Retreat of the HPI Research School on Service-Oriented Systems Engineering (Universitätsverlag Potsdam 2017)* [8].

Some technical reports and papers that were published during the Ph.D. study but not included in this thesis because of a topic difference:

Aragats Amirkhanyan. **Testbed Automation for Network Security and Security Analytics**. In *Proceedings of the 8th Ph.D. Retreat of the HPI Research School on Service-Oriented Systems Engineering (Universitätsverlag Potsdam 2015)* [9].

Aragats Amirkhanyan, Andrey Sapegin, Marian Gawron, Feng Cheng, and Christoph Meinel. **Simulation of User Behavior on A Security Testbed Using User Behavior States Graph**. In *Proceedings of the 8th International Conference on Security of Information and Networks (SIN'15, ACM Press)* [10].

Aragats Amirkhanyan, Andrey Sapegin, Marian Gawron, Feng Cheng, and Christoph Meinel. **Poisson-based Anomaly Detection for Identifying Malicious User Behaviour**. In *Proceedings of the International Conference on Mobile, Secure and Programmable Networking (MSPN'15, Springer)* [11].

Chapter 2

Geospatial Data

In this chapter, we introduce the world of geospatial data and different methods, approaches, and challenges that always come together with them when you work with such data.

2.1 Geocoding

Geocoding is the computational process of transforming a postal address description to a location on the Earth's surface (spatial representation in numerical coordinates - the pair of the latitude and longitude) [12]. *Reverse geocoding*, on the other hand, converts geographic coordinates (latitude and longitude) to a description of a location, usually the name of a place or an addressable location. Geocoding relies on a computer representation of address points, the street/road network, together with postal and administrative boundaries [12].

Geocoding is a subject of interest since the early 1960s [12]. It is a part of Geographic Information System (GIS) and spatial analysis. Nowadays, there are a lot of vendors that fully provide geocoding and reverse geocoding functionality on-premise. They are cloud-based applications with their application programming interface (API). Quite often they are part of the online map services, such as Google Maps, Bing Maps, Yahoo Maps, Here and other. Each service has its own implementation of geocoding. It involves multiple datasets and processes that work together. And each of them is made up of sub-components and sub-operations. We provide the abstract principle of geocoding in the ArcGIS System

[13]. We provide the following information because the general understanding of geocoding would help to work with it and apply to our research methods for analytics.

In Figure 2.1, we presented the schema of the geocoding process that consists of several module-steps. In Figure 2.2, we illustrate the first stage of geocoding - parsing and multiples representation of the address. As an input, we have an original string representation of the address. For example, Friedrichstr. 83, 10117 Berlin Germany. The first step of the process is parsing. On this step, we need to determine address elements, such as street name, house number, index, city state and so on. The result of parsing is the set of address elements. But meanwhile, the address can be parsed in more than one way and we need to find any of these representations of the concrete address. Therefore, multiple representations of the address are the next step. We need it because it would increase the chance of finding the address in the indexed database. The motivation for that is the fact that some address elements can be written in different ways or with an abbreviation. As a result, we have the table presented in Figure 2.2

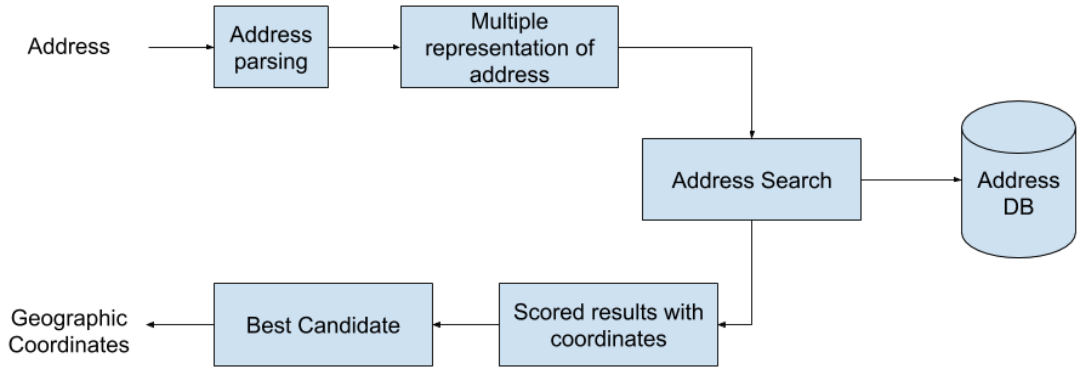
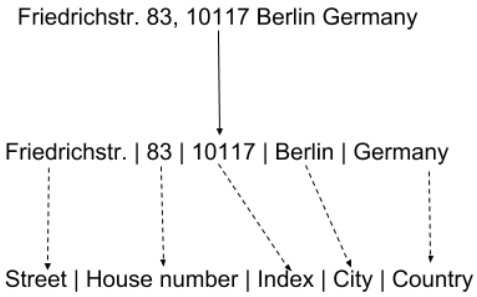


Figure 2.1: Geocoding: schema of the process.

Once we have the parsed address representation, we can search for it in the database. The design of the database structure and indexes are a key aspect of the search performance. For our explanation, it is important only the fact that we are allowed to search by address components and a combination of them. The result of the search is the list of address candidates. If we cannot find address candidates by the full set of criteria, we can reduce the number of criteria and do



Street	House number	Index	City	State	Country
Friedrichstr.	83	10117	Berlin	-	Germany
Friedrichstrasse			BE	Berlin	Deutschland
Friedrichstraße					

Figure 2.2: Geocoding: address parsing.

search again. After we found matched candidates, we need to score the potential candidates. For scoring, we use all address elements. The score shows how found candidates are close to original geocoding address. After that, we need to filter candidates based on the minimal predefined score value. The best candidates are ranked by score and locations with information about latitude and longitude. This information is the final result.

The cloud-based services provide the result of geocoding as a *JSON* [1] (JavaScript Object Notation) or *XML*[2] (eXtensible Markup Language) response with the list of the best candidates. Each entry contains additional geographic related information except for coordinates. Here we show the example of the Google Geocoder request and schema of the response in corresponding Listings 2.1 and 2.1. More information about Google Geocoder, you can find in the Google documentation [14]. Many other cloud-based geocoding services work with similar API (Application Programming Interface) and provide a similar concept of requests and

[1] https://www.json.org
[2] https://en.wikipedia.org/wiki/XML

responses. Therefore, in this section, we considered only one service - Google Geocoder - as an example for explanation.

```
{
 "address": string,
 "location": LatLng,
 "placeId": string,
 "bounds": LatLngBounds,
 "componentRestrictions": GeocoderComponentRestrictions,
 "region": string
}
```

Listing 2.1: Google geocoder request.

```
"results"[]: {
 "types"[]: string,
 "formatted_address": string,
 "address_components"[]: {
   "short_name": string,
   "long_name": string,
   "postcode_localities"[]: string,
   "types[]": string
 },
 "partial_match": boolean,
 "place_id": string,
 "postcode_localities"[]: string,
 "geometry": {
   "location": LatLng,
   "location_type": GeocoderLocationType
   "viewport": LatLngBounds,
   "bounds": LatLngBounds
 }
}
```

Listing 2.2: Google geocoder response.

We described the main principle of geocoding because it is a basement for further research proposals in this thesis. Even separately, geocoding is a powerful tool to support a business decision in a business intelligence platform. The future of geocoding also involves three-dimensional geocoding, indoor geocoding, and multiple language returns for the geocoding platforms.

2.2 Geolocation

Geolocation is the identification or estimation of the real-world geographic location of an object [15]. Depending on an object different methods of geolocation are used. If we consider the mobile devices, the main source of geographic coordinates is a GPS module. If the GPS module is unavailable, the geographic information can be gained from cell towers, which is less accurate as GPS but has improved recent years. WLAN and IP address can be used as well for geolocation of computer devices connected to the Internet.

In our research, we work with different datasets. Some contain concrete geographic coordinates, some contain geographic coordinates of the geographic polygon area. Also, some data do not have coordinates but contain textual or another type of information where information about the location is encoded. Location information is stored in different ways in our data, but we need to transfer it into one common representation - geographic coordinates - the pair of longitude and latitude. Therefore, we face the challenge of data geolocation. The challenge can be solved with the method of normalization, entity recognition, extraction of location-related entries and followed geocoding.

In Figure 2.3, we present the diagram of the geolocation component. The geolocation component is a combination of address normalization and geocoding. We have already discussed geocoding in Section 2.1, therefore, this module is presented in the diagram in simplified form, in order to complete the entire concept. The geolocation component accepts data that have potentially geographic information. Such information could be in the textual feature or geographic feature of data. The textual representation of geographic information is usually the names of streets, cities, the point of interests and so on. The geographic feature could contain a similar set of information but as separated values. We will work with such type of data in Chapter 5.

As an initial step, we process data and extract textual and geographic features. We need to clean the textual information and it means to remove irrelevant words, signs, symbols from the text, for example, conjunctions. After cleaning, we pass the text through the entity recognition module. In our research methods for data processing, we use an entity recognizer based on the NLP library proposed

by Stanford University - Stanford Named Entity Recognizer (NER) [16] [17]. This tool can recognize up to 7 class of entities: Location, Person, Organization, Money, Percent, Date, Time. In our case, we are interested in Location entities. Therefore, we pass the results of NER to the merging module that combines the union address from location entities and geographic features (places, a point of interests and so on). The address representation is the input for the geocoding modules that are responsible for transferring address information into geographic coordinates. We have discussed this module earlier. Here we point out that we use Google Geocoding API [14] jointly with the cache system to reduce the number of requests to Google services. Before to send the request to Google Geocoding API, we check whether there is the mapping between a requested address and geographic coordinates in our cache. If we found one, we use this result. If not then we make a request, save the result of mapping an address to geographic coordinates and process workflow further.

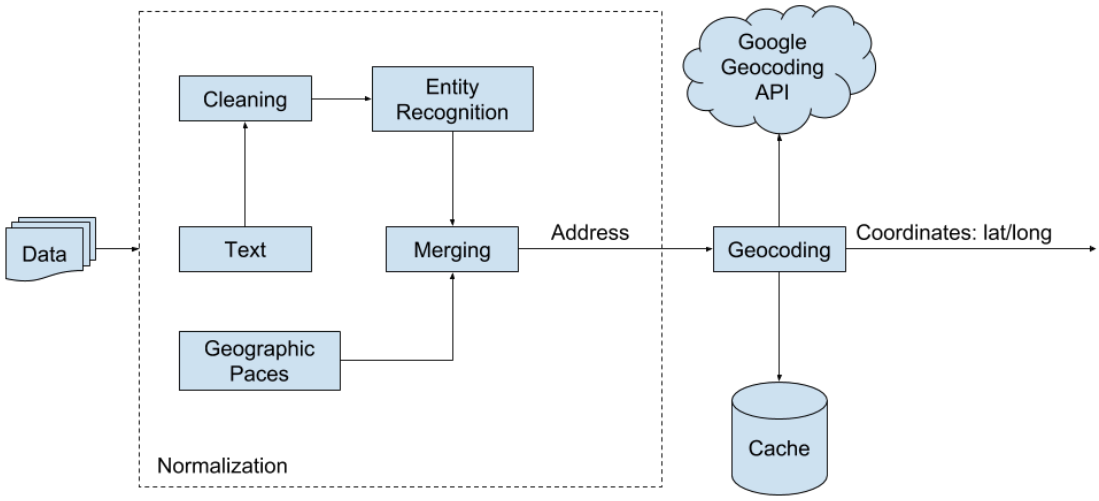


Figure 2.3: Geolocation: schema of the process.

We assign the produced geographic coordinates to the corresponding data. Therefore, as the result, we have georeferenced data - data that have geographic coordinates. Such data is the basement for our research in this thesis.

2.3 Data Gathering

Data is a basement for many kinds of research. We need a sufficient amount of diverse geospatial and geospatiotemporal data. One of the sources of such data can be location-based social networks. Nowadays, social media services produce a huge amount of data and a big part of them is geospatial, such as geotagged images, posts, tweets, videos and so on. We can gather them for our research purpose. Therefore, we use a social network Twitter to gather needed publicly available georeferenced data. It is a quite reasonable decision because, nowadays, Twitter is something more than just a social network. Twitter is the source of world news [18] and it is the place, where breaking news about important events appear firstly. Therefore, we can suppose that if something important happens around some area then tweets about it will almost immediately appear on Twitter. It is important from the perspective of the analysis of situational awareness if we work with geotagged (georeferenced) tweets.

Twitter provides a quite powerful API to fetch required data [19] [20]. One of them is commercial Twitter's Firehose, which guarantees access to 100% of tweets, and another one is public Streaming API, which does not guarantee the percentage of receiving data in real-time. More comparisons between data from Twitter's Streaming API and Twitter's Firehose, you can find in the paper of Fred Morstatter et al. [21]. The main disadvantage of using free API is that Twitter does not guarantee that you will get all requested data. But studies [20] have estimated that using Twitter's Streaming API users can expect to receive anywhere from 1% of the tweets to over 40% of tweets in near real-time. Also, we can increase the percentage of receiving tweets by applying more strict criteria. The criteria can be keywords, usernames, locations, named places, etc. For example, if you ask Twitter to provide only georeferenced tweets from some concrete city then with a high level of likelihood you will get almost all georeferenced tweets from the specified area. In our case, we use location as a criterion to gather data from concrete cities. For our experiments, we collect georeferenced data from the area around London and New York with their neighborhoods. We use these cities because they are the most active Twitter cities [22] and, secondly

because the main language of tweets posting in London and New York is English, which simplifies the content analysis.

The coordinates of monitored areas are {-0.569042, 51.247948; 0.303813, 51.727184} for London and {-74.263876, 40.491435; -73.718404, 40.918649} for New York, where the first pair is latitude and longitude coordinates of the south west and the second pair is latitude and longitude coordinates of the north east. Areas described by coordinates pairs area approximated. For our research experiments and proposals that we present in this thesis, we use data mostly from those specified monitored areas.

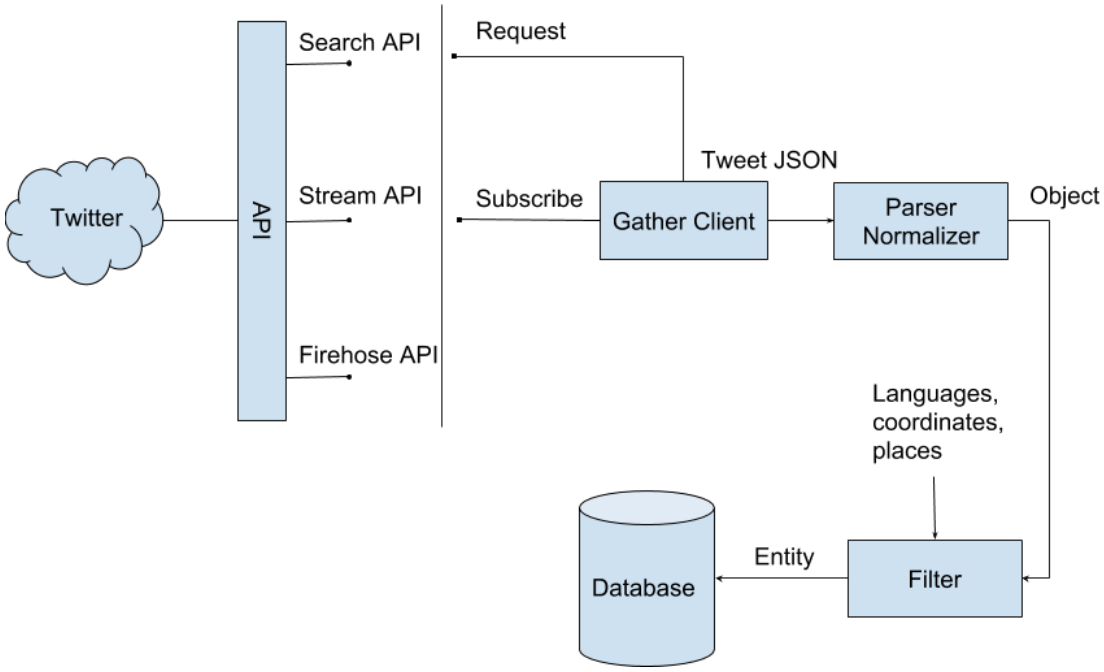


Figure 2.4: Gathering tool.

In Figure2.4, we illustrate the diagram of the gathering tool that we implemented to gather data from Twitter. Twitter is one of the sources of data that we use in our research. As we said before, Twitter provides different APIs to fetch data. In our gatherer, we use Search API and Stream API. With search API, we request historical data based on the request criteria, such as user id, location, hashtag, keyword and so on. With Stream API, we can subscribe from now to future messages. When we subscribe, we need to specify the criteria. Since our focus is on georeferenced data, the main criterion is a location in the form of a pair

of coordinates - latitude and longitude. After we subscribed, we start to receive tweets in JSON format. All tweets have geographic coordinates or a geographic area, such as a geographic polygon or multipolygon. The next would be parsing. The tweet JSON is quite verbose data and we need to parse and transfer it into the object format that will be useful for our research purpose. The UML (Unified Modeling Language) diagram of the class is shown in Figure 2.5. Meanwhile, we can apply some soft filtering, such as filtering by coordinates, languages, places. We apply a filter by coordinates because Twitter returns us messages that are not inside of the specified area but inside of an overlapping area. For example, if we subscribed for London, we can receive messages attached to the entire country - Great Britain. Such messages would not contribute any values into our research proposals, therefore, we filter them out. After all steps of preprocessing, parsing and filtering, we save a formed entity into the database. And further, we would work with a database as a source of data for algorithms. We fetch data from the database, apply the algorithm and save immediately results back to the database. We will describe concrete approaches around georeferenced data in the next chapters.

2.4 Data Normalization

Data normalization is the part of processing georeferenced data. Our gathering tool subscribes for Twitter's stream API and we start to receive tweets messages in the JSON format. The JSON objects are already normalized data that contain a lot of fields, which you can learn on the official website [19] and in Listing 2.4. Tweet messages contain valuable information that can be extracted and normalized, and we need to convert data into the more structured hierarchy of objects that is suitable for programming processing.

```
1 {
2   "created_at": "Thu Apr 06 15:24:15 +0000 2017",
3   "id_str": "850006245121695744",
4   "text": "1\/ Today we\u2019re sharing our vision for the future of the
        Twitter API platform!\nhttps:\/\/t.co\/XweGngmxlP",
5   "user": {
6     "id": 2244994945,
7     "name": "Twitter Dev",
```

```
8       "screen_name": "TwitterDev",
9       "location": "Internet",
10      "url": "https:\/\/dev.twitter.com\/",
11      "description": "Your official source for Twitter Platform news,
            updates & events. Need technical help? Visit https:\/\/
            twittercommunity.com\/ \u2328\ufe0f #TapIntoTwitter"
12    },
13    "place": {
14    },
15    "entities": {
16      "hashtags": [
17      ],
18      "urls": [
19        {
20          "url": "https:\/\/t.co\/XweGngmxlP",
21          "unwound": {
22            "url": "https:\/\/cards.twitter.com\/cards\/18ce53wgo4h\/3xo1c",
23            "title": "Building the Future of the Twitter API Platform"
24          }
25        }
26      ],
27      "user_mentions": [
28      ]
29    }
30  }
```

Listing 2.3: Tweet JSON example.

In Figure 2.5, we show the diagram of normalized Tweet Entity - the entity that we save into the database. We show the diagram of classes with relations to some sub-entities. We quite often will refer to this diagram or its parts in the scope of the thesis in the different corresponding chapters. Here we go through a top view of the diagram and later we explain each important field and how we calculate, extract and normalize them.

The main entity is called *TweetEntity*. Basically, it represents a tweet message that we receive from the Tweeter API. It has original fields, such as *id* (*postId*), *username*, *userId*, *entities*, *inReplyToStatsIdStr* and so on. Information about standard tweet fields, you can find in the official documentation [19]. In this section, we go through the advanced fields that are built by our normalization module.

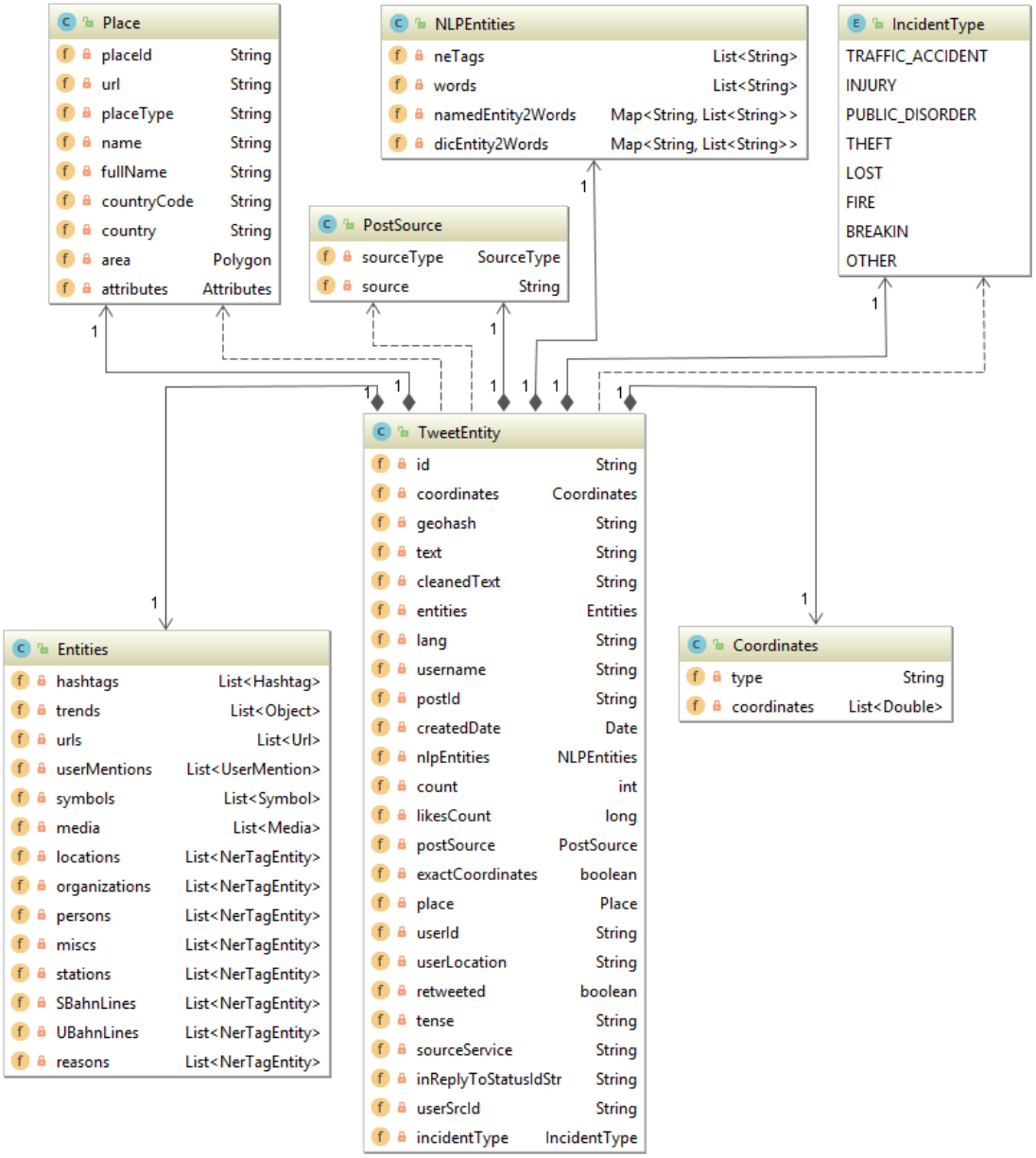


Figure 2.5: Tweet entity with subclasses and relations between them.

Coordinates

The entity *Coordinates* contains two fields *type* and list of coordinates. The first one reflects the type of the geographic feature. In most cases, it is *Point*. The second one is the pair of longitude and latitude coordinates. Some tweets contain coordinates. In such a case, we just take them. But quite often coordinates are

```
1: procedure CALCULATECENTER(polygon)
2:     totalWeight ← 0
3:     x ← 0
4:     y ← 0
5:     z ← 0
6:     markers ← polygon.coordinates
7:     for each m in markers do
8:         lat ← m.lat * Math.PI/180      ▷ Convert latitude from degrees to radians.
9:         lng ← m.lng * Math.PI/180  ▷ Convert longitude from degrees to radians.
10:        dX ← Math.cos(lat) * Math.cos(lng)              ▷ to Cartesian coordinates
11:        dY ← Math.cos(lat) * Math.sin(lng)
12:        dZ ← Math.sin(lat)
13:        dW ← 1
14:        totalWeight ← totalWeight + dW              ▷ Combined total weight for all.
15:        x ← x + dX * dW       ▷ Compute weighted average x, y and z coordinates.
16:        y ← y + dY * dW
17:        z ← z + dZ * dW
18:    end for
19:    x ← x/totalWeight
20:    y ← y/totalWeight
21:    z ← z/totalWeight
22:    lng ← Math.atan2(y, x) ▷ Convert x, y, z coordinate to latitude and longitude.
23:    hyp ← Math.sqrt(x * x + y * y)
24:    lat ← Math.atan2(z, hyp)
25:    marker ← newLatLng(lat * 180/Math.PI, lng * 180/Math.PI)
26:    return marker
27: end procedure
```

Figure 2.6: Calculate the center for the polygon.

missing. Instead of them, tweets contain a *place* entity that describes any point of interest, such restaurant, square, city, stations and so on. Places are described with the area of the *Polygon* type. This information is quite valuable, but we need to have the center coordinates of the polygon that we store in the coordinates field in TweetEntity. For that in the scope of the normalization process, we perform a calculation of the center of the polygon and save the result into *coordinates* field.

The algorithm for calculation of the polygon center is presented in Figure 2.6. The algorithm is implemented based on the specification taken from *GeoMind-Point.com* [23]. The method *calculateCenter* accepts *polygon*, which contains the list of coordinates that describe the area. Firstly, we extract the list of coordi-

nates from the polygon. We call them markers. Each marker contains the pair of latitude and longitude (lat/lng) coordinates. We go through all markers and for each of them, we convert latitude and longitude from degrees to radians and then convert them to Cartesian coordinates. The next operation is a calculation of weights that are used for iterative calculation of x,y,z coordinates. In the end, we finally convert x,y,z back to latitude and longitude coordinates As a result, we return *LatLng* object, which is initialized by coordinates in degrees.

Geohash

Geohash - the string representation of latitude and longitude coordinates. Geohash is a latitude/longitude geocode system invented by Gustavo Niemeyer [24]. It is a hierarchical spatial data structure which subdivides space into buckets of grid shape [24]. We calculate and save geohash values in entities because such representation of coordinates could simplify the geospatial analytic in some cases. Also, we use geohash in the clustering algorithms that we present in Chapter 3 and 4. In Figure 2.7 we show how the geohash function divides the Earth Map on the example of Google Maps.

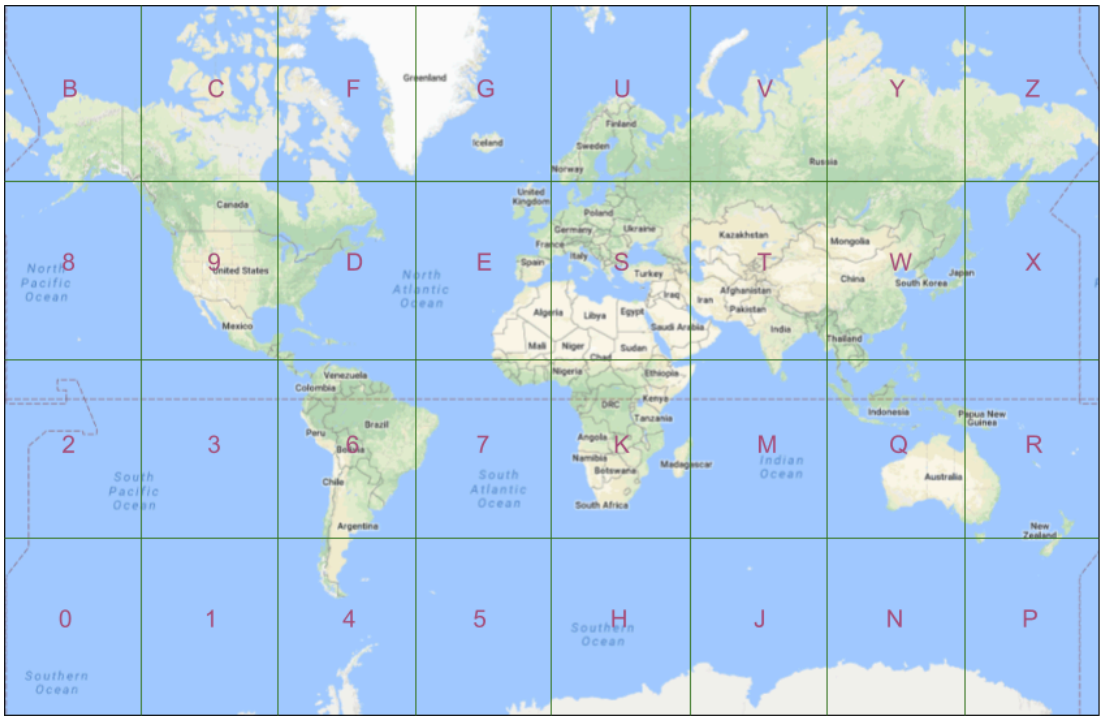


Figure 2.7: Google Maps with geohash grid.

As we said the geohash function subdivides the area into buckets of grid space. It subdivides the area in a hierarchical way depending on the precise, which we want. For example, if our coordinates are located in the square *D* (North America) and we specified the precise as 1, then the geohash value would be just "D". Meanwhile, if we specify the precise 2, it means that we need to subdivide the square **D** one more time and choose the sub-square inside of the **D** square. Therefore, the results would contains 2 characters (*D* + the letter of sub-square), such as *DP*, *DR*, *DX*, *D0*, *D8* and so on. The geohash function allows subdividing area 12 times. Therefore, we can specify the geohash values with 12 characters - 12 precise levels. It allows completely to convert latitude and longitude coordinates into the string representation. Meanwhile, the geohash function is reversible. It means that we can restore complete geographic coordinates from the geohash value. More precise a geohash value, more precise geographic coordinates we can restore.

Geohash function is a very powerful feature of geographic conversion. We use it in our proposed algorithms for clustering as preprocessing step to improve performance. We describe more details about applying the geohash function in clustering in the next chapters.

2.5 Data Storage: Storage for Geospatial and GeoSpatiotemporal Data

After data gathering and normalization, we face the challenge of storing data. For that, we need to choose an appropriate database. A database [25] is an organized collection of data. A database management system (DBMS) is a complex computer-software application that is aimed to provide an organized data. The DBMS tries to meet as many applications as possible. Therefore, we have a lot of databases that are developed for different use cases. In our case, we work with geospatiotemporal data and we need to consider this fact in terms of choosing database storage.

Databases use a special data structure, which is called a database index, to improve the speed of data retrieval operations. Indexes can be created using one or more columns of a table. After each insert operation, a database updates a

table and data structure of the index. When we do a retrieval operation based on columns that are presented in the index, we search through a database index and then fetch data from a table by references.

Working with geospatial and geospatiotemporal data, we face a challenge of storing such data. This challenge is solved in many modern databases by using spatial indexes. One of the most preferable solutions for a spatial index is R-tree. The R-tree was proposed by Antonin Guttman in 1984 [26]. This index is suitable for a common real-world use case, such as to find restaurants within 2 km of the current location or to find all venues in the predefined location area.

In our research, we work with data that contain geographic coordinates and we need to store such data in a database for easier performing queries, analytics, and processing. Therefore, the database jointly with data are the basement for our geospatiotemporal research, analytics, and methods that we present in this thesis in the next chapters. In order to support our research and developments of methods and frameworks, we want to have a possibility to perform a spatial search over datasets. It means that the database should support spatial index for geospatial data.

For your research, we use such the MongoDB database [27]. MongoDB is a NoSQL JSON-document oriented database. In order to store geospatial data, MongoDB supports GeoJSON [28]. The GeoJSON is specified as embedded JSON documents with fields: (1) "type" that specifies the GeoJSON object type and (2) "coordinates" that specify the geographic coordinates. Coordinates are presented as a pair of longitude and latitude coordinates. Valid longitude values are between -180 and 180 degrees, both inclusive. Valid latitude values are between -90 and 90 degrees (both inclusive). According to the documentation [29] of MongoDB, geospatial queries on GeoJSON objects are calculated on a sphere; MongoDB uses the WGS84 [30] reference system for geospatial queries on GeoJSON objects.

The MongoDB database fully suits to out normalized entity presented in Figure 2.5. The big advantage of this database is that MongoDB supports a spatial index. And we use this feature to index our entities by *coordinates* field (look at Figure 2.5). Having the geospatial index over coordinates field, we can perform geospatial operators. Here we present the list of some operators that we use for our research analytics in different use cases.

- $geoIntersects - Selects geometries that intersect with a GeoJSON geometry.
- $geoWithin - Selects geometries within a bounding GeoJSON geometry.
- $near - Returns geospatial objects in proximity to a point.
- $nearSphere - Returns geospatial objects in proximity to a point on a sphere.
- $geometry - Specifies a geometry in GeoJSON format to geospatial query operators.
- $maxDistance - Specifies a maximum distance to limit the results of $near and $nearSphere queries.
- $minDistance - Specifies a minimum distance to limit the results of $near and $nearSphere queries.

2.6 Summary

In this chapter, we introduced the main geospatial concepts around geospatial data. We explained how geocoding works, its purpose and we learned some existing services and their implementations. Geocoding is a basement for geolocation that we described next. We described how we do geolocation for non-geospatial data and which tools and approaches we use for that. Then we dived deeper into data. Firstly, we explained how and where we collect data for research. Most importantly, what type of data we most are interested in. Data brought a new challenge - data normalization that we discussed in the next section.

We learned the Twitter example that helped us to show the process of data gathering and data normalization. We showed some concrete examples of field normalization and entity extraction and recognition. At the end of the chapter, we discussed a storage for geospatiotemporal data with a focus on existing geospatial databases that support a geospatial index.

This chapter contributes into understanding the domain and challenges in working, processing, normalization and storing geospatial data. The next chapters are devoted to more concrete methods and frameworks for geospatiotemporal data analytics. They contain data clustering in different aspects (spatial and

temporal), spatiotemporal pattern mining, data analytics, application implementation, and different use cases.

Chapter 3

Clustering of Georeferenced Data

In this chapter, we describe our proposed method for clustering geospatial/georeferenced data. The basement for this research contribution, we prepared in the previous chapter, where we addressed challenges around gathering, normalization, geolocation and storage of geospatial data. We start the chapter with the section about related work (Section 3.1), continue with the description of our proposed method (Section 3.2) and conclude with the summary (Section 3.3).

3.1 Related Work

When we work with georeferenced data, it quite often means working with online maps. Online maps are knowledge base for mapping and visualization of georeferenced data. They help understand data and do analytics. If you want to map a huge amount of data on the online maps, you have to deal with clustering algorithms. There are many client side and server side implementations of geo (geographic) clustering algorithms for online maps. We have a look at them in Section 3.1.1 and 3.1.2.

Clustering is a big topic of research in unsupervised machine learning area. Therefore, some of them can be applicable for clustering georeferenced data. Especially, we are interested in spatial and density-based clustering algorithms. We have a look at some of them in Section 3.1.3 and 3.1.4.

3.1.1 Grid-based Geo Clustering

One of the common approaches for clustering georeferenced data for online maps is grid-based clustering [31]. This approach uses the concept of division of geographic maps into the grid (set of cells) of similar size and grouping geographic points inside each cell into one cluster. To facilitate such concept, many implementations of grid-based clustering use the geohash procedure. *Geohash* is a latitude/longitude geocode system invented by Gustavo Niemeyer [24]. It is a hierarchical spatial data structure which subdivides space into buckets of grid shape [24]. Since the geohash function changes the representation of geographic coordinates, it can be a part of the data normalization process, as we described in Section 2.4.

The grid-based clustering works in the simple principle of dividing the map into squares of a certain size and then grouping markers inside of such squares [31]. Geohash can be presented as a string with 1 up to 12 characters. The number of characters reflects the accuracy of representation of coordinates. Each new character in the geohash string is one more division of grids. The geohash function is reversible. It means that you can extract latitude and longitude coordinates from the geohash value. The level of recovery of coordinates depends on the length of geohash value. If it contains all 12 characters then we can fully recover latitude and longitude coordinates. The fewer characters in the geohash value, the less accurate latitude and longitude. Generally, it means that you can specify how accurate you want to determine the location in both representations of coordinates.

Keeping in mind the concept of geohash function, we can apply it for grid-based clustering approach. So, grid-based clustering uses different levels of the precision of the geohash value to a group near located geographic points regarding a zoom level. In Figure 3.1 we show how gird-based clustering looks on the online map. We have a map that we divide on grids depending on a zoom level. Then we simply group markers that are inside of each grid-cell. You can see cells with their markers and the number in the center of the cell that shows how many markers are inside of that cell. The center of the cluster becomes the center of the grid cell.

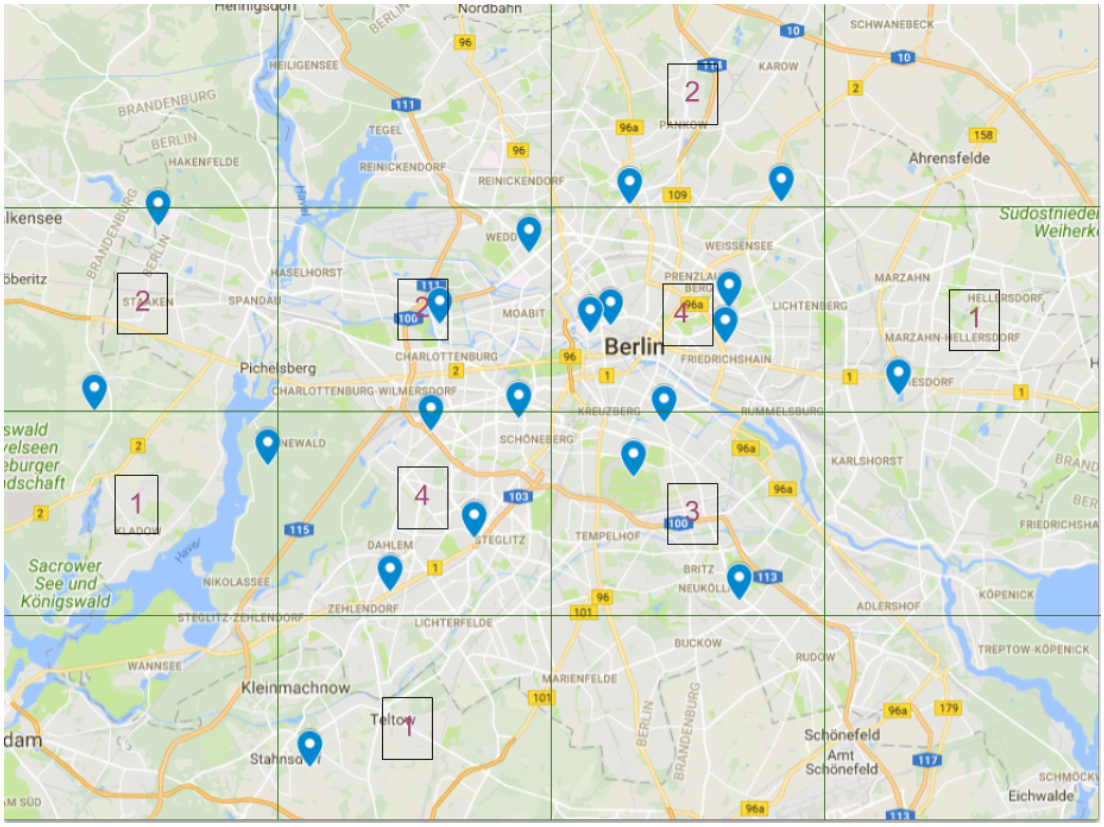


Figure 3.1: Grid-based clustering illustration.

The main advantage of such approach is the simplicity and time efficiency. By this approach, you avoid the calculation of the distance between geographic coordinates and work only with hash values that can be easily grouped by using efficiently designed hash tables. But this approach has a disadvantage. The main disadvantage of the grid-based clustering is that it is not accurate in the case when geographic points are located close to the border of grids. Also, the center of the cluster is not in the center of the densest area.

3.1.2 Distance-based Geo Clustering

The distance-based clustering is similar to the grid-based clustering. But instead of the geohash principle, the distance-based clusters are created based on the distance between the marker and the center of the cluster. The center of the

cluster is specified algorithmically through iteration of the existing markers [31]. The distance-based clustering can provide little bit better accuracy for clustering of markers. But it still has the problem that the center of the cluster could be not in the densest area.

You can see the illustration of the distance-based clustering in Figure 3.2. Markers are included in the circle, to which they are close by the distance measurement. Numbers inside circles reflect the number of markers inside of the cluster.

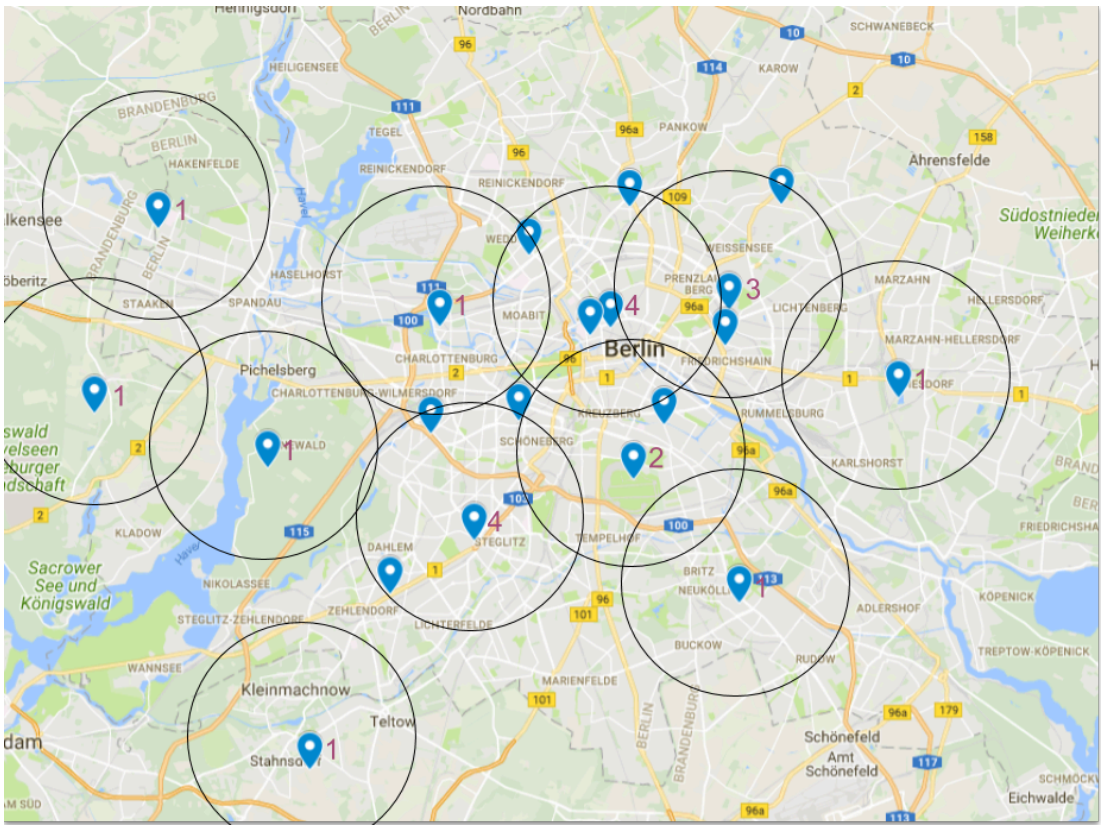


Figure 3.2: Distance-based clustering illustration.

3.1.3 Density-based Clustering

Clustering is a well-known topic from the unsupervised machine learning area and there are many existing clustering algorithms. But we are interested in the algo-

rithms that are suitable for clustering georeferenced data. M.Parimala et al. [32] provided a survey on density-based clustering algorithms for mining large spatial databases. The algorithm can be distinguished in following types: (1) connectivity based clustering (hierarchical clustering), (2) centroid-based clustering, (3) distribution-based clustering and (4) density-based clustering.

The most popular solutions for density-based clustering are DBSCAN [33], OPTICS [34] and many of their modifications. There are many papers that use density-based clustering to cluster georeferenced data for different purposes. For example, Tamura, K. et al. [35] and Sakai et al. [36] presented in their papers how they used a density-based spatial clustering algorithm for extracting attractive local regions in georeferenced documents. Also, Kisilevich et al. [37] developed P-DBSCAN for exploration and analysis of attractive areas using collections of geotagged photos.

3.1.4 Spatial Clustering

Spatial clustering is a general name for a clustering algorithm that considers spatial aspect as a feature for data clustering. All defined types of clustering georeferenced data can be considered as spatial clustering, because they use geographic information, which is a spatial feature, to group data. In this subsection, we would like to go through some survey of existing spatial clustering algorithm to complete related work and find additional insights that could help in our research contribution.

Every year, the research world presents the survey papers around spatial clustering algorithms [38] [39] [40]. Such papers give a general overview of the domain from different perspectives. Neethu et al. [39] 2013 distinguish spatial clustering algorithms in 4 categories: hierarchical, partitional, density-based, grid-based. They discussed existing solutions for each category. We have touched the last two type of clustering algorithm in this chapter, as well. Varghese et al. used a similar approach for the survey in 2015. Meanwhile, Qiliang Liu et al. [40] in 2012 used another approach for the survey. They investigated existing clustering algorithm from data mining domain and discussed them in terms of several capabilities, such as (1) discovery of arbitrary shaped clusters, (2) discovery of clusters

with uneven density, (3) robust to noise, (4) heavily reliant on prior knowledge, (5) consideration of both spatial proximity and attribute similarity.

All these surveys and description of clustering algorithms of georeferenced data for online maps from previous subsections would help us to propose our approach for clustering georeferenced data in the next section.

3.2 Online Clustering of Georeferenced Data for Online Maps

When you work with a big amount of georeferenced data and you want to display them all together on online maps, you face the challenge of data clustering. Depending on the purpose of the clustering, results, which you want to find out, you define your own requirements for the clustering algorithm. If one of the existing clustering solutions meets your requirements, you can easily use one of them.

For our research purpose, we also define the requirements with provided motivation.

- **Online algorithm.**[1] It should be able to process its input piece-by-piece in a serial fashion without having the entire input available from the start.
  Motivation: Many sources of georeferenced data are streaming based. Therefore, the big advantage is to be able to work with continues incoming data and providing processing, clustering and analytics results at any time.
- **Distance-based clustering.** The close distance should be the fact for a decision of grouping georeference points.
  Motivation: Since we work with spatial data, we need to define the metric for grouping closed located geographic points. The geographic distance is a reasonable feature because of the natural aspect. It prevents the case when close geographic points are assigned to different clusters, which we have in the grid-based clustering algorithm.
- **Cluster's center is the density-based center.** The center of the cluster must be in the densest area.
  Motivation: From the perspective of geospatial analytics, the densest area of the distribution of geographic coordinates is the most interesting area in

[1] https://en.wikipedia.org/wiki/Online_algorithm

terms of exploring knowledge and anomalies. Therefore, we are interested in clustering data around such dense areas.

- **Map-Reduce clustering.** The algorithm should be able to work as a map-reduce algorithm. *Motivation*: In the era of distributed computation, the one major requirement to algorithm is to be able to work in the disturbed/parallel mode. Map-reduce is one of such paradigms that we want to fulfill.

In our research approach, we try to combine the advantages of presented in previous section geospatial clustering algorithms jointly with defined above requirements. Further, in this section, we present our proposal for geo clustering algorithm with a full description of all aspects of the method jointly with pseudo-code and experiments.

3.2.1 Data

To design, develop and test our clustering approach, we collected geospatial data from Twitter. For that, we used the designed gathering tool presented in Section 2.3. We collected data from London from the area defined by following coordinates - {-0.569042, 51.247948; 0.303813, 51.727184}, where the first pair is the longitude and latitude coordinates of the south-west corner of the area and the second pair is the coordinates of the north-east corner, correspondingly. Totally, we collected around about 1 million (1021054) Twitter's posts that contain geographic coordinates.

3.2.2 Method

We start the section with an overview of entities, which we use in our approach. You can find them in Figure 3.3. The first entity is *LatLng*, which contains latitude and longitude coordinates. It presents geographic coordinates of points (markers). The second one is the *Marker* entity that contains the position field (*LatLng*) and the list of markers. This list of markers is the result of grouping markers with similar coordinates based on geohash preprocessing, which will be introduced in this section later. The meaning of *Marker* is the point on the online map that we are going to cluster. And since we are going to cluster markers, we

should have the entity that reflects the result of clustering. And for that, we have the last entity called *Cluster*. It contains the following fields: the center (*LatLng*), the list of markers and the radius of the cluster.

Figure 3.3: Diagram of entities.

Once we introduced entities, we can describe the methods for clustering geospatial data. Our approach is the grid and density-based geo clustering that clusters data in two steps: (1) geohash preprocessing and (2) real-time density-based geo clustering.

The first step is quite simple and can work piece-by-piece of data. It takes coordinates of the points, calculates the geohash values with the specified precision and groups them according to the same geohash value. By this step, we reduce the number of points for further clustering. The key point is which level of the precision to use. As we mentioned in Section 3.1.1, the grid-based clustering suffers from bad clustering of points that are located near to the border of grid cells. Due to this issue, we should not fully delegate clustering to the geohash preprocessing. For example, if we want to cluster geospatial data with a zoom level equaled 10, then probably the best precision will be 6 (these recommended values come from experiments (Section 3.2.4) and your desire of accuracy). Using these recommendations and applying geohash preprocessing, we reduce the number of points by grouping very close points, which will be grouped into one cluster in any distance-based clustering algorithm. By other words, if our cluster has diameter 100 km, then the points in the distance of 2 meters with a high likelihood will be in the same cluster. And we can group them into the one marker and count them as one marker with weight 2 after geohash preprocessing. It also explains why the *Marker* entity in our diagram in Figure 3.3 contains the list of other *Markers*.

Figure 3.4 shows the simplified view of how geo hashing preprocessing workflow works. The principle is quite simple. As we mentioned, we need the geohash preprocessing to reduce the number of points (markers) by grouping very close points. Generally, we could say that we pass p points, and as a result, we have v geohash values, where v is usually less than p, and how v is less p depends on the precision of the geohash function we chose. Thus, Figure 3.4 shows that we pass coordinates (latitude and longitude) and the precision equaled 4 (the range is from 1 to 12) and based on input parameters, we calculate the hash values. All values have the size equaled 4. And we can see that the pairs of coordinates (43.11, -44.21) and (43.21, - 44.11) have the same geohash value with the precision 4 that equals *ep8t*. If we would have the precision greater than 4 then the considered pairs of coordinates could produce different hash values. For example, with the precision equaled 7, the first pair of coordinates produces the value *ep8t1zz* and the second produces *ep8tu60*.

Based on our experiments, we determined the best recommendations for the precision of the geohash function depending on the zoom level. The result is presented in Table 3.1, which will be described in Section 3.2.4. We need to mention one more time that we use the geohash preprocessing not to fully cluster data, but to reduce the number of points, that will be transferred further to the density-based geo clustering.

In Figure 3.5, we illustrate the initial step of clustering - geohash preprocessing, which is a kind of grid-based clustering. The task of the clustering algorithm is to cluster georeferenced data that are very close to each other. Georeferenced data are presented in the figure as red and green points. As we said earlier, grid-based clustering is based on the geohash function that divides a map into cells of the predefined size depending on accuracy. After division, we group red points that are close enough to each other, by other words, points that are in the same cell. We show the grouped points as green points in the figure. The output of this preprocessing clustering is input for the next step of the clustering - density-based clustering.

The next step of clustering is the density-based geo clustering, which is presented in Figure 3.6. As input data for the algorithm, we use points after the geohash preprocessing. Actually, it is not obligatory to use preprocessed data,

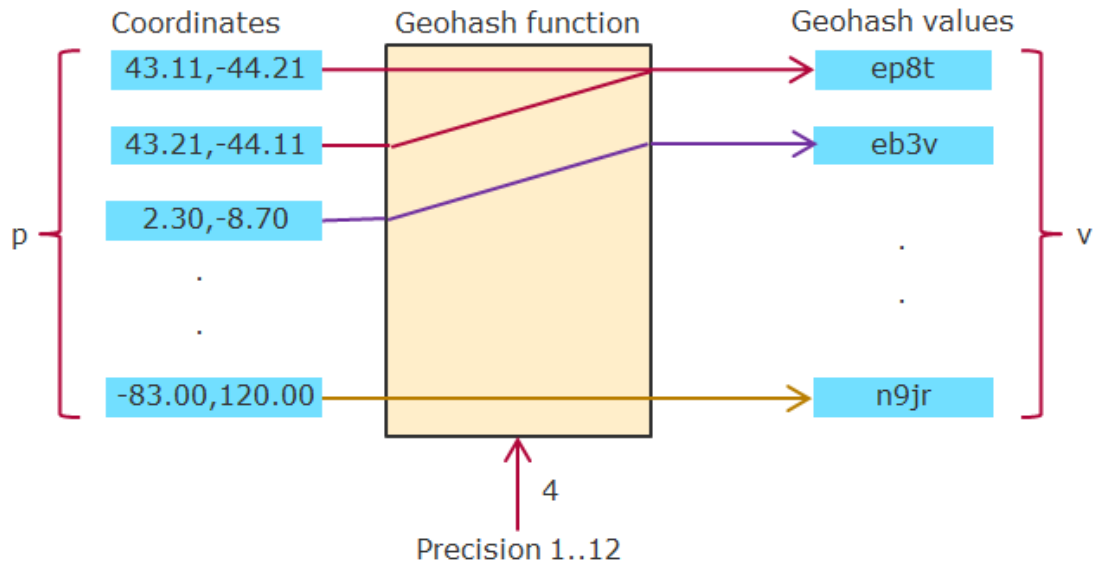


Figure 3.4: Illustration of the geohash preprocessing.

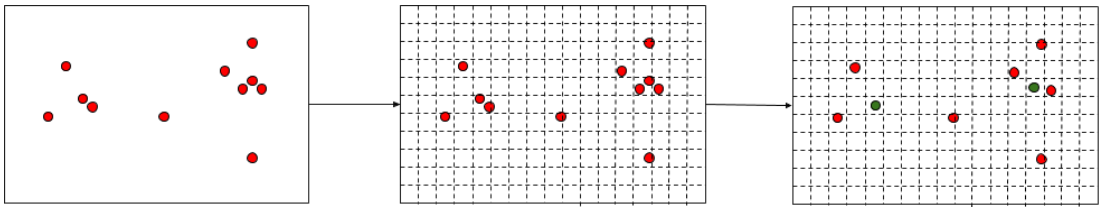


Figure 3.5: Illustration of the grid-based clustering.

but on another hand, it significantly improves the performance since the number of points is reduced.

We announced that our algorithm is an online algorithm. And it means that it can process its input data piece-by-piece in a serial mode and it does not need the entire data to start the process. We could pass points by batches and have the result at any time. Just one assumption is that, in order to cluster new batch of points, the algorithm should know about existing clusters. In details, it means that if we have the first batch with 1000 points, and as a result, we have 10 clusters. Then if we want to cluster new batch with 2000 points, we need to say to the algorithm that we have already existing clusters and use them to group points from a new batch. But if existing clusters do not cover the area of

new points (some points are out of the clusters' range), the algorithm can create additional clusters. And after applying the second batch, we could have 10 or more clusters, but not less.

Figure 3.6 shows the simplified view of the density-based geo clustering. The figure shows that the algorithm takes the stream of points as the input parameter, and as a result, it produces the list of clusters. If we have n points in input then we have m clusters in output. The results of clustering can be used at any time, but if we have new incoming n points we must pass clusters to the input of the algorithm to adjust them. It is showed by the arrow from clusters to the *Cluster* box. Also, it is important to mention that after clustering n points, it could be that maximum $n/3$ points will be pulled from the clusters. It means that these points must be passed again to the input of the algorithm, and by this way, again adjust already created clusters.

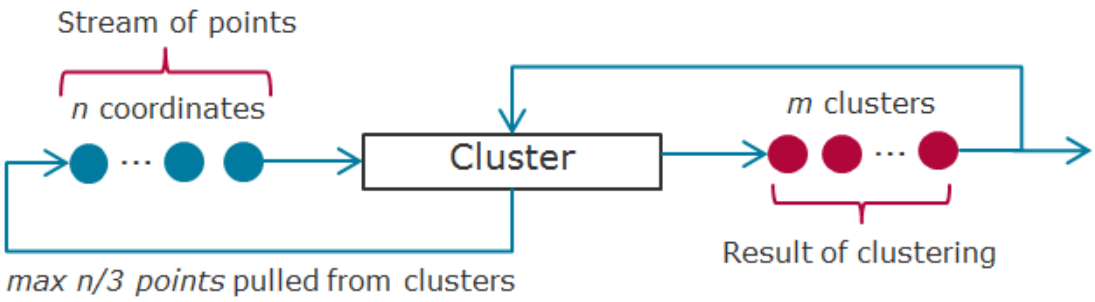


Figure 3.6: Schema of the density-based geo clustering.

In Figure 3.7, you can find the more detailed step-by-step illustration of how our density-based geo clustering algorithm works. The cross reflects the center of the cluster (centroid), blue points reflect clustered points (points belong to the cluster) and red points reflect non-clustered points (points do not belong to any cluster). On the first frame, we start with the state when 2 blue points are grouped into the one cluster. We see that inside of the created cluster there are three red points, which are not clustered. We assume that these points are close to our cluster and they must be added to it. When we do it, we have 5 blue points and we recalculate the centroid. The result, you can see in the second frame. But after recalculating the centroid, the centroid moves to the more dense area. By this way, we got that new 5 red points are inside of the cluster. So,

it means that we must cluster them and we do it. And as a result, we got new cluster with the new centroid illustrated in the third frame. We remind that blue points reflect clustered points and red points reflect non-clustered points. And we want to notice that the point marked with *, which was blue (clustered) on the first and on the second frames, became the red point (non-clustered) on the third frame. It is explained, that after changing the centroid, the cluster moved and the * point became out of the cluster's range and it affected that the * point was pulled from the cluster and marked with red color.

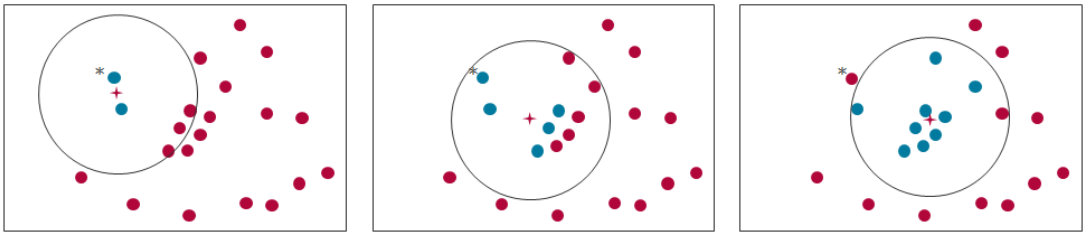


Figure 3.7: Illustration of process steps in the density-based geo clustering.

We combine two previous figures: (1) the schema of the density-based clustering (Figure 3.6) and (2) the detailed illustration of geo clustering (Figure 3.7). The result of combination, we present in Figure 3.8, in order to have summarised view on the clustering algorithm.

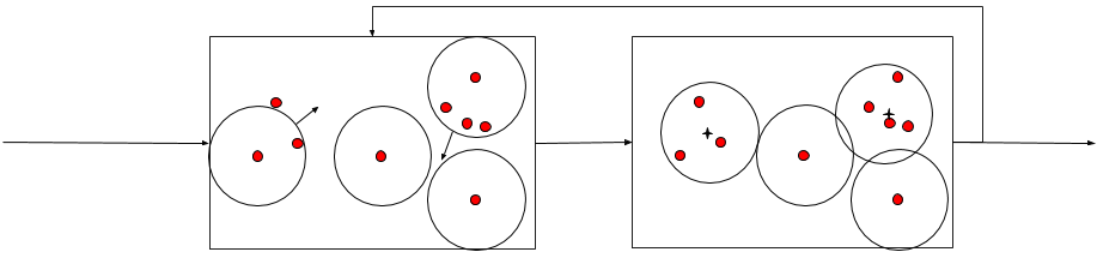


Figure 3.8: Illustration of the density-based clustering.

3.2.3 Algorithm

This section is dedicated to the analysis of the algorithms that are used in our approach of geo clustering. As we mentioned in Section 3.2.2, our approach contains two steps. Therefore, in this section, we consider the algorithm for

geohash preprocessing and the algorithm for our density-based geo clustering. But before to start, we need to draw your attention that in implementations of the algorithms, we use some well-known supporting algorithms, such as (1) the algorithm for calculation of the distance between two geographic points [41] and (2) the algorithm for calculating the geographic center [23]. Here, we present these algorithms in simplified pseudo-code: Figure 3.9 and Figure 3.10. Further in the text, we use the name of the methods to refer to them: *distance()* for the first algorithm and *center()* for the second .

In Figure 3.9, we present the algorithm for calculation of a distance between two geographic points. The method uses the "haversine" formula to calculate the great-circle distance between two points – that is, the shortest distance over the Earth's surface – giving an "as-the-crow-flies" distance between the points (ignoring any hills they fly over) [41].

The method accepts 2 geographic points *p1* and *p2*. Firstly, we need to define the radius of the Earth, which is approximately 6371 km. Then we calculate the delta between latitude and longitude coordinates that are *dLat* and *dLng*. The coordinates values are converted to radians by multiplication on *Math.PI/180*, where *Math.PI* is a mathematical constant originally defined as the ratio of a circle's circumference to its diameter and approximately equaled to 3.14159. The next steps in the method are taken from "haversine" formula for calculation of the great-circle distance between two points. After that, we multiply the **c** value with the Earth radius in order to get the distance in kilometers. We can easily to transfer kilometers to meters by multiplication on 1000 (1 kilometer = 1000 meters). And we do that on the last step before to return the final result.

The second supporting algorithm is presented in Figure 3.10. This algorithm is responsible for the calculation of the geographic center among geographic points. We adjust the original algorithm, to make it more efficient in terms of repeated recalculation of the center based on incoming points. Such improvements come from our requirements to our geo clustering algorithm. The method *center()* accepts two points: c - center geographic point and p - new incoming geographic point. Additionally, the method accepts the parameters s - the size of the cluster. This parameter reflects how many points were used to calculate the current geographic center. Such additional information simplifies the recalculation of the

1: **procedure** DISTANCE($p1, p2$)
2: $R \leftarrow 6371$ ▷ Radius of the Earth in km
3: $dLat \leftarrow (p2.lat - p1.lat) * Math.PI/180$
4: $dLng \leftarrow (p2.lng - p1.lng) * Math.PI/180$
5: $a \leftarrow (Math.sin(dLat/2)) * Math.sin(dLat/2) + Math.cos(p1.lat * Math.PI/180) * Math.cos(p2.lat * Math.PI/180) * Math.sin(dLng/2) * Math.sin(dLng/2)$
6: $c \leftarrow 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a))$
7: $d \leftarrow R * c$
8: $d \leftarrow d * 1000$ ▷ Distance in meters
9: **return** d
10: **end procedure**

Figure 3.9: Calculation the distance between two geographic points.

cluster's center. To calculate the initial cluster's center, we need to have 2 points. In such case, we will have c as one point, p as a seconds point and s as equaled 1. After calculation of the center, we would have a new center and new size of the cluster. Both these new results will be as an input for the algorithm in the next iteration. Therefore, s for the next iteration would be 2 then 3 and so on. It makes the algorithm iterative and we can easily recalculate the center based on incoming new points.

Further, in this chapter and the next chapter, we refer to these supporting algorithms by using the name of the methods: *distance()* for the first algorithm and *center()* for the second one.

The first step of the geo clustering algorithm is preprocessing. Figure 3.11 reflects the algorithm of the geohash preprocessing that we use to group points, which are very close to each other. Firstly, we create the *map* variable, where we will store the result of preprocessing. The algorithm starts with the loop over the markers. At each iteration, we calculate the geohash value based on the *precision* and save it into the *hash* variable. Then we need to retrieve the existing *value* from the *map* based on the hash key. The type of the *value* variable is *Marker*. If the *value* is *null*, we create a new one and set the position equaled to the position of the considering marker. After that, we add our marker to the list of markers in the *value* variable and put the result back to the *map*. After completion of the algorithm, we have the *map* variable that contains the m key-value markers,

```
1: procedure CENTER(c, p, s)          ▷ c - center, p - new point, s - size of cluster
2:     lat1 ← c.lat * Math.PI/180
3:     lng1 ← c.lng * Math.PI/180
4:     dX1 ← Math.cos(lat1) * Math.cos(lng1)
5:     dY1 ← Math.cos(lat1) * Math.sin(lng1)
6:     dZ1 ← Math.sin(lat1)
7:     lat2 ← p.lat * Math.PI/180
8:     lng2 ← p.lng * Math.PI/180
9:     dX2 ← Math.cos(lat2) * Math.cos(lng2)
10:    dY2 ← Math.cos(lat2) * Math.sin(lng2)
11:    dZ2 ← Math.sin(lat2)
12:    x ← (dX1 * s + dX2)/(s + 1)
13:    y ← (dY1 * s + dY2)/(s + 1)
14:    z ← (dZ1 * s + dZ2)/(s + 1)
15:    lng ← Math.atan2(y, x)
16:    hyp ← Math.sqrt(x * x + y * y)
17:    lat ← Math.atan2(z, hyp)
18:    latLng ← newLatLng(lat * 180/Math.PI, lng * 180/Math.PI)
19:    return latLng
20: end procedure
```

Figure 3.10: Calculation of the geographic center.

which size is usually less than the size of the passed markers. It is so if the original markers contained close points and we chose the appropriate precision. The algorithm shows what was illustrated in Figure 3.4.

The second step of our approach, we present by two algorithms to make the explanation clearer. The first one is the method for adding a new marker to the cluster, which is presented as the simplified pseudo code in Figure 3.12. Firstly, we retrieve some variables from the cluster that we will use in the algorithm. Then we start with checking, whether the center of the cluster is *null*. If the center is *null* and the passed marker is the first marker in the cluster, then we set the center of the cluster as a position of the marker. If the passed marker is not the first, then we need to calculate a new center based on the new marker and existing markers. For that, we invoke the method *center()* and pass the center's position of the cluster, the marker's position and the size of the markers list. After that, we need to add the marker to the list of the markers.

The algorithm in Figure 3.13 is the second part of our approach for geospatial

```
1: procedure buildGeoHashMap(markers, precision)
2:     map ← {}
3:     for each m in markers do
4:         hash ← geohash(m.lat, m.lng, precision)
5:         value ← map[hash]
6:         if value == null then
7:             value ← newMarker()
8:             value.position ← marker.position
9:         end if
10:        value.markers.add(m)
11:        map[hash] ← value
12:    end for
13:    return map
14: end procedure
```

Figure 3.11: Algorithm of the geohash preprocessing.

```
1: procedure addMarker(cluster, mark)
2:     center ← cluster.center
3:     markers ← cluster.markers
4:     size ← cluster.markers.size
5:     if center == null then
6:         center ← mark.position
7:     else
8:         center ← center(center, mark.position, size)
9:     end if
10:    markers.add(mark)
11: end procedure
```

Figure 3.12: Assigning the marker to the cluster.

clustering. In this algorithm, we go through markers and try to assign them to existing clusters based on which cluster is the closest to the marker. We start with assigning the *cluster* variable as *null* and assigning the *distance* variable as the *radius*. Then we iterate by existing clusters, calculate the distance between the center of the cluster and the position of the marker. For calculation the distance between points, we use the method *distance()*, which was mentioned as the well-known supporting algorithm at the beginning of the section. Calculating the distance between the marker's position and the center of the cluster gives us a possibility to find the minimal distance, by other words, the closest cluster. If we did not find such cluster, we create a new one and add it to the list of the

```
1: procedure CLUSTER(markers, clusters, radius)
2:     for each m in markers do
3:         cluster ← null
4:         distance ← radius
5:         for each c in clusters do
6:             d ← distance(c.center, m.position)
7:             if d <= distance) then
8:                 distance ← d
9:                 cluster ← c
10:            end if
11:        end for
12:        if cluster = null then
13:            cluster ← newCluster()
14:            clusters.add(cluster)
15:        end if
16:        addMarker(cluster, m)
17:    end for
18: end procedure
```

Figure 3.13: Density-based geospatial clustering.

clusters. Since now the *cluster* is not *null*, we can add the marker to the cluster.

As we remember, after adding a new marker to the cluster, the center of the cluster is adjusted and we could have some markers, which are out of the cluster's area. We should find them and pull from the clusters. This process is quite simple. We just go through all markers of all clusters and check the distance between the marker and the center of the cluster. If the distance is more than the radius, then we need to pull this marker and recalculate the center. As a result, we would have the list of pulled markers that will be passed again to the input of the density-based geo clustering algorithm to adjust just created clusters.

3.2.4 Experiments and Evaluation

To prove the concept and evaluate it, we developed the service for clustering georeferenced data. The application is written in Java with some additional frameworks. To store geospatial data, we used MongoDB, because this database provides geospatial index and suitable to store geospatial (The motivation and requirements for storing geospatial data, we have discussed in Section 2.5). Data come from Twitter, and for experiments, we collected about 1 million (1021054)

Twitter's posts with information about their location. The result of clustering is displayed on the OpenStreetMap[1] by using the *LeafletJS*[2] library. The view of the service with clustered markers in the Berlin area is illustrated in Figure 3.14.

Figure 3.14: OpenStreetMap with clustered markers.

Once we implemented the service based on our approach, we made experiments of clustering dataset with different configurations of the zoom level and geohash precision. To run tests, we used the computer with the following characteristics: Intel(R) Core i5-2400 CPU @ 3.10Ghz and 12,0 GB of RAM. And for all experiments, we used the entire dataset (about 1 million markers) and the radius equaled 40 pixels. The real radius in meters is calculated based on the zoom level and meters/pixel according to the Table 3.1.

Table 3.1 shows the correspondence between the zoom level and geohash precision, which we found as optimal correspondences based on our experiments. Additionally, in the table, we presented the correspondence to the meters/pixel and degrees. We need it because, when we cluster data, we specify the radius in some unified units, which is the same in all zoom levels, to save the proportion.

[1] https://www.openstreetmap.org
[2] https://leafletjs.com

Table 3.1: The correspondence between the zoom level, the geohash precision, meters to pixel and degrees.

Zoom level	Geohash precision	Meters/Pixel	Degree
0	3	156,412	360
1	3	78,206	180
2	3	39,103	90
3	3	19,551	45
4	3	9,776	22.5
5	3	4,888	11.25
6	4	2,444	5.625
7	4	1,222	2.813
8	5	610.984	1.406
9	6	305.492	0.703
10	6	152.746	0.352
11	6	76.373	0.176
12	7	38.187	0.088
13	7	19.093	0.044
14	8	9.547	0.022
15	8	4.773	0.011
16	9	2.387	0.005
17	10	1.193	0.003
18	11	0.596	0.001
19	12	0.298	0.0005

For online maps, these units are pixels. But when we calculate the distance between the points, we need the radius in meters. Thus, we have the correspondence between pixels and meters for each zoom level.

Figures 3.15 and 3.16 shows the result of geohash preprocessing. In the input, we have about 1 million geo (geographic) markers. We run preprocessing with different levels of the geohash precision to find out how preprocessing time is changing depending on the precision of the geohash function. In Figure 3.15, you can see that when we increase the precision then the time increases as well. By increasing the precision, we increase, also, the number of markers (Figure 3.16). We already mentioned that we use geohash preprocessing to collapse very close points. So, we can notice that starting with the precision equaled about 9, the number of markers increases slightly. And at the end of the graph (the precision equaled 12), we can see that from 1 million markers we have only about 230

thousand (234517) unique markers.

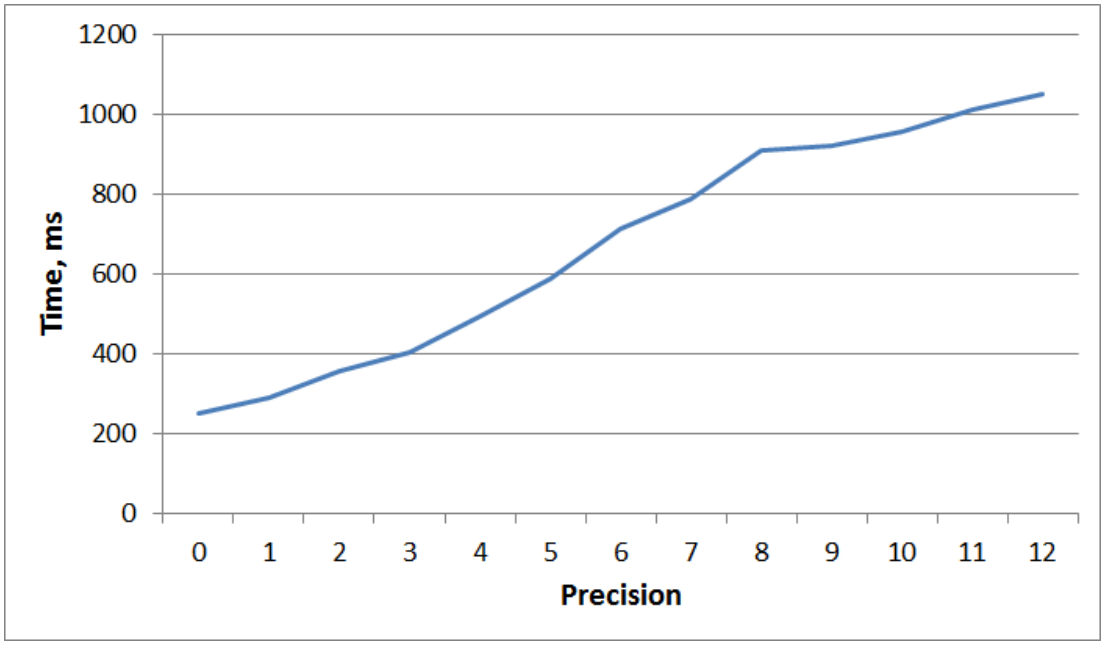


Figure 3.15: Time computation of the geohash preprocessing depending on the precision of the geohash function.

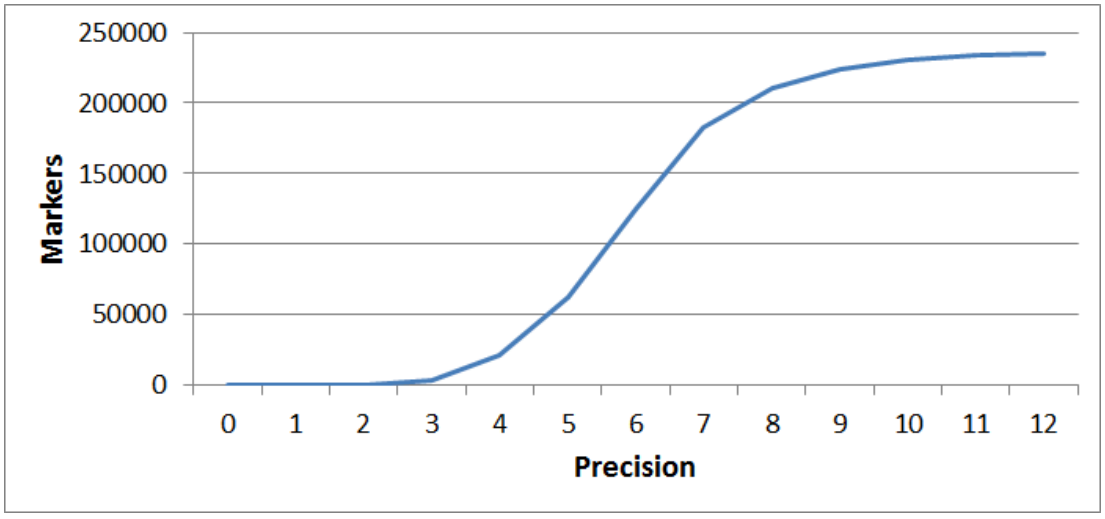


Figure 3.16: The number of markers produced by the preprocessing algorithm depending on the precision of the geohash function.

The next figures (3.17, 3.18, 3.19 and 3.20) show the result of testing the density-based geospatial clustering algorithm for two zooms: 1 and 6. Time computation on precision graphs shows that the computation time increases when the precision increases. And it is due to the increase in the number of the markers. We can notice that the behavior of the graphs is affected by the result of geohash preprocessing (Figure 3.16). So, it means that the computation time depends on the number of markers after geohash preprocessing.

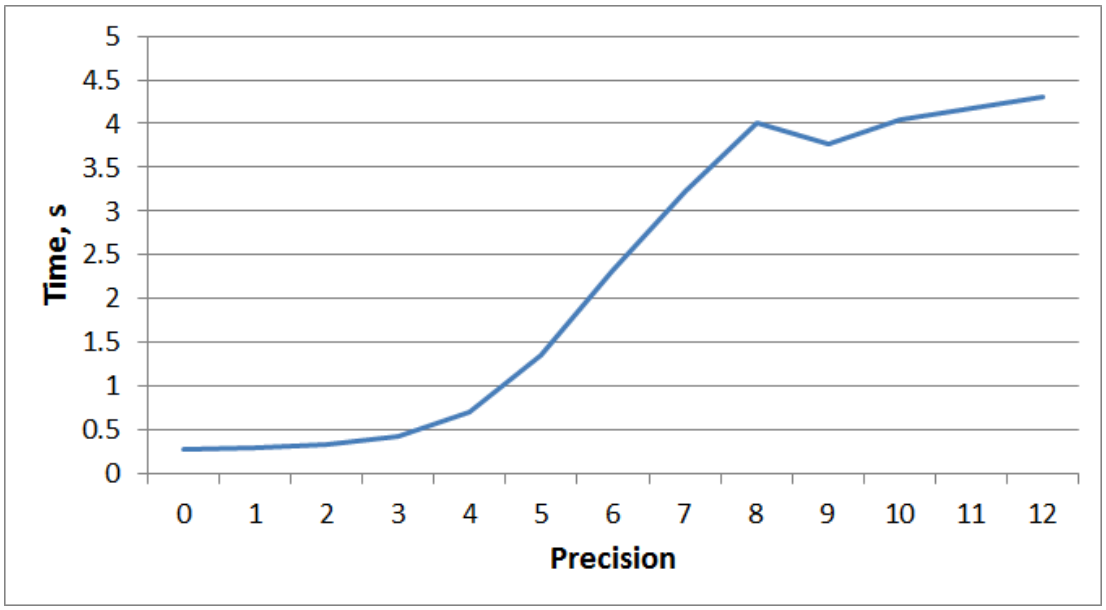


Figure 3.17: Computation time of geospatial clustering depending on the precision of the geohash function in case of the zoom level = 1.

We would like to notice that graphs of computation time for different zooms have different scales of time. It can be easily explained since it happens because not only the number of markers affects on the computation time, but also the number of clusters. In details, it means that the high geohash precision produces more markers in the geohash preprocessing stage (Figure 3.16). But on another hand, the high zoom level produces more clusters in the density-based geospatial clustering stage, since we have the fixed radius in pixels but changeable in meters and the entire amount of data spreads over the whole map. When the zoom is high, then the radius of the cluster is small and more number of clusters we can

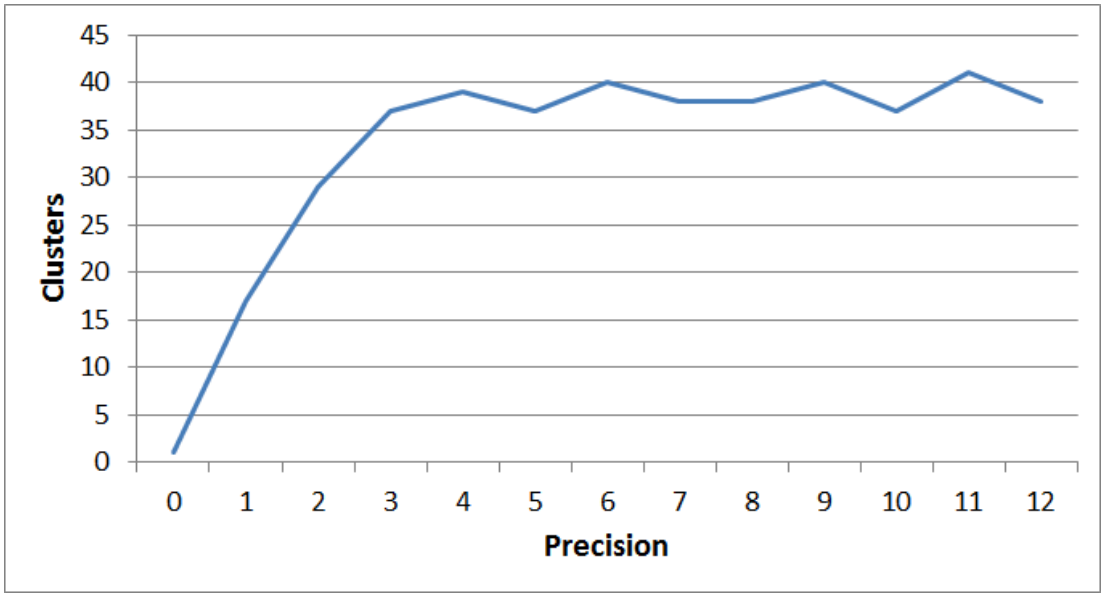


Figure 3.18: The number of clusters produced by the algorithm depending on the precision of the geohash function in case of the zoom level = 1.

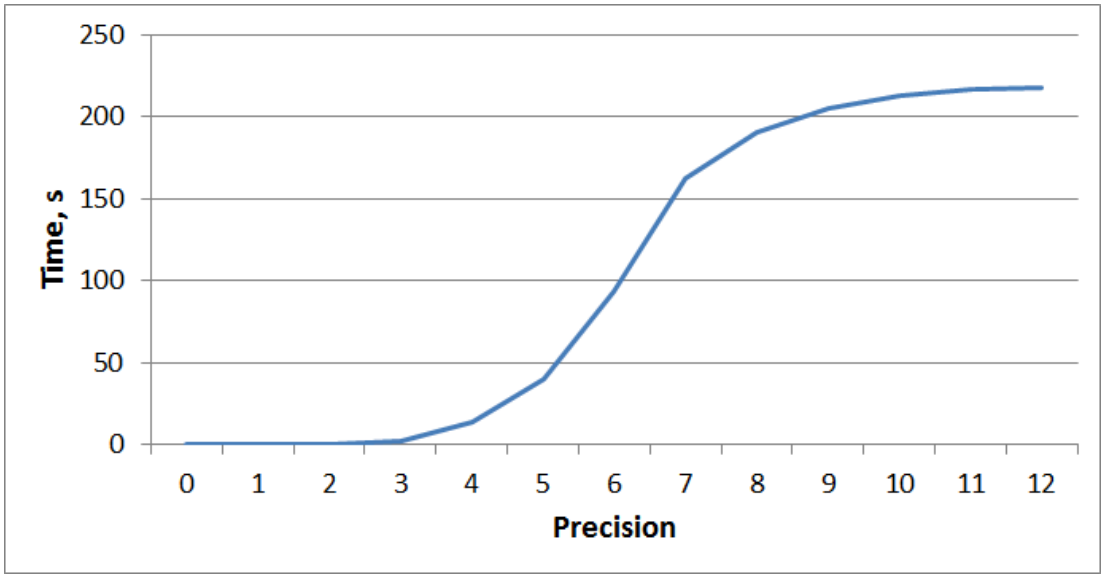


Figure 3.19: Computation time of geospatial clustering depending on the precision of the geohash function in case of the zoom level = 6.

map to the online map. But the number of clusters is limited by the map area, and if we look at the graphs in Figure 3.18 and in Figure 3.20, we can see that

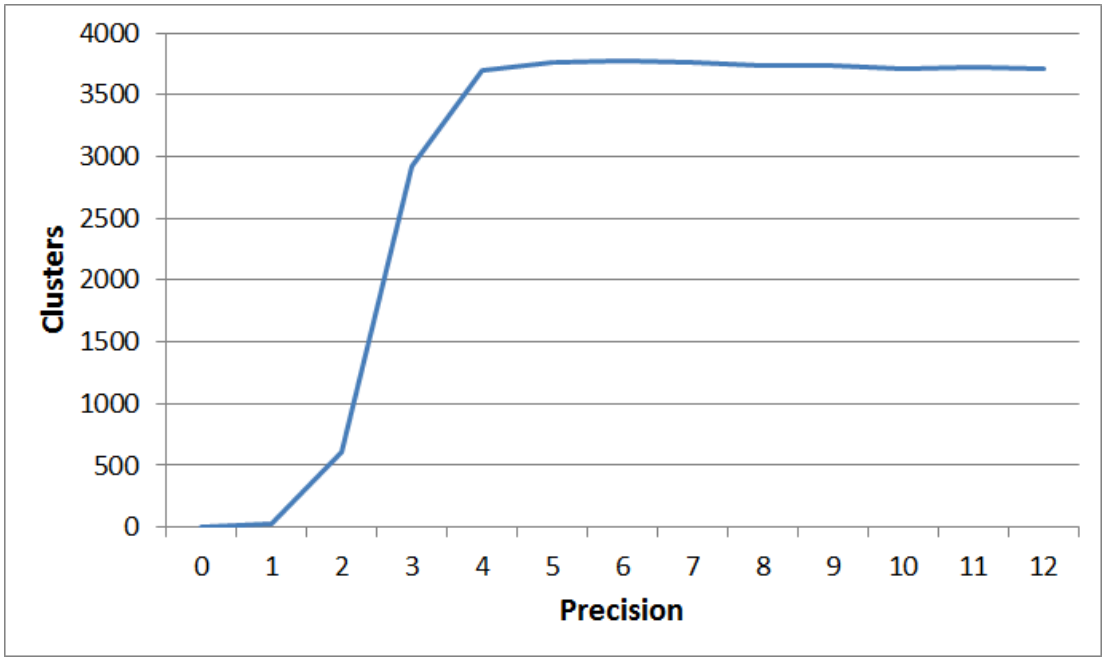


Figure 3.20: The number of clusters produced by the algorithm depending on the precision of the geohash function in case of the zoom level = 6.

the appropriate precision for the specified zoom is when the number of clusters starts to change very slightly. For example, for the zoom 6 this precision could be 4 and for the zoom 1 is 3 (Table 3.1).

We want to draw your attention one more time that we use the entire amount of data mapped to the whole online map for testing all zooms. But in a real case, when we zoom in the map, we are interested only on the visible part of the map. So, in real service, we would have fewer markers and clusters. Every zoom means dividing the map area approximately by two. So, the zoom in 10 means dividing by 2 in the power of 10. Also, we want to mention that the number of clusters in real use-cases should be something about 100. Because mapping more than 100 clusters, for example, 1000 clusters, to the map is almost the same as mapping original markers without clustering. So, if we limit the number of clusters then we exclude the influence of the number of clusters on the computation time. Actually, we do this limitation in a more natural way. When we zoom in the map, we reduce the visible part of the map, and meanwhile, we

save the radius of clusters in pixels, which will be calculated into meters according to the current zoom (Table 3.1). It means that we are saving the proportion, and by this way, we are keeping the number of clusters in some specified range for the visible part of the online map. And with the appropriately chosen geohash precision, we achieve the reasonable time of clustering. A good example to prove the last statements is the graphs in Figures 3.17 and 3.18s. Since in real use-case for the zoom level equaled 1, we can consider the entire data and the whole online map (at least the half of it). And in the figure, we can see that with the optimal precision 3, the number of clusters is about 37 and the computation time is about 400 milliseconds, which is a reasonable time for online geo clustering.

3.3 Summary

We presented a new approach to cluster georeferenced data for online maps that can improve the analysis of geospatial data by improving visual analysis. In this chapter, we described in detail the concept of the approach, provided the pseudo-code of the algorithms and a full explanation of them. To test and evaluate the approach, we implemented the tools and services to collect test data - georeferenced data from Twitter. We applied our geospatial clustering approach on these data and displayed the results on the online map. Also, we provided the results of experiments and evaluation that give information about how to configure the algorithm and how the dependency of input parameters influence on performance and accuracy of results. Based on that, we gave our recommendations about the usage of the proposed clustering algorithm.

The approach still can be improved. And thus, as future work, we are interested in testing the approach with more data and with more sophisticated data distribution on the map. Also, it would be very interesting to adjust the approach for the map-reduce concept to work on distributed systems. Additionally, we are motivated to adjust our approach and apply it in clustering any spatial data.

One of the challenges for the improvement of the algorithm, we address in the next chapter, where we propose the solution for extension of the algorithm for both geospatial and temporal aspects.

Chapter 4

GeoSpatioTemporal Clustering

The motivation of this section comes from the previous chapter, where we developed the initial approach of clustering geospatial data for online maps. In the previous chapter, we discussed the possible extension of the algorithm for one more aspect, in order to make it more applicable for current geospatiotemporal data analysis needs. Therefore, we start this chapter with motivation and related work and then we dive into the proposing approach with all needed aspects of explanation.

4.1 Related Work and Motivation

Nowadays, social media services produce a huge amount of data. A big part of them is georeferenced (geospatial) data, such as geotagged images, tweets, and so on. Georeferenced data give more possibilities for analysis of public social data, therefore we can be interested not only in the content of data but also in the location, where these data are produced. Therefore, many researchers started to use georeferenced data in their researches. One of the most popular cases is to use georeferenced data for natural disasters prediction. For example, Sakaki et al. [42] use Twitter users as social sensors to detect earthquake shakes. Another use case of using public social data to analyze natural disasters was presented by De Longueville et al. [43]. In their paper, they showed how location-based social networks (LBSN) can be used as a reliable source of geospatiotemporal

information, by analyzing the temporal, spatial and social dynamics of Twitter activity during a major forest fire event in the South of France in July 2009.

The amount of georeferenced data increases, therefore, it gives the possibility to detect not only global events, such as natural disasters but also local geospatial events. In 2013, Walther et al. [44] presented the paper, where they have shown how to detect local geospatial events in the twitter stream. According to the authors, they are interested in detecting local events, such as house fires, ongoing baseball games, bomb threats, parties, traffic jams, Broadway premiers, conferences, gatherings and demonstrations in the area they monitor.

With increasing the amount of social georeferenced data, detection of local geospatial events is becoming more promising. For that, there are different techniques and some of them imply the initial step - geospatiotemporal clustering. This step is based on the natural assumption that if something happens in some place then the messages about that appear around this event and during some period of time. Therefore, in the case of the paper [44], as the first step of detection, they try to cluster georeferenced tweets based on the specified radius and predefined time period. And then, as the second step, they apply machine learning to classify possible events. In this event detection algorithm, the first step of clustering data is very important because it affects the efficiency of the second step - machine learning classifier. And as better we do clustering, better we can classifier possible local geospatial events. Therefore, the goal of this thesis's chapter is to propose a novel approach for geospatiotemporal clustering of georeferenced data that can efficiently produce the geospatiotemporal clusters for the geospatial event detection algorithms.

Most studies on discovering clusters for ordinary data are impractical for clustering spatiotemporal data because the knowledge discovery process for spatiotemporal data is more complex than for non-spatial and non-temporal [45]. Therefore, it requires novel approaches, in order to extract useful knowledge from such type of data. Kisikevich et al. [46], in 2010, presented the survey of spatiotemporal clustering. Their report includes the classification of different types of spatiotemporal data, the description of the state-of-the-art approaches and methods and the description of trajectory clustering as a type of spatiotemporal clustering. Additionally, they presented several possible application domains.

The ST-DBSCAN approach was presented in 2006 by Birant et al. [45]. They presented their adjustment of the DBSCAN algorithm for spatiotemporal data. In contrast to the existing density-based clustering algorithms, their approach has the ability to discover clusters according to non-spatial, spatial and temporal values of the objects.

We are motivated to provide the method for geospatiotemporal clustering that can produce the density-based in space and intensity-based in time clusters. Such clusters would have the centers in the densest area in space and in the intensest area in time. Meanwhile, we meet the requirement of the online algorithm. It means that the algorithm should be able to process its input piece-by-piece in a serial fashion without having the entire input available from the start. With these requirements, the algorithm can be useful for clustering georeferenced data from social networks to provide the clusters that can be candidates of local geospatial events. And in this way, we can support and improve the geospatial event detection algorithms. Therefore, in this chapter, we propose the approach for geospatiotemporal clustering that can be used as an initial step for geospatial event detection algorithms.

The remainder of the chapter is organized as follows: We start with Section 4.3, where we propose our method. This section contains the detailed explanation of methods with the illustration of figures in Section 4.3.1 and formalized pseudo code of algorithms that build the method in Section 4.3.2. After that, we summarise the section and discuss future work in Section 4.4.

4.2 GeoSpatioTemporal Data Normalization

Originally we work with data that contain location and time information jointly with textual information. Our data are georeferenced messages from Twitter. But additionally to geotags, the tweet message could contain additional information about the location in the form of location words - street names, squares, points of interests. Such words could adjust coordinates more accurate. By the same principle, tweets could contain information about the time. For example, it could be that the tweet posted today is describing the situation happened yesterday. Therefore, detection of time-related keywords could help adjust time feature as

well. All such specifics should be considered in the process of transformation of data into geospatiotemporal data.

In Figure 4.1, we present the overview of the main steps of the process of data transformation into geospatiotemporal textual data. In the original data, we are interested in location, time and text features. The original text should be cleaned before processing. Cleaning means removing irrelevant words that are not important in terms of describing the topic. Such words could be "stop words", such as "the", "an", "a" and so on. Also, the cleaning process means the extraction of twitter related entities, such as hashtags. The next step is passing the text through the Named Entity Recognizer module. Here we use of the most popular library for named entity recognition in natural language processing (NLP) proposed by Stanford - Stanford Named Entity Recognizer (NER) [16] [17]. The result of this recognizer is entities of different categories (classes). Depending on modules used in the recognizer, it can recognize 3,4 or 7 classes:

- **3 classes**: Location, Person, Organization
- **4 classes**: Location, Person, Organization, Misc
- **7 classes**: *Location*, Person, Organization, Money, Percent, *Date*, *Time*

We are interested in location words and date/time words. Therefore, we use the module with 7 classes. Location words can be passed through the process of the geolocation that we have described in Section 2.2 and then merged with the coordinates taken from the original data. The location merging could be valuable in the case if coordinates are attached to the entire country/city and in the text, there is the reference to the concrete street or point of interest. Therefore, the combination of the country/city name and street name could give more valuable location information. The same procedure could be applied for time and time/date entities.

The result of the framework is geospatiotemporal textual data. In comparison with the original data, the result data contain more valuable, accurate and normalized set of features that can be used for further clustering. We have already proposed the method for clustering geospatial data in the previous chapter. In this chapter, we propose the method for geospatiotemporal clustering.

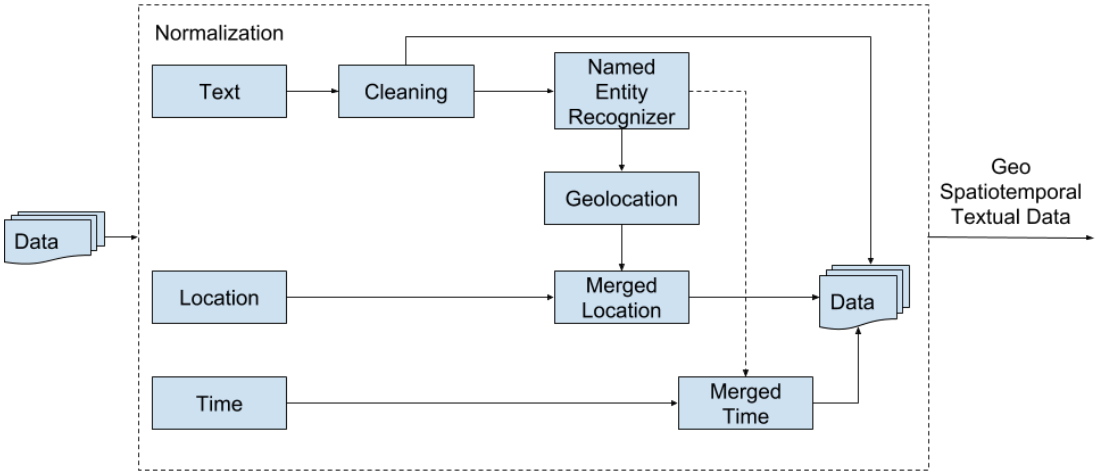


Figure 4.1: The process of data normalization and transformation into geospatiotemporal textual data.

4.3 Density and Intensity-based GeoSpatioTemporal Clustering with Fixed Distance and Time Radius

4.3.1 Method

In this section, we describe the main concept of the proposed approach. We start with an overview of entities, which we use in our approach. In Figure 4.2, we present the UML (Unified Modeling Language) diagram of the entity classes. Understanding of them is important for the further explanation of the algorithm in the next section. The main class has the name *Cluster*. It has the following fields: *center* - the center of the cluster, *points* - the list of points that constitute the cluster, *radius* - the geographic radius of the cluster, *date* - the center date of the cluster in time space, *timeRadius* - the radius of the cluster in time space. Additionally, we have *minDistance* and *maxDistance* that show the distance to the closest and the farthest points from the center. Also, we have *minDate* and *maxDate* that show the time borders of the cluster. Another supporting entities are *Point* and *LatLng*. *LatLng* consists of latitude and longitude coordinates. *Point* consists of the *position* (*LatLng* type) and the *date*.

Once we introduced entities, we can start to describe the methods for clustering. The clustering algorithm is the geospatiotemporal algorithm that has 2 aspects of clustering: spatial (geographic space) and temporal (time space). The

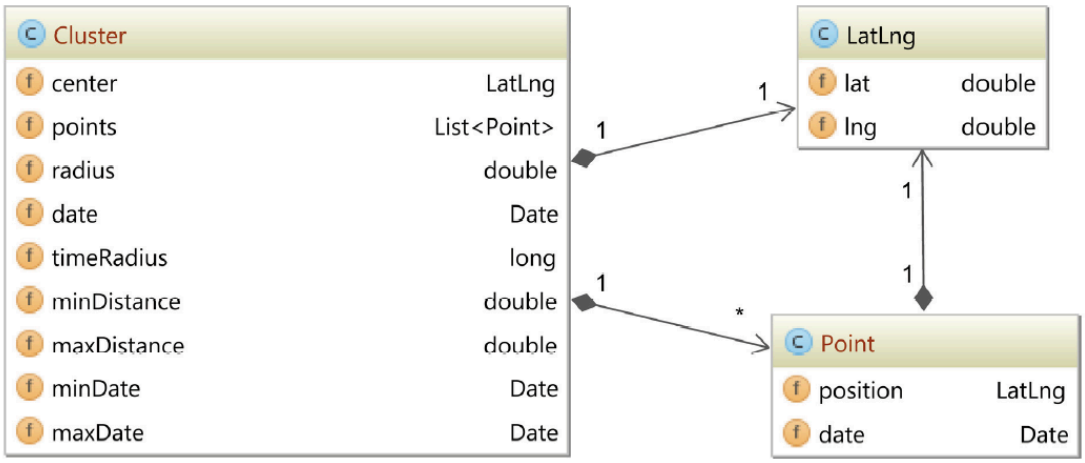


Figure 4.2: Diagram of entities.

spatial aspect of clustering is supported by the algorithm proposed by Amirkhanyan et al. [1] in 2015 and presented in Chapter 3. We use this approach because it works in online mode and it produces the density-based clusters based on the specified radius. We use this algorithm with additional adjustments that are explained in the next section. The detailed explanation of the geo clustering algorithm, you can find in the paper [1] and in Chapter 3. But here, we provide only a brief overview of the approach jointly with the figure taken from Chapter 3 Section 3.2.2 that it is needed for the complete description and understanding of this chapter independently and consistently.

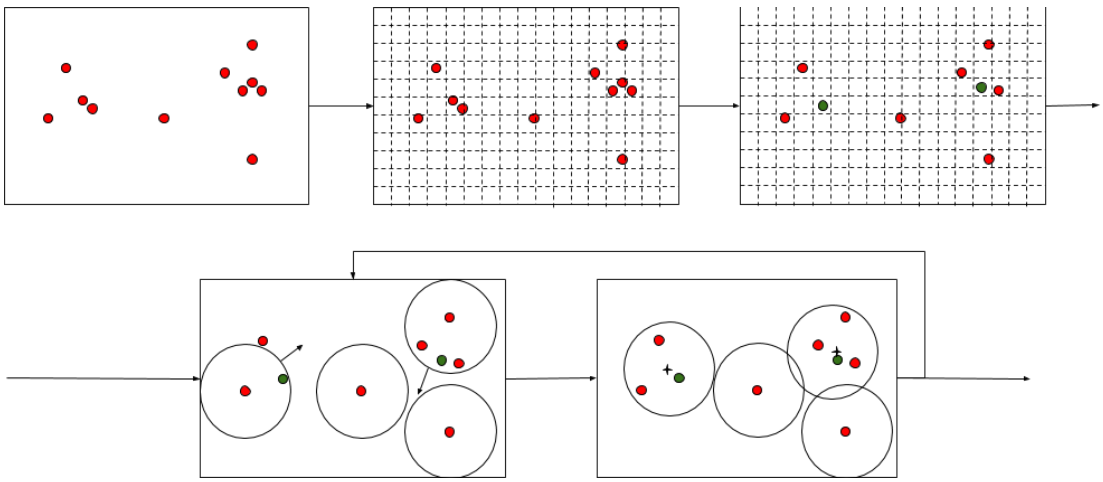


Figure 4.3: Illustration of the geospatial density-based clustering.

In Figure 4.3, you can see the detailed illustration of the geospatial clustering. The task of the clustering algorithm is to cluster georeferenced data. Georeferenced data are presented in the figure as red and green points. The first step of the clustering is the grid-based clustering, which we described in Section 3.2.2. Grid-based clustering is based on the geohash function that divides a map into cells of the predefined size depending on accuracy. After division, we group red points that are close enough to each other, by other words, points that are in the same cell. We show the grouped points as green points in the figure. After the grid-based clustering, we do the density-based clustering as the second step of the algorithm. We cluster points that close to each other based on distance measurement, define the center of the cluster (centroid) in the densest area and do it repeatedly until we handle all georeferenced points. As a result, we have geospatial clusters that have the center in the densest area.

Next, we consider the second aspect of clustering - temporal aspect. The straightforward (naive) method for that is to divide the timeline into fragments with the predefined size. The disadvantage of this solution is that the center of the time frame is not in the intensest area. Meanwhile, theoretically, if we suppose that some event happened then it should produce the burst time peaks that reflect the time of the main activities. In time series analysis, such time peaks provide the main interest. And usually, researchers, which work in time-serial analysis, are interested in the detection of such peaks and even in the prediction of them. From our side, we are interested in the combination of the naive fragmentation of the timeline and peak detection, in order to produce the time fragments with the high intensity of points.

For the temporal aspect of clustering, we propose to use the intensity-based fragmentation of the timeline. In this approach, the center of the fragment is calculated with the formula $\frac{\sum_{i=1}^{n} t_i * p_i}{\sum_{i=1}^{n} p_i}$, where t - time, p - the number of points. With that, we have the time center in the intensest area. Also, we are interested that time fragments would have the local time peaks. Therefore, we make the approach iterative that changes the center and borders of the time fragment depending on the change of the intensity distribution

In Figure 4.4, you can see the illustration of the approach. We imply that the state of the fragment changes under influence of incoming points. After

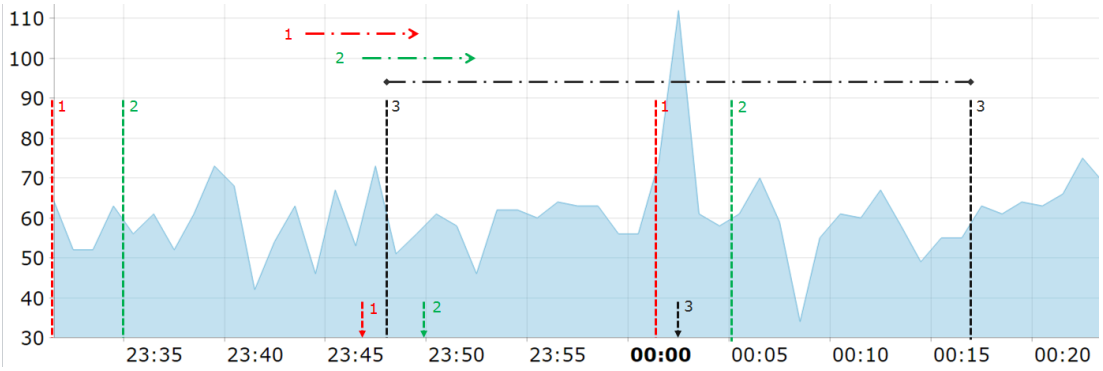


Figure 4.4: The intensity-based fragmentation of the timeline (temporal clustering).

each incoming point, inside of the time frame, we need to recalculate the center and move the frame to the more intense area. For explanation, we consider 2 iterations. At one point in time, we could have the time fragment with the borders and center presented by the red dash lines (1). Then we assume that new points came and they are time close to the right border. After that, we need to recalculate the center, in order to keep the center in the more intensive area. Therefore, after that recalculation, we have to move the time fragment to the new center (2 - green dash lines in the figure). In this way, new incoming points force the time frame to continuously move, in order to keep the state with the intensity-based center. Naturally, points are sorted by time and the movement goes iteratively to the right. The time frame moves until the time frame gets the stable position. In our example, it happens when the time frame reaches a position 3 (black lines), which is characterized by the time peak (00:02). This time peak forces the time frame to slow down movement and finally stop. After that, we can start to build the next intensity-based time frame.

4.3.2 Algorithm

This section is dedicated to the description of the algorithm methods that are used in our approach of geospatiotemporal clustering. To simplify the explanation, we present our approach by means of describing several algorithm methods. All together, they make up the idea of the approach.

We start the description with presenting the method for calculation of the time center of the temporal part of the cluster. It is calculated by the formula

```
1: procedure centerDate(clusterDate, pointDate, weight)
2:     clusterTime ← clusterDate.getTime()
3:     pointTime ← pointDate.getTime()
4:     newClusterTime ← (clusterTime * weight + pointTime)/(weight + 1)
5:     return newDate(newClusterTime)
6: end procedure
```

Figure 4.5: The calculation of the time center of the cluster (iterative version).

$\frac{\sum_{i=1}^{n} t_i * p_i}{\sum_{i=1}^{n} p_i}$, where t - time, p - the number of points. In Figure 4.5, we present the pseudo code of the implementation of this formula. You can see that the method accepts the cluster date, point date and the total weight, which is the number of the points in the cluster. Firstly, we retrieve the time in milliseconds from the dates (lines 2-3) and then we calculate the average value (line 4). New average value becomes the new cluster date (line 5). You can notice that, in statement 4, we do not apply the full formula, but we use the simplified iterative version, in which we consider the current cluster date (*clusterTime*) as an intermediate result. Such iterative simplification helps us to save the computation time and not calculate the average time for all points again.

```
1: procedure addPointToCluster(cluster, point)
2:     center ← cluster.center
3:     date ← cluster.date
4:     points ← cluster.points
5:     size ← cluster.points.size
6:     if center == null then
7:         center ← point.position
8:     else
9:         center ← center(center, point.position, size)
10:    end if
11:    if date == null then
12:        date ← point.date
13:    else
14:        date ← centerDate(date, point.date, size)
15:    end if
16:    points.add(point)
17:    updateMinMaxPoint(cluster, point)
18: end procedure
```

Figure 4.6: Assigning the point to the cluster.

Later, in this section, we present the algorithm for geospatiotemporal clustering. But before that, we need to draw your attention that in the pseudo-code we use references to the supporting methods, which we introduced in the previous chapter (Section 3.2.3). These supporting algorithms are well-known: (1) the algorithm for calculating the distance between two geographic points [41] and (2) the algorithm for calculating the geographic center [23]. We do not provide their implementations in this chapter, but we just use the name of the methods to refer to them: *distance()* for the first algorithm and *center()* for the second one. Another supporting method is *isInsideTime()*. This method is used to find out whether the date of the point is in the scope of the time frame of the cluster. By other words, it checks whether the time distance between 2 temporal points is less or equal to the time distance.

In Figure 4.6, we present the algorithm for adding the point to the cluster Firstly, we retrieve some variables from the cluster that we use in the algorithm (lines 2-5). Then we check, whether the center of the cluster is *null*. If the center is *null* and the passed point is the first point in the cluster, then we set the center of the cluster as a position of the point. If the passed point is not the first, then we need to calculate a new center based on the new point and existing points. For that, we invoke the method *center()* and pass the cluster center, the point's position and the size of the points list. The same procedure we do for the date. If the cluster does not have the date then the point date becomes the cluster date. Otherwise, we calculate the center date for the cluster with the method *centerDate()*, which we introduced above (Figure 4.5). In the end, we need to add the point to the list of the points. Also, we calculate some supporting parameters, such as min and max distances between the cluster center and points, and the time borders of the clusters. We need these parameters for detecting possible overlaps among the clusters, which we show later in this section.

In Figure 4.7, we present the algorithm for adding the point to the closest cluster. We go through the clusters and check the conditions. Firstly, we check whether the point is inside the cluster's time radius. For that, we use the method *isInsideTime()*. The next step is the calculation of the distance between the point and the cluster center. By this way, we try to find the closest in space cluster that is inside of the accepted time radius. If we did not find the appropriate

```
1: procedure addToClosestCluster(point, clusters, radius, timeRadius)
2:     cluster ← null
3:     distance ← radius
4:     overlapping ← false
5:     for each c in clusters do
6:         if !isInsideTime(point.date, c.date, timeRadius) then
7:             continue;
8:         end if
9:         d ← distance(c.center, point.position)
10:        if d <= distance) then
11:            distance ← d
12:            cluster ← c
13:        end if
14:        if cluster = null then
15:            if !clusterOverlapping then
16:                clusterOverlapping = d < (cluster.maxDistance + radius)
17:            end if
18:        end if
19:    end for
20:    if cluster = null then
21:        if !overlapping then
22:            cluster ← newCluster(radius, timeRadius)
23:            clusters.add(cluster)
24:        else
25:            return point
26:        end if
27:    else
28:        return point
29:    end if
30:    addPointToCluster(cluster, marker)
31: end procedure
```

Figure 4.7: Finding the closest geospatiotemporal cluster and assigning the point to it.

cluster then we can create a new one. But we can do it only if this new cluster will not have the overlapping with existing clusters (lines 14-18). For detection of overlapping, we use the *maxDistance* parameter that we calculate in the algorithm from Figure 4.6. In the end, if we found the closest cluster or we are allowed to create the new cluster, which does not overlap the existing clusters (lines 20-23), we add the point to the cluster (line 30). Otherwise, we return the point to the pool and we will try to assign this point later in the next iteration (lines 25, 28).

```
1: procedure CLUSTER(points, clusters, radius, timeRadius)
2:     overlapPoints ← null
3:     while points.size! = 0 do
4:         point ← points.nextPoint
5:         tClusters ← retriveTimeCloseClusters(clusters, point)
6:         overlapPoints ← addToClosestCluster(point, tClusters, radius, timeRadius)
7:         if points.size == 0 then
8:             points ← pullPoints(clusters)
9:             points ← overlapPoints
10:        end if
11:    end while
12: end procedure
```

Figure 4.8: The density and intensity-based geospatiotemporal clustering.

Figure 4.8 presents the simplified top view of the clustering algorithm. The method takes the *points*, *clusters*, *radius* and *timeRadius*. We iterate through the points in the loop until we empty the queue. For each point, we retrieve the time close clusters (clusters that have the time frame appropriate for the considering point). Then we invoke the method *addToClosestCluster()*. This method returns some possible non-clustered points that we save into the *overlapPoints* variable for the next iteration. Also, it could be that some points become out of the spatial and temporal radius of the cluster after adjusting the spatial and temporal centers (recalculation of the cluster center and date). Therefore, we need to pull such points and pass them again into the queue. For that we use the method *pullPoints()*, which we describe below.

In Figure 4.9, we present the final important component of our density and intensity-based geospatiotemporal clustering algorithm. In this method, we go through the clusters and check whether the points of the clusters are still inside the temporal and spatial spaces. For that, we use methods *isInsideTime()* and *isInRadius()*. If some points do not fulfill the conditions, we pull them from the points list. After that, we need to recalculate the cluster center (geospatial and temporal). We return the pulled points back to the queue for further handling.

```
1: procedure pullPoints(cluster)
2:     points ← cluster.points
3:     for each point in points do
4:         if !isInsideTime(cluster.date, point.date, cluster.timeRadius) then
5:             pulledPoints ← cluster.points.pull(point)
6:         end if
7:         if !isInRadius(cluster.center, point.position, cluster.radius) then
8:             pulledPoints ← cluster.points.pull(point)
9:         end if
10:    end for
11:    recalculateClusterCenterAndDate(cluster, pulledPoints)
12:    return pulledPoints
13: end procedure
```

Figure 4.9: Pulling the points from the clusters that are out of the cluster's range.

4.3.3 Development and Testing

For testing and developing, we used different datasets collected from Twitter from 2 cities: New York and London in different date ranges. One of the datasets that we used is data from Chapter 3. It allowed testing our algorithm with keeping the same potential clusters in spatial space. We integrated the clustering algorithm to the developed application for analysis geospatiotemporal data. We will describe this application jointly with use cases in details in Chapter 6. Meanwhile, we have used screenshots of the application to explain proposed approaches in this and previous chapters (Figure 3.14 and Figure 4.4). Integration of the algorithm to the application helped us to visualize the results of clustering on online maps during development and testing.

4.4 Summary

We developed and presented the approach for geospatiotemporal clustering. The approach meets requirements of the online algorithm and provides clusters with the center in the densest area in space and in the intensest area in time. Therefore, we call our approach - the density and intensity-based clustering algorithm. The proposed approach can be used for analysis of georeferenced data from social networks. Particularly, it can be used for initial clustering georeferenced data

that can be used in the geospatial event detection algorithms. In this chapter, we described in detail the concept of the approach, provided the pseudo code of the algorithm methods and the full explanation of them. The presented algorithm was implemented and tested with collected from Twitter georeferenced data.

We are interested in the future development of geospatiotemporal clustering. In the current implementation, we have to specify the distance and time radius. It brings some limitations, in the case if the geospatial event covers a bigger area and happens a longer time than we specified. Therefore, one of possible future work is to make the algorithm with the flexible time and distance radius that can be changed depending on the changeable distribution of the time intensity and spatial density.

For the case of geospatiotemporal clustering data from social networks, it is important to consider also the content feature. Therefore, we would like to add a new dimensional into the clustering algorithm - textual aspect. With that, we want to provide an online algorithm that can produce geospatiotemporal clusters among similar topics (words). Such an algorithm could provide the initial detection of local geospatial events from social georeferenced data.

The result of the chapter and future possible directions of research are aimed to provide the methods for discovering knowledge in geospatiotemporal data. Also it can support algorithms for local geospatial events detection.

Chapter 5

The Framework for Spatiotemporal Sequential Rule Mining

In this chapter, we consider geospatiotemporal data from the perspective of pattern and rule mining jointly with the use case around public safety awareness.

5.1 Motivation and Related Work

Data mining techniques are used efficiently in different application areas. One of such areas is crime data analytics, which is the subject of our interest in this chapter. There are many success cases of applying data mining approaches to crime data, in order to discover crime patterns. Every case is specific and depends on data (information) and the addressing goal. One of the common approaches for crime data analytics is the hotspot map - mapping the distribution of crimes on the map to find the most dangerous places and changes over them. Such visualization with statistical analysis helps police wisely control police patrols in the city.

In 2004, H. Chen et al. [47] made an overview of crime data mining techniques that can be used for crime data analytics. They proposed the general framework and some case studies (*Coplink*) that are aimed to unify the application of data mining techniques depending on data. In 2006, Shyam Varan Nath et al. [48]

presented the paper - crime pattern detecting using data mining. In that paper, they investigated how to convert crime information into a data mining problem. They showed the benefits of using data mining approaches for criminal data analytics. More specifically, they used clustering based models (k-means) to help in the identification of crime patterns. Later, in 2011, Mohler et al. [49] presented the approach - self-exciting point process modeling of crime. Their approach is based on space-time clustering, which is modeled in seismology by self-existing point process. They showed that these methods are well suited for criminological application and they illustrated how hotspot maps can be improved using the self-exciting point process framework. In 2013, Tong Wang et al. [50] [51] presented the approach - *Series Finder*. This approach detects the pattern of crimes based on the similarity: (1) the crime-crime similarity - how similar crimes in a pattern set; (2) pattern-crime similarity - whether a crime is similar enough to set that it should be included in. Based on these similarities, they try to build a sequential pattern. The similarity calculation is based on the similarity of attributes. Therefore, for that approach, it is important to have detailed data with as much as possible significant attributes.

We can see that, depending on data, their attributes and the goal, different data mining techniques can be applied. In this chapter of the thesis, we want to address the challenge of sequence pattern and sequential rule mining in spatiotemporal data. For this challenge, we are aimed to develop a framework that can unitize the sequential pattern/rule mining techniques, in order to find patterns and rules among spatiotemporal items. The use case for this framework can be crime data analytics and discovery of crime patterns.

The remainder of the chapter is organized as follows: Further, in this section, we introduce the general idea of the sequence pattern mining. Then in Section 5.2, we make an overview of the proposed framework with a detailed description of the workflow and main components, such as geocoding and pattern discovery modules. Section 5.3 reflects the explanation of some algorithms that are part of the framework workflow. To explain the algorithms, we provide the simplified pseudo-code of methods. Also, we provide the use case (crime data mining) in Section 5.4. After that, in Section 5.5, we discuss the proposed framework, conclude the chapter and provide future directions for research.

5.1.1 Sequence Pattern Mining: Introduction

Sequence pattern and sequential rule mining are techniques that can be used in different application areas including crime data analytics. Sequences are a very common type of data structures that can be found in many domains such as bioinformatics (DNA sequence), sequences of clicks on websites, the behavior of learners in e-learning, sequences of what customers buy in retail stores, sentences in a text, etc. An important data mining problem is to design an algorithm for discovering hidden patterns in sequences [52].

A lot of researches on this topic in the field of data mining and various algorithms have been proposed [52] [53]. A sequential pattern is a subsequence that appears in several sequences of a database. A sequential rule is a rule of the form $\mathbf{X} \rightarrow \mathbf{Y}$, where X and Y are sets of items (itemsets). A rule $\mathbf{X} \rightarrow \mathbf{Y}$ is interpreted as if items in X occurs (in any order), then it will be followed by the items in Y (in any order). The very detailed survey of sequential pattern mining is presented in the paper [53].

5.2 The Framework Design

In this section, we provide an overview of our framework. The overview includes the description of the data entities, framework workflow, geocoding module and pattern discovery module. And later in Section 5.4, we consider the use case.

5.2.1 Entities

The framework works with spatiotemporal data that should have three properties (features): (1) *type* - the string representation of the item, (2) *time* - the temporal aspect of the item, and (3) *geographic coordinates* - spatial aspect of the item. The item with these three parameters, we call the spatiotemporal item (ST-item). In Figure 5.1, we present the diagram of the entity classes. Understanding of them is important for further explanation. The main class has the name *ST_Item*. It has the following fields: *type*, *time* and *location*. The location has fields: *geohash*[1] - the hash string representation of the coordinates, *latitude* and *longitude* - the

[1] https://en.wikipedia.org/wiki/Geohash

geographic coordinates of the item. The *type* field should have the limited number of the predefined string values that can be used to categorize data.

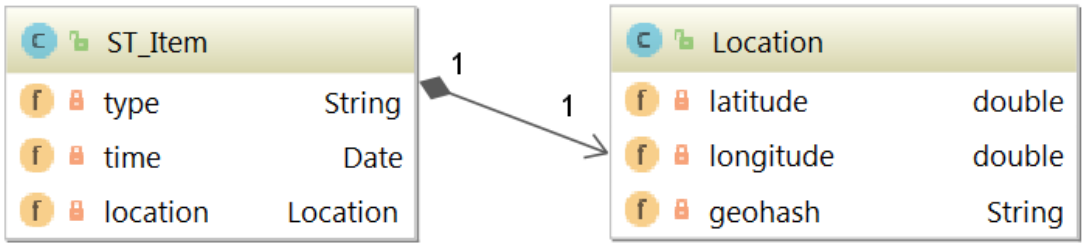


Figure 5.1: Diagram of entities.

For further explanation, we need to define the term - the sequence of ST-items. We have already mentioned that data, with which our framework should work, should have three fields: *type*, *time* and *location*. Based on that, we build sequences - the ordered list of items. The method for building such sequences will be discussed in the next section. Here, we need to point out that the sequence for our type of data is the list of sorted in time items that are located close to each other within the specified location area. The sequences of ST-items, we call spatiotemporal sequences - ST-sequences.

5.2.2 Workflow

Since we introduced the entities and required definitions, we can explain the framework in detail. In Figure 5.2, we present the diagram of the framework that shows the main workflow. The initial step is the normalization. Within this step, we extract required fields (*type*, *time*, *location*). If the location is represented as a string address, we apply the geocoding procedure, which we will describe later in this section. Within the geocoding step, we convert the address information into geographic coordinates by using Google Geocoding API [14]. After the normalization and geocoding steps, we have items that fit the entities diagram from Figure 5.1. The next step is the STS-Builder (spatiotemporal sequence builder). The purpose of this step is to build the sorted in time sequences of items that are close to each other in spatial and temporal aspects. Since after the STS-Builder we have ST-sequences, we can apply sequence pattern and sequential rule mining algorithms, in order to find any patterns and/or association

rules. The result of the framework is patterns and sequential rules found in spatiotemporal data. The final results depend on the configuration that was used in the procedure of sequences creation. We will discuss this procedure later in Section 5.3.

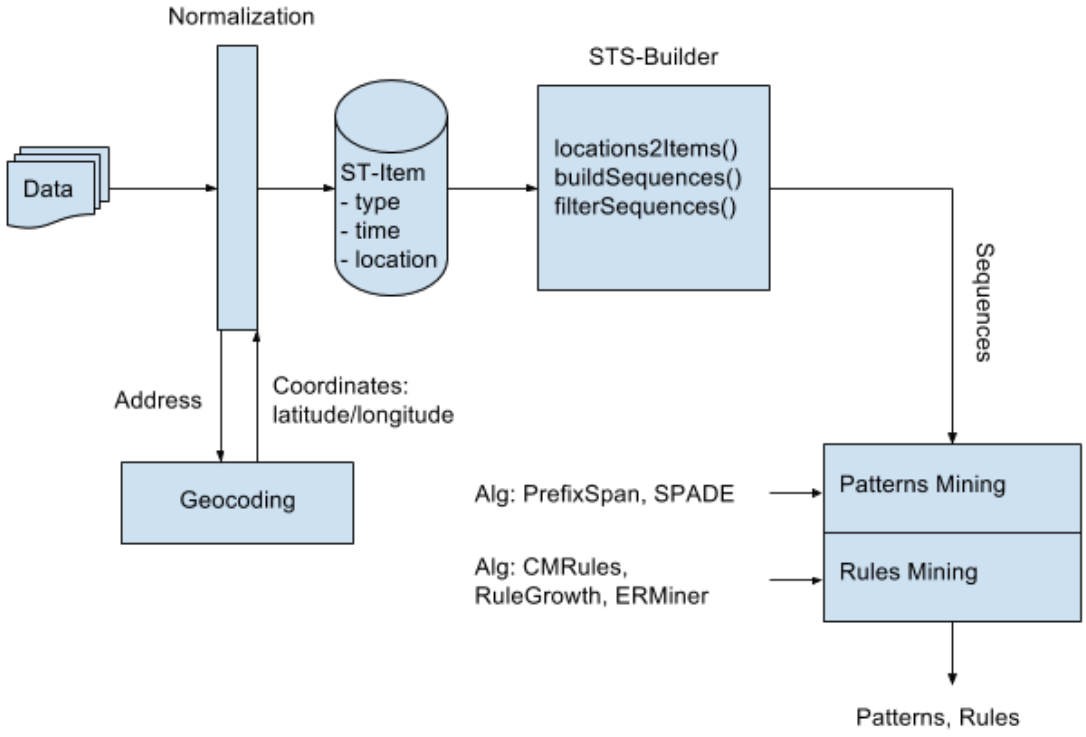


Figure 5.2: Framework for spatiotemporal pattern/rule mining.

5.2.3 Geocoding

In Figure 5.3, we present the module that performs geocoding of data. Geocoding is the conversion of the address into the geographic coordinates - the pair of the latitude and longitude coordinates. For this procedure, firstly, we extract location-related fields and normalize them - we build the full concatenated address. We have to do it because the information about the address can be split and stored in several fields of data. Based on the complete full address, we do requests to the Google Geocoding API [14] that returns the list of possible locations sorted by the relevance. In order to reduce the number of requests to the Google

service, we build the cache that stores requested location information. The result of the Geocoding module is the location entity. The location entity contains the pair of the latitude and longitude coordinates and geohash value (Figure 5.1). On the final stage, we assign the location to the corresponding item in the dataset.

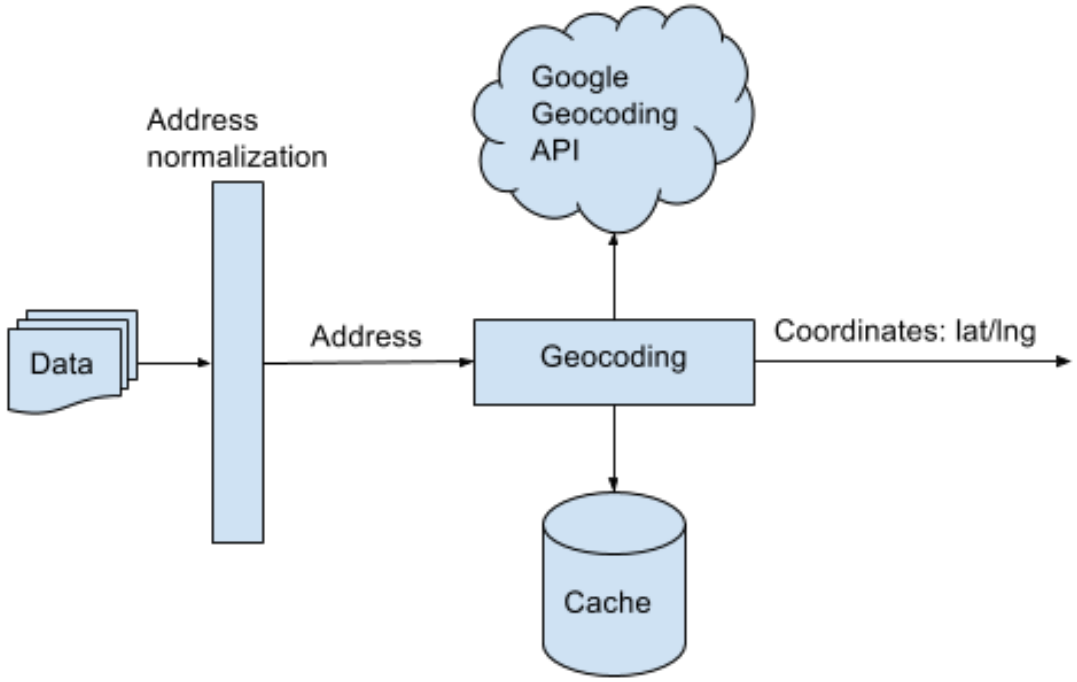


Figure 5.3: Geocoding module of the framework.

5.2.4 Pattern and Rule Discovery

In Figure 5.4, we present the pattern discovery module. This module is the abstract application level above pattern mining methods. The module accepts the string representation of ST-sequences - the sequences of the *type* values of ST-items. These string sequences are the result of the STS-Builder module (Figure 5.2). Despite that the string sequences have more a human readable format, many pattern mining algorithms work with numerical data. Therefore, we have to convert (mapping) strings to numbers. For that, we have the type mapping

step, in which we try to map the string values to integers and build the mapper that stores the correspondences. The mapper is the set of combinations between string and integer values. After the mapping step, we work with the numerical sequences. And it allows us to apply most pattern mining algorithms in the mining submodule. After the detection of patterns and rules, we use the *Mapper*, built in the initial step, to convert numerical representation into a string representation. The result of this module is the final result of the framework.

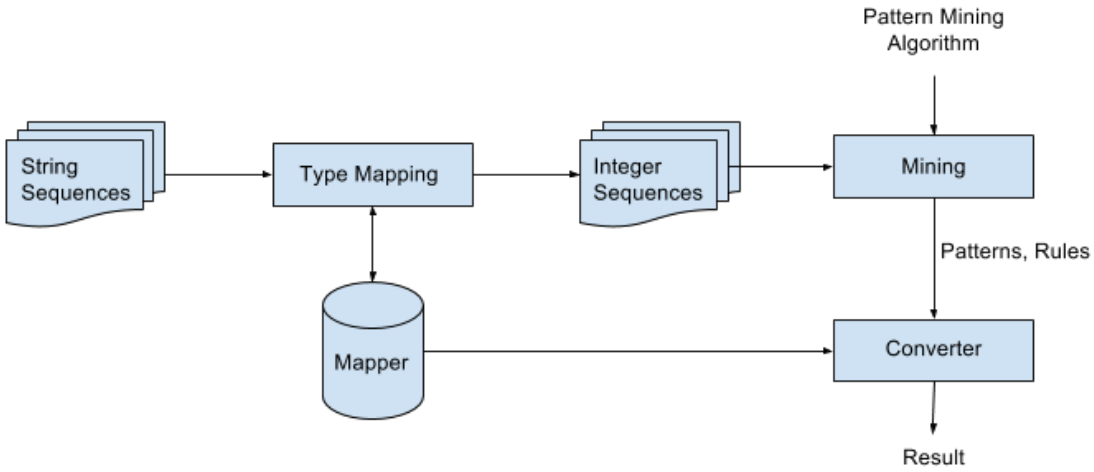


Figure 5.4: Pattern discovery module.

5.3 STS-Builder: Algorithms

This section is dedicated to the description of the algorithms encapsulated into the STS-Builder module. To simplify the explanation, we present the procedure within 3 methods with separated descriptions.

The goal of the STS-Builder module is to build spatiotemporal sequences. For that, we need to group ST-items based on the location and time. We do it by 2 steps: fetch items from the defined location area and then create time-ordered sequences. To have the possibility to fetch needed ST-items by the location parameter, we create *location2Items* storage. In Figure 5.5, you can see the algorithm for that. The method goes through items, extract their location keys (line 4) and group items based on keys (lines 5, 10). In our case, the location

```
1: procedure LOCATIONS2ITEMS(items)
2:     locations2Items ← {}
3:     for each item in items do
4:         key ← item.getLocation().getKey()
5:         locItems ← location2Items.get(key)
6:         if locItems == null then
7:             locItems ← []
8:             locations2Items.put(locItems)
9:         end if
10:        locItems.add(item)
11:    end for
12:    return locations2Items
13: end procedure
```

Figure 5.5: Build the locations to items mapping.

key is the geohash value with the predefined accuracy. If we do not have the list of items with considering location key, we create a new list and put it into the storage (lines 6-9). As a result, we have *locations2Items* storage, from which we can easily extract items based on the location key.

We have already the initial understanding of the term - spatiotemporal sequence (ST-sequence) that we introduced in the previous section. With this understanding, we look at Figure 5.6, where we present the algorithm that builds sequences from ST-items. The method takes 2 parameters: *items* - the list of ST-items and *timeDistance* - the max time distance between items in the time ordered sequence. Firstly, we sort items by time and then we iterate through them. For each item, we retrieve the time field (line 7). If it is needed, we defined the last time in the sequence (lines 8-10). Then we check the time distance between the current item and the last time in the sequence (line 11). If the time distance is less than the parameter *timeDistance*, we add the item into the current *sequence* (line 12). Otherwise, we add the current *sequence* into the *sequencesList*, create a new *sequence* and add the item into the new built sequence (lines 14-16). After that, we need to update the *last* time of the sequence (line 18). At the end of the method, we have the list of the ST-sequences - *sequencesList*.

Finally, in Figure 5.7, we present the helper method that is used to filter not valid sequences. The filter excludes the sequences, which size is less than the specified *size*. We have this method to exclude short and not valid sequences that

```
1: procedure BUILDSEQUENCES(items, timeDistance)
2:     sort(items)
3:     sequence ← []
4:     sequencesList ← []
5:     last ← 0
6:     for each item in items do
7:         time ← item.getTime()
8:         if sequences.isEmpty() then
9:             last ← time;
10:        end if
11:        if time − last < timeDistance then
12:            sequence.add(crime)
13:        else
14:            sequencesList.add(sequence)
15:            sequence ← []
16:            sequence.add(crime)
17:        end if
18:        last ← time;
19:    end for
20:    return sequencesList
21: end procedure
```

Figure 5.6: Build spatiotemporal sequences from items.

```
1: procedure FILTERSEQUENCE(sequences, size)
2:     seqsFiltered ← []
3:     for each seq in sequences do
4:         if seq.size > size then
5:             seqsFiltered.add(seq)
6:         end if
7:     end for
8:     return seqsFiltered
9: end procedure
```

Figure 5.7: Filter sequences based on their size.

bring no value for pattern and rule mining. For example, the sequence with one item cannot contribute anything to pattern or rules discovery. Such sequences are noise for our framework; therefore, we filter them out. As a result, we have the filtered list of sequences for further work.

5.4 Use Case Study: Spatiotemporal Crime Analytics

Our proposed framework is aimed to discover patterns and rules in spatiotemporal data. But originally, we were motivated to propose the method for spatiotemporal sequential rule mining in crime data. Therefore, in this section, we show one of the use cases, to which we can apply our framework. The use case is crime data mining. For development and test, we used publicly available crime data: from Cambridge PD [54] and from Chicago PD [55]. Data from other police departments can be found on the website *https://data.world*.

Crime datasets contain the list of crime incidents happened in the corresponding cities. Each crime item has many fields, such as report id, crime time, crime type, reporting area, neighborhood, location and so on. Among these fields, we are interested only in 3 fields: *type*, *time* and *location*. Therefore, we need to do field extraction and data normalization, in order to build ST-items - the first stage in our framework (Figure 5.2).

The crime type is used to categorize crime incidents and it has the set of predefined values, such as *theft*, *assault*, *burglary* and so on. Therefore, we just keep the values without any changes. Meanwhile, the second parameter - crime *time*, we have to parse, because each dataset has its own format for string date representation. For example, it can be 11.02.2016 15:45, 09-12.2016 4:30 PM or any other representation. Quite often, it depends on the locale representation of the date. For parsing, we use pattern-based date parsers that convert the string representation of the date into the unified date object.

Mostly, crime data do not have exact geographic coordinates of crimes, in order to protect the privacy of crime victims. Addresses are shown at the block level only and specific locations are not identified. The location can be presented as the name of the street or coordinates of the street. Such limitation brings the limitation to our framework for the considering case study, which we will discuss in Section 5.5. Additionally, because the location information can be split into different parts and stored in the different fields, in the normalization stage, we have to extract location-related fields and concatenated them, in order to build the complete full address. After that, the complete full address is used in the Geocoding module to convert it into the geographic coordinates and to build

the location entity with corresponding fields. As a result, we have the location entity with geographic coordinates (latitude and longitude) and geohash value. Therefore, after the normalization and geocoding steps, we have ST-items that are bricks to build the ST-sequences.

The next step is to build spatiotemporal sequences from spatiotemporal items. For that, we have the STS-Builder module (Figure 5.2). We have already defined the definition for the spatiotemporal sequence in the previous section. Here, we would like to rephrase the definition according to crime data. For the considering case study, we would have a crime spatiotemporal sequence (CST-sequence). By analogy with the original definition, the CST-sequence is the time order list of crimes happened close to each other in the temporal and spatial aspects. The CST-sequence is characterized with the time distance, which is the maximal time distance between crime-neighbors in the sequence, and the location area, which specifies the geographic area, within which we build sequences. The idea of building such crime sequences come from the assumption that crime incidents, which happened within the specified time and in the same neighborhood, can be associated with each other. And current considering use case is aimed to find such patterns and association rules in crime data by using the proposed framework.

Additionally, to the description of the use case, we would like to provide the description of parameters that influence on final results. All parameters (configuration of the framework) come from the fact that ST-items have 3 possible fields: *type*, *time* and *location*. For crime data that we consider, we have the set of predefined crime types. For different datasets, we have the different number of types (30-50 types). Some of them are common, such as *theft*, *violence*, *assault* and some are very specific, such as *ritualism*. You can consider all crime types and try to find patterns and rules among all the data. But, also, if you want to find out, whether there is a pattern between *theft* and *violence*, you can work only with these crime types. Adjustment of the type parameter can help to find interesting patterns and rules.

Next 2 parameters are related to the temporal and spatial aspects of data. The first one is the time distance between crimes and the second is the geographic area. With the first parameter, we define what is the maximum time that can be between crimes, in order to put them into the same sequence. It can be minutes

or hours and even days. And it depends on patterns and rules, that you try to discover. With the second parameter, we define in which geographic area we want to build sequences. If you are interested in the local geospatial sequences, you can specify the area with the radius as few meters. If you are interested in patterns for neighborhoods, then you probably can specify the area with several hundreds of meters.

Our framework is aimed to provide the flexible mechanism of changing configuration (parameters), in order to fulfill the requirements for patterns and rules discovery in spatiotemporal data. The concrete values depend on data and goals.

Data Samples

As we mentioned above we played with two crime datasets to develop our framework: crime data from Cambridge PD [54] and from Chicago PD [55]. The original data can be downloaded the from public source [55]. Generally, data are presented as a CSV file with many parameters. We are interested mostly in crime type, date/time and address/location. One of the first steps in our framework is data normalization and geocoding. We match the address with geographic coordinates. We have discussed this stage in Section 5.2.3. The sample result of that step is presented in Figure 5.8.

```
address,geohash,longitude,latitude
MARTIN+ST+&+MASSACHUSETTS+AVE+Cambridge+MA,9vfcfz7v13re,-97.2567491,32.6920583
0+Pearl+STREET+Cambridge+MA,drt3n028jyeu,-71.10257709999999,42.3646901
Harvard+STREET+&+Portland+STREET+Cambridge+MA,drt2vs61ptj2,-71.1224127,42.3428805
GARDEN+STREET+&+WATERHOUSE+STREET+Cambridge+MA,9q4gu55zsmmt,-119.7016915,34.4269758
100+Kirkland+STREET+Cambridge+MA,drt3jfthzj6z,-71.10733139999999,42.3778589
0+Blanchard+ROAD++Cambridge+MA,drt3hsrb2qq6,-71.1587713,42.3867041
CHILTON+ST+&+CONCORD+AVE+Cambridge+MA,dr5pqgpc9dnw,-74.22390279999999,40.6661628
Danehy+Park+&+New+STREET+Cambridge+MA,drt3jkdmye4p,-71.13310299999999,42.3890049
100+MOULTON+STREET+Cambridge+MA,drt3hvpx45xq,-71.1480517,42.3919686
MAGAZINE+STREET+&+WILLIAM+STREET+Cambridge+MA,9vrfjwzy66yq,-90.09911509999999,29.92098009999999
INMAN+STREET+&+MASSACHUSETTS+AVENUE+Cambridge+MA,drt3n10rt9ht,-71.1029302,42.3700681
River+St+&+Green+St+Cambridge+MA,djwqdb4dcwez,-81.0865851,32.0805863
100+REED+ST+Cambridge+MA,drt3jmfzxbh5,-71.13236010000001,42.396155
0+Whittemore+Avenue+Cambridge+MA,drt3jq2vhn52,-71.1352685,42.3985041
0+WATERHOUSE+STREET+Cambridge+MA,drt3jd9ctu94,-71.1228367,42.3772962
Broadway+&+Trowbridge+St+Cambridge+MA,dr73zk1ungzg,-73.85838059999999,41.1115472
LAWRENCE+STREET+&+Corporal+McTernan+STREET+Cambridge+MA,9xj64vq2yxhz,-104.9874692,39.7555691
0+SHADY+HILL+SQUARE+Cambridge+MA,drt3jfgvuu01,-71.1091481,42.3793962
0+EMMONS+PLACE+Cambridge+MA,drt3jfkx229q,-71.1083096,42.3768892
```

Figure 5.8: The results of geocoding module: mapping addresses to geographic coordinates.

STS-Builder (Section 5.3) builds spatiotemporal sequences. Example of such sequences, you can find in Figure 5.9. These sequences are a basement for applying pattern/rule mining algorithms. You can use any appropriate algorithm

```
CRIMINAL_TRESPASS ASSAULT
OTHER_OFFENSE BATTERY THEFT
CRIMINAL_DAMAGE ASSAULT
CRIMINAL_DAMAGE OFFENSE_INVOLVING_CHILDREN
CRIMINAL_DAMAGE BATTERY
THEFT CRIMINAL_DAMAGE BATTERY BATTERY OTHER_OFFENSE
OTHER_OFFENSE ASSAULT
CRIMINAL_DAMAGE OTHER_OFFENSE
CRIMINAL_DAMAGE CRIMINAL_DAMAGE
CRIMINAL_DAMAGE BATTERY
BATTERY DECEPTIVE_PRACTICE
ROBBERY CRIM_SEXUAL_ASSAULT
BATTERY ASSAULT CRIMINAL_DAMAGE CRIMINAL_DAMAGE CRIMINAL_DAMAGE
```

Figure 5.9: Illustration of built sequences based on defined spatiotemporal criteria.

```
WEAPONS_VIOLATION,ASSAULT,THEFT,GAMBLING ==> NARCOTICS #SUP: 3 #CONF: 0.5
WEAPONS_VIOLATION,ASSAULT,THEFT,BATTERY,GAMBLING ==> NARCOTICS #SUP: 3 #CONF: 0.5
WEAPONS_VIOLATION,ASSAULT,CRIMINAL_TRESPASS,GAMBLING ==> NARCOTICS #SUP: 3 #CONF: 0.5
WEAPONS_VIOLATION,ASSAULT,GAMBLING,OTHER_OFFENSE ==> NARCOTICS #SUP: 3 #CONF: 0.6
WEAPONS_VIOLATION,BATTERY,GAMBLING,OTHER_OFFENSE ==> NARCOTICS #SUP: 4 #CONF: 0.5
```

Figure 5.10: Detected sequential rules from experiments.

depending on a goal. For example, you can use a sequential rule miner based on ERMiner [56]. For that you define *support* and *confident* parameters. We used support = 3 and confident = 0.5. The result of our experiment is presented in Figure 5.10. In that figure, you can see that based on our defined parameters and original data set, we have a sequential rule, such as weapons violation, assault, theft, gambling could force a narcotics-related crime with 50% confidence. Playing around parameters and algorithm you can get different results.

The challenge of using the framework is to define your goal, mining algorithm and variable parameters that can give you valuable results: patterns/rules and explore hidden knowledge in your data.

5.5 Summary

We presented the framework for spatiotemporal sequential rule mining. The framework contains several modules, which have its own responsibility: data normalization, geocoding, spatiotemporal sequences building and pattern/rule mining. We explained each module separately jointly with a pseudo-code of algorithms. All modules together contribute to the framework workflow, which

is designed to work with spatiotemporal data. The proposed framework converts non-spatiotemporal data into spatiotemporal items and builds spatiotemporal sequences. Such a procedure allows applying pattern mining techniques, in order to find sequential patterns and sequential rules in spatiotemporal data.

After the framework presentation, we considered one of the use cases, to which we can apply our framework. The presented use case is the discovery of spatiotemporal sequential rules in crime data. To design this use case, we used publicly available crime data. The obtained results have some limitations because of its own data limitation. Crime data do not contain exact geographic coordinates of crime incidents to protect the privacy of victims. Such limitation of data influences the quality of results. It means that because of the not exact coordinates, the approach finds sequential rules of crimes that happen in the bigger area than we expect. By other words, we are restricted by the minimal size of the geographic area, in which we can build crime sequences and discover patterns. Therefore, we can conclude that the quality of results depends on the amount of data and the accuracy of fields, including how accurate location coordinates are.

Regardless of the data accuracy and quality, the framework can successfully convert spatiotemporal data into ST-items and ST-sequences that are the basement for applying pattern mining algorithms in spatiotemporal space. Therefore, we are interested in the further development of the framework, design new use cases and integration of other data mining techniques. Meanwhile, we want to keep the focus on the spatiotemporal data analytics. Therefore, one of the possible directions for further research is to research the possibility of applying geospatiotemporal clustering algorithms from Chapter 3 and 4, in order to discover geospatiotemporal anomalistic patterns. Additional improvement can be the integration of the geospatial data structure or database, such as MongoDB, that can provide a more flexible way to build geospatiotemporal sequences based on the geographic coordinates of data.

Chapter 6

Application for Analysis and Visualization of GeoSpatiotemporal Data

6.1 Motivation and Related Work

Nowadays, social media services produce a huge amount of publicly available data. And researchers are very interested in analyzing such data for different purposes. According to the article [57], the list of top ways researchers use social data includes different topics, such as discovering trends, analyzing and optimizing marketing, monitoring and understanding a large audience, natural disasters, media, and business. It is just a short list of possible topics. In fact, the range of possibilities for analysis of public social data is wider and it increases with increasing the popularity of social networks. Social networks are very reflective to global events and if something happens, information about it will appear in social networks almost immediately: new trends, new top hashtags, new discussions and so on.

The important moment in the development of social networks is 2009 when the location-based social network Foursquare[1] was released. Before that, users used locations in their messages, but not so often. Foursquare brought a new way of communication between people. Users started actively to use location-based

[1] https://foursquare.com

social networks (LBSN) and share their locations with friends through check-ins at venues. The popularity of LBSN affected big social networks, such as Facebook and Twitter. Users started more often to attach the location to their messages.

Georeferenced messages give more possibilities for analysis of publicly available social data because we can be interested not only in the content of messages but also in the location, where these messages are produced. Therefore, many researchers started to use georeferenced data (geotagged) in their researches. One of the most popular cases is to use georeferenced data for analyzing, detecting and predicting natural disasters. For example, Sakaki et al. [42] used Twitter users as social sensors to detect earthquake shakes. They investigated the real-time interaction of events, such as earthquakes, in Twitter and proposed an algorithm to monitor tweets and to detect an earthquake target event. Another use case of using publicly available social data to analyze natural disasters was presented by De Longueville et al. [43]. In their paper, they showed how location-based social networks (LBSN) can be used as a reliable source of spatiotemporal information, by analyzing the temporal, spatial and social dynamics of Twitter activity during a major forest fire event in the South of France in July 2009. Later in 2013, the amount of publicly available social georeferenced data increased and researchers had the possibility to detect not only global but, also, local geospatial events [44].

Another use case of using publicly available georeferenced social data is to understand user behavior around some area or to explore the area around, in order to improve situational awareness. There are many papers that focus on such topics [58] [59] [60] [61] [62].

In 2013, Kalev Leetaru et al. [63] presented their deep study of the geography of Twitter. They analyzed data from Twitter posted around the world and built many statistics, such as total tweets per day, average tweets per hour, all exact location coordinates and the top 20 cities by percent of georeferenced tweets. Also, their study includes linguistic, textual and user profiles analysis of data. In 2014, Muhammad Adnan et al. [64] presented their results of analysis georeferenced data from Twitter. In their work, they concentrated on the social dynamics of Twitter usage based on data from London, Paris and New York City. They showed the areas of tweeting activity, they provided the results of ethnicity analysis of Twitter users, a geography of tweets of different ethnic groups and

gender analysis. Another analysis of georeferenced data, Diansheng Guo et al. [65] presented in 2014. In their work, they were aimed to detect non-personal and spam users on the georeferenced Twitter network. Their approach contains extracting user characteristics, constructing training datasets, conducting supervised classification for detecting non-personal users and the evaluation of the approach. In 2015, Umashanthi Pavalanathan et al. [66] presented interesting results. They showed that young people and women more often write georeferenced tweets; users, who geotag their tweets, tend to write more, making them easier to geolocate; and the text-based geographic location is significantly more accurate for men and for older people.

Location-based social networks are very reflective of global natural disasters, such as earthquakes or forest fires. But the amount of data, produced by LBSN, increases every year dramatically, and it gives us a possibility to use these data to analyze and detect also local events. Therefore, based on the trend of increasing the popularity of LBSN and georeferenced content, we can suppose that in near future all events happening around will be reflected in LBSN and blogs. Therefore, we are interested in developing the approach for visualization and analysis of publicly available social georeferenced data that can improve situational awareness for research purposes. Another motivation for developing the application is to utilize the clustering algorithm that we proposed in the previous chapters. The proposed in previous chapters algorithms, methods and frameworks in combination with data and developed application around of that can make data analysis more efficient and results of analysis more valuable. By this way, we can propose the basement for geospatiotemporal data analytics platform.

The remainder of the chapter is organized as follows: Section 6.1.1 reflects the thesis scope and a brief overview of related applications. Section 6.2 gives the overview around data gathering and normalization that is connected to the Section 2.3. After that, we describe the implementation details including software architecture and user interface in Section 6.3. The next section (Section 6.4) is devoted for use case analysis that we do by using the developed application. The final aspect of the chapter - filtering irrelevant data that is addressed in Section 6.5. We conclude the outcome of the application, use case, and analysis in Section 6.6.

6.1.1 Thesis Scope and Related Applications

The main research focus of this thesis is to provide methods and frameworks for geospatiotemporal data analysis. The developed and presented in previous chapter methods can be used in different use cases. But in the scope of this work and use case sections, we addressed the challenge - to develop an application that can analyze and visualize in real-time publicly available social georeferenced data to provide situational and public safety awareness. This challenge requires a complex solution, therefore, during work on the project, we face many additional challenges that come from the practice that we presented in the previous chapters and will present further.

Our developed application works with publicly available social data that is as a basement for research. And nowadays, there are a lot of social analytics tools, such as TweetDeck, Twitonomy, Hootsuite, Tweepsmap, Geofeedia and so on. Many of them provide powerful functionality for analysis and visualization of data for different purposes and solve some challenges mentioned above. But mostly, they are aimed to support marketing and business that is a different focus from ours. Therefore, in the next following chapters 6 and 7, in the scope of developing the application and considering use cases, we address challenges around providing situational and public safety awareness.

6.2 Data Gathering and Normalization

Data is a basement of research and development. Therefore, the first challenge is to choose the source of georeferenced. Initially, to prove the concept, we decided to use publicly available data from Twitter. It is a quite reasonable decision because, nowadays, Twitter is something more than just a social network. Twitter is the source of world news [18] and it is the place, where breaking news about important events appear firstly. Nowadays, Twitter produces hundreds millions of tweets per day [67] including georeferenced tweets. Therefore, we can suppose that if something important happens around some area then tweets about it will almost immediately appear on Twitter.

Twitter provides a quite powerful API to fetch the required data. The only way to access 100% of all tweets in real-time is through the Twitter "Firehose",

but the access to the Twitter "Firehose" is very expensive and it can be used mostly for commercial use cases. The other option for accessing tweets is using one of Twitter's direct API offerings [20]. The main disadvantage of using free API is that Twitter does not guarantee that you will get all the requested data. But studies have estimated that using Twitter's Streaming API users can expect to receive anywhere from 1% of the tweets to over 40% of tweets in near real-time [20]. Also, you can increase the percentage of receiving tweets by applying more strict criteria. The criteria can be keywords, usernames, locations, named places, etc. For example, if you ask Twitter to provide only georeferenced tweets from some concrete city then with a high level of likelihood you will get almost all georeferenced tweets from the specified area. For our experiments, we use the area around London. Firstly, because London is one of the most active Twitter cities [22] (based on the Paris-based research firm Semiocast, 2012) and, secondly because the main language of tweets posting in London is English.

6.3 Implementation

The implementation section contains two subsections. The first one (Section 6.3.1) reflects the programming architecture of the application with technical software details. The second subsection (Section 6.3.2) show and describe the application in terms of the user interface and working interaction.

6.3.1 Architecture

In this section, we show the overview of the application's architecture that we built for visualization and analysis of publicly available social georeferenced data. The server side of the application is written in the Java programming language. You can find the architecture in Figure 6.2. The architecture is shown in simplified view with the combination of programming modules and their relations. The application architecture contains the gathering tool that we introduced in Chapter 2 in Figure 2.4. It means that the gathering tool presented in the figure contains by itself many other sub-modules that are responsible for collecting, parsing, normalizing and filtering data. The detailed overview is given in Figure 2.4 in

Chapter 2. Here we consider this module is a preprocessing step between raw data and entities in the database that are basement fur further research components.

According to the architecture figure, firstly, we subscribe the Twitter's stream, because Twitter is the source of data for our application. When we subscribe, we need to specify the criteria, and as we mentioned in the previous section, we are interested in monitoring London, since London is one of the most active Twitter cities and we can receive from there the most intensive georeferenced data. Therefore, we specify the approximate coordinates of London, which are {-0.569042, 51.247948; 0.303813, 51.727184}, where the first pair is the longitude and latitude coordinates of the south-west corner of the area and the second pair is the coordinates of the north-east corner, correspondingly.

Once we subscribed for the Twitter's stream, we start to receive tweets, which match our criteria. Messages have JSON (JavaScript Object Notation) [1] format, therefore, we have to parse and normalize them. We do it to convert them to the programming language's object with fields and values to make it easier to handle and store data in data storage. All these steps are happening in the gathering tool that we introduced in Figure 2.4. For our application, the gathering tool is also responsible for filtering data. We can filter any parameters, such as languages, coordinates, places or some keywords. At least, we need such a step, because you can receive from the stream more tweets than you expect. Some of them can be irrelevant for your search criteria. For example, you can receive tweets attached to Great Britain, but not to London. Since London overlaps Great Britain, all tweets, posted to the Great Britain's twitter place, appear also in the London's stream. Therefore, we need to filter them out, in order to exclude not considerable tweets. Later in this chapter in Section 6.5, we discuss deeper the topic of filtering. Here, we only point out that data could contain irrelevant for our use case messages, therefore, we need to filter them out.

The tweet has a lot of fields. The full structure of the tweet's object, you can find on the official website [19]. Also, you can find our diagram of the tweet's object in Figure 2.5, which we presented before. For our application concept, we do not need the full structure. Therefore, we extract only needed fields and build a limited data structure, which is presented in Figure 6.1. As you can see, the

[1] http://www.json.org

entity *Post* contains just seven fields: *id*, *tweet_id*, *date*, *coordinates*, *text*, *hashtags* (the list of hashtags), *language* (the code of language). Based on this simple data structure of the entity, we build our concept of the application. Therefore, as you can see in Figure 6.2, we save the entity to the database after normalization. And we are doing it in real-time in terms of processing data in online mode - process each incoming data without waiting for the entire dataset.

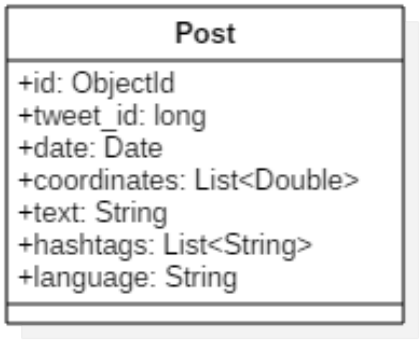


Figure 6.1: Diagram of the post entity.

To store data, we use MongoDB [1]. MongoDB is a NoSQL document-oriented database. MongoDB provides a geospatial index and suitable to store georeferenced data. Also, this database provides a possibility to store capped collections with a fixed number of entities and fixed size of the storage. Both geospatial index and capped collection feature meet our requirements to provide real-time visualization and statistical analysis of georeferenced data because we are aimed to store only the fixed number of recent tweets and provide the possibility for quick search by coordinates of the area. We store only the last 200 000 tweets posted in London. This number of posts reflects about 5 days.

Generally, MongoDB can be replaced by any other database that meets the mentioned requirement for storing georeferenced data. The more detailed motivation about choosing data storage, we provided in Section 2.5 regarding a storage for geospatiotemporal data.

In Figure 6.2, we present the software architecture overview of the application with relations among modules and external software, such as database and Twitter API. First of all, the application has full access to the database. Therefore,

[1] https://www.mongodb.org/

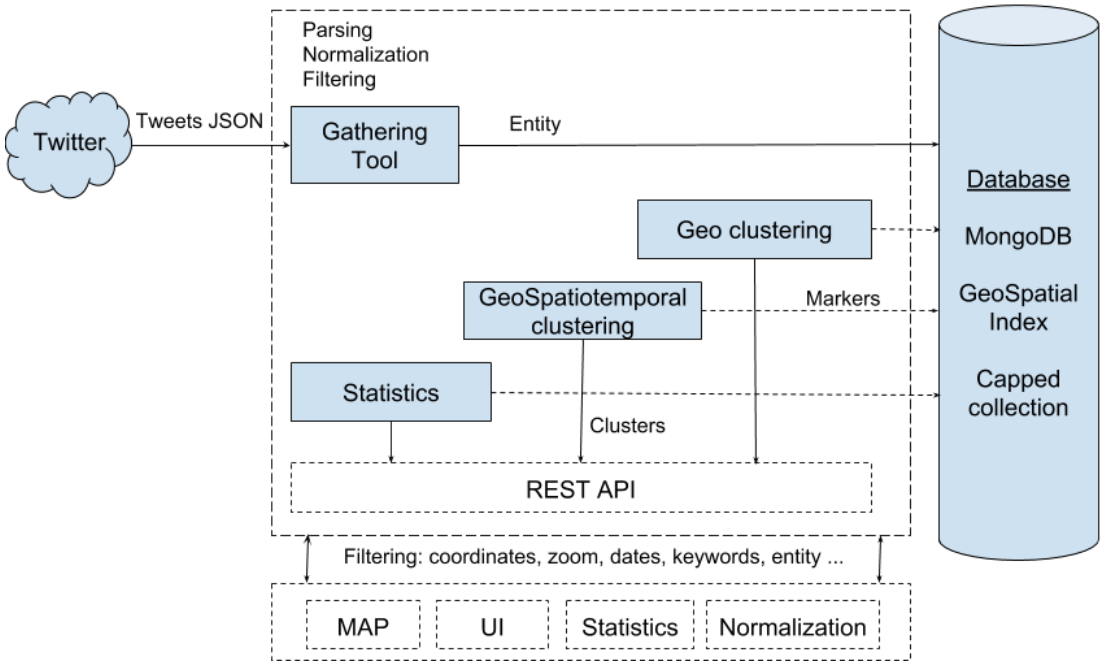


Figure 6.2: Application's architecture.

it can provide access to data through REST API. Between Database and REST API, we have clustering and statistical modules: The *Geo Clustering* module reflects the approach presented in Chapter 3 and the *GeoSpatiotemporal Clustering* module reflects the approach presented in Chapter 4. These two modules provide clustering approaches to work with geospatial data. And our application is built around these two modules to utilize their capability for data analysis. The third module is *Statistics*. The main responsibility of the modules is to calculate statistics information that will be delivered to the user interfaces, such as timeline (the intensity of messages - messages per minute), the most popular hashtags, keywords, locations and so on. This module is also responsible for providing parameter values for filtering data. We will talk about data filtering in terms of supporting analysis of situational awareness in Section 6.5.

The user interface communicates with the application through designed REST API (Representational state transfer; Application programming interface). The API allows to specify different criteria, such as coordinates, zoom, dates, keywords and so on. These criteria are used to filter the result returning to the user interfaces, as well as parameters for the configuration of clustering modules.

There is the possibility to switch between different clustering modules, but both of them do grouping the retrieved data from the database and returning the aggregated result to the client. Clustering algorithms provide the advantage for identifying anomalies. Also, the clustering is improving user experience performance in terms of showing grouped cluster data instead of thousands of data markers on an online map.

6.3.2 User Interface

This section is devoted for describing the user interface and methods, which we use to visualize georeferenced data and statistics, and by this way, to improve analysis of situational awareness. In Figure 6.3, you can find the screenshot of the user interface. As you can see, the main part of the user interface is the online map, because monitoring the area on the online map is one of the most convenient ways to monitor the situation around. We use OpenStreetMap[1]. On the screenshot, you can see the map of London. On this map, all circles reflect clustered posts, where the color reflects the weight of the cluster: red circles reflect clusters with more 100 posts, yellow circles reflect clusters with more 10 posts, green circles reflect clusters with less 10 posts and white circles reflect certain posts or clusters that contain posts with absolutely the same coordinates. You can click on the white circle and see which message is attached to the concrete coordinates. We assume [44] that if some activity happens in the area then people around this area start to post tweets describing the activity and we can see it on the online map in real-time.

To improve situational awareness, we provide some useful statistics. Below the map in Figure 6.3, you can see the timeline that reflects the intensity of posts. By this statistic, you can find anomalies in case of increasing the intensity of posts from some average value. Such anomalies could reflect that something important happens at that moment. More detailed use-cases are presented in Section 6.4.

Other statistics are the most popular hashtags and languages. Both statistics you can find on the left sidebar in Figure 6.3. It is important to show the list of the most popular hashtags in real-time, because, quite often, hashtags reflect

[1] https://www.openstreetmap.org

the trend that is happening just now in social networks. And in the combination of the most popular hashtags and geographic coordinates, we can improve our understanding of the situation. Additionally, we provide the statistics of the most popular languages. According to our statistics, the next most popular languages in London after English are Arabic (about 2% from Figure 6.5), Spanish (2%), French (1%) and Polish (1%). We can filter by some concrete language to find out the local national activities, such as national holiday celebration or culture festivals.

On the left sidebar, you can find the search form, where you can specify different parameters, such as keywords or key phrases, hashtag, language, and date range. Additionally, you can specify the coordinates of the monitoring area. Every time when you change the state of the search form or the online map by moving or zooming, the user interface sends asynchronous HTTP requests to the server: the first one to find and cluster data, the second one to calculate statistics for retrieved data. Therefore, you can see in real-time how the situation is changing in the monitored area.

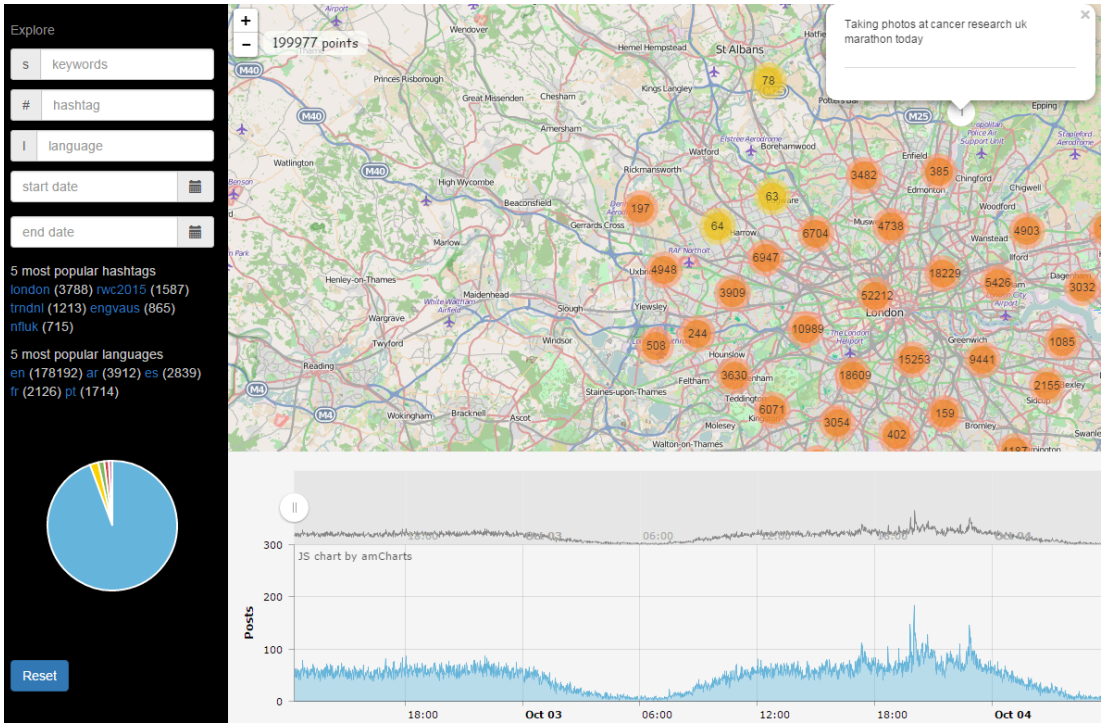


Figure 6.3: User interface.

6.4 Use Case: Visualization And Analysis of Public Social Georeferenced Data to Provide Situational Awareness

In this section, we show some use cases of using our approach for visualization and analysis of publicly available social georeferenced data. The use cases could be manifold. The application could be very useful for journalists, law enforcement, police departments or marketing departments of companies. The popularity of the georeferenced messages increases every year and we believe that, in near future, all real events will be immediately reflected in social networks, such as Twitter, Instagram, Facebook, and others. Therefore, our application is aimed to utilize georeferenced data and provide the improvement in the analysis of situational awareness based on such data.

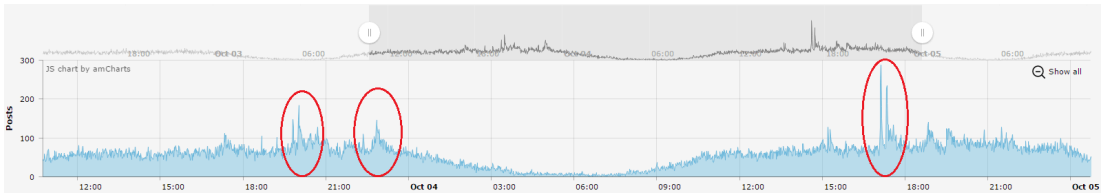


Figure 6.4: Timeline: the intensity of posts in London from the 2nd of October until the 5th of October.

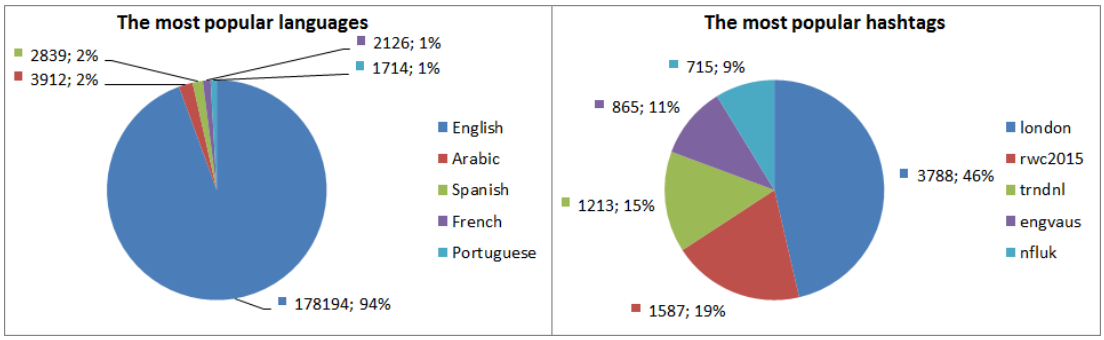


Figure 6.5: Statistics of the most popular languages and hashtags in London during the period from the 2nd of October until the 5th of October 2015.

For our use case, we use georeferenced data from Twitter that is very reflective to social geospatial events. Our application is aimed to show this geospatial reflection by providing data visualization and analysis. We collected data for 5 days (200 000) from London and analyzed them with the help of our application.

The data range for analysis is from the 2nd of October 2015 (about 12:00) until the 5th of October (about 12:00) 2015. These data are collected in the scope of our research in 2015 [4]. The timeline of the intensity of posts with selected date range - the from the 2nd of October until the 4th of October - is presented in Figure 6.4. Also, in Figure 6.5, you can find the statistics of the hashtags and languages. As can you see in Figure 6.4, there are some peaks. We consider further some of them to find out what happened at that time, what affected these peaks. But before that, we would like to draw your attention that we use time in CET (Central European Time), but time in London is CET-1. It is important in terms of proving findings based on the reflected news on the Internet.

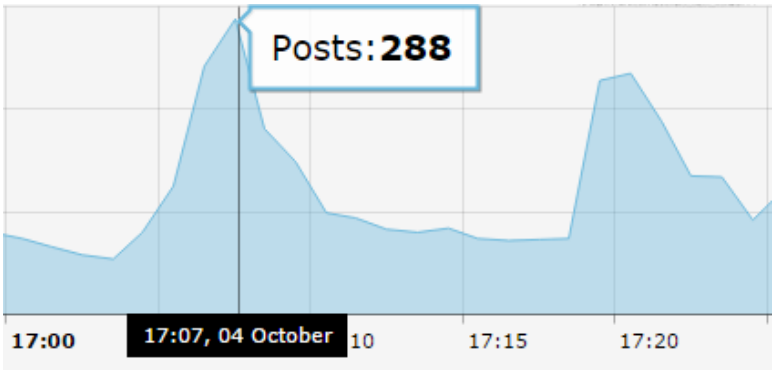


Figure 6.6: Timeline with two peaks during the football match between Arsenal and Manchester United.

Two most noticeable peaks are at 17:06-17:07 on the 4th of October (242 and 288 posts per minute) and at 17:20 on the 4th of October (235 posts per minute). We requested data only from this period of time and analyzed the content of posts and the most popular hashtags (Figure 6.6). We found out that the most popular hashtags were *#afcvmufc* and *#arsenal* and the main topic of messages was the match between two famous English football clubs: Arsenal and Manchester United. During the match the intensity of posts increased, but why there are such two high peaks? After reading news and text descriptions of the match, we found out that in that match Arsenal won Manchester with the score 3:0. And it is the surprising result. The first peak can be explained that the first goal was scored at 17:06 and after that football fans started to share their feelings and thoughts about this goal. One minute later, the second goal was

scored. Therefore, we have the double peak at the time range 17:06-17:07. But after some minutes at 17:20, the third goal was scored and it affected the second double peak at 17:19-17:20 (228, 235 posts per minute). All conclusions can be made from the content of the tweets, but to prove our guess, we found the proof in the Internet news.

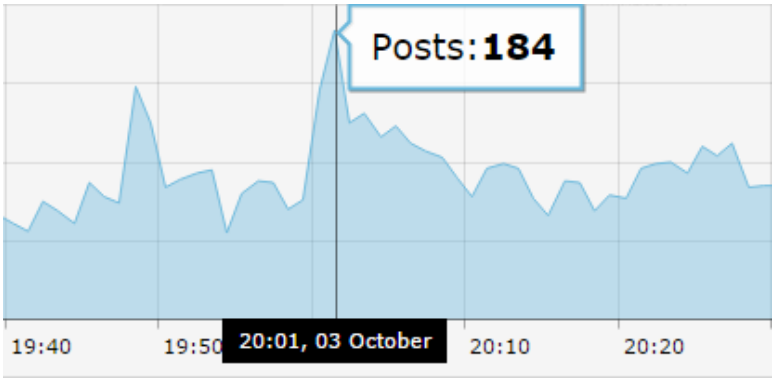


Figure 6.7: Timeline with two peaks during the football match between Chelsea and Southampton.

That weekend had one more football game - Chelsea against Southampton on the 3rd of October. The result is 1:3. Chelsea lost, and it was also surprising for football fans. Goals from that match, also, reflect the peaks of the timeline in Figure 6.7. The first anomaly was at 19:48-19:49 (148 posts) and the second was at 20:01-20:02 (184 posts), when the score became 1:2 and 1:3, correspondingly.

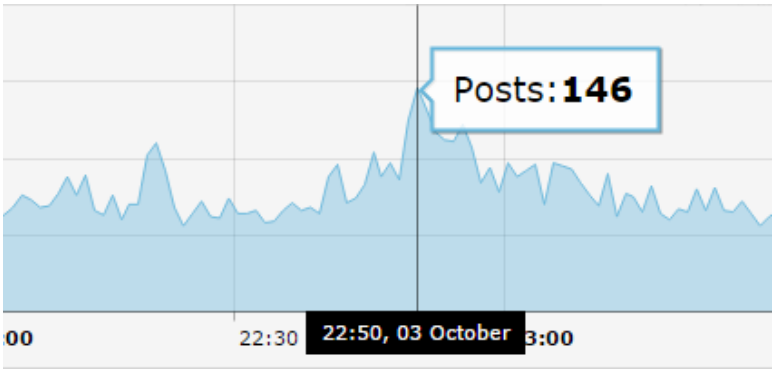


Figure 6.8: Timeline with a peak after the rugby match between England and Australia finished.

One more sport's event was on that weekend. It was Rugby World Cup

in London. And at 21:00 on the 3rd of October, there was the game between England and Australia. As you can see from Figure 6.8, there is a peak at 22:50. It is the time when the game finished. Additionally, the hashtags *#rwc2015* and *#engvaus* are one of the most popular hashtags on that weekend (1587 and 865 times are mentioned). You can find the statistics of hashtags in Figure 6.5. After analyzing the content of tweets at that time and tweets with mentioned hashtags, we can conclude that England lost the game. We proved our guess again by news from the Internet. According to the news, England lost Australia with score 13-33, and by this way, England got only the third position in the tournament table and reduce a chance to go out from the group. Such bad news for English fans was reflected in georeferenced data.

6.5 Data Filtering from Perspective of Situational Awareness

We have our own specific goal, which is the provision of situational and public safety awareness. To achieve this goal, we work with social georeferenced data that describe the situation around the place, from which they were posted. But we do not always have valid georeferenced data, and we have to work with a huge amount of data that are useless for our analysis and they do not reflect valuable information and do not describe the situation around. The challenge that we address in this section is the analysis of the value of public georeferenced data from Twitter from the perspective of providing situational awareness. In the scope of this challenge, we want to understand what and which parameters can help us to recognize (1) whether some concrete message does have any value in describing the situation around or not, (2) based on which characteristics we can fully exclude invaluable data from our dataset and (3) which statistics of data can help us to better understand the situation around. All together should support our research project. Therefore, we are aimed to collect social georeferenced data, build statistics and analyze them. Results of these statistics can help us to exclude invaluable data and develop advanced filters to support analysis of data from the perspective of providing situational awareness.

6.5.1 Data

Data is an essential part of any research. For our research, we use publicly available social georeferenced data from Twitter. Twitter provides a quite powerful API to fetch the required data. We have already discussed the capability of Twitter API in Section 2.3 about data gathering and challenges around that. In our case, we use location as a criterion because we are interested in georeferenced data, and we collect data from London because London is one of the world's most active Twitter cities [22]. To collect data, we use the Java-based application that connects to Twitter's Streaming API, specifies needed criteria and starts to receive tweets in real-time. For our experiments, we decided to collect data from the area that covers London and its neighborhood. The coordinates of the monitored area are {51.247948, -0.569042; 51.727184, 0.303813}, where the first pair is latitude and longitude coordinates of the south west and the second pair is latitude and longitude coordinates of the north east. We parse all received data, normalize them and save them into the database for further analysis. For research and further analysis, we collected 1 million tweets that cover about 12 days in 2016: from the 28th of January 11:37:30 AM until the 8th of February 12:41:48 PM. These data are collected in the scope of our research in 2016 [5].

6.5.2 Data Analysis

This section is the main part of the chapter. In this section, we provide different valuable statistics and full description and analysis of them. Also, we try to find out how our results could be used to support the analysis of situational and public safety awareness. It means that based on our statistical results, we try to propose (1) methods for reducing the amount of data by excluding (removing) invaluable data and (2) filters that can support analysis of georeferenced data. Both proposals (excluding data and filters) are aimed to support analysis of situational awareness based on social georeferenced data.

Table 6.1: Tweets' languages statistics.

English	Undefined	Spanish	Arabic	Portuguese	French	Others
83.80%	6.76%	1.66%	1.37%	1.16%	0.92%	4.33%

We start with the statistics in Table 6.1. In this table, you can see the most popular languages of tweets posted from London. Information about languages we take from the tweet object, which is provided by Twitter [19]. Firstly, we can see that the official language is in the first place. About 83.80% of all tweets are posted in English. Then we can see that for 6.76% of tweets, Twitter was not able to identify the language. In most cases, it means that tweets without the language do not contain sentences or even words. They are just a set of hashtags, user mentions, symbols, and URLs. So, such tweets do not have semantic sense. In the next places, we have Spanish (1.66%), Arabic (1.37%), Portuguese (1.16%), French (0.92%) and so on. Now, we need to consider how we can use obtained information to support analysis of data. Firstly, we definitely can exclude all tweets with the undefined language, because they bring no sense and we can not use them to analyze situational awareness. By this way, we can reduce the amount of data by 6.76%. Another thing that we can do, we can use the knowledge about used languages to filter dataset by languages or filter dataset from tweets in other foreign languages. It can support the analysis of data because we would fetch only relevant data.

Table 6.2: Tweets' place types statistics.

City	Admin	Country
79.08%	18.50%	2.42%

Usually, tweets contain the place information, because users are asked to attach a geographic place to the tweet before to publish it. Additional, Twitter asks users to attach exact coordinates. So, the final tweet could have the exact geographic coordinates and attached place. Places could be *city*, administrative area (*admin*), *country* or some concrete place of interest (*poi*), for example, Big Ben in London. In our dataset, almost all tweets have attached places (only 5 tweets do not have a place), but only 11.53% of data have exact coordinates. In Table 6.2, you can see which place types are usually attached to tweets. In the first place we have the *city* with about 79.08%, then the *admin* (administrative area) with about 18.50% and in the last place we have the *country* place type with about 2.42%. In our research, we are interested only in the tweets that have

the *city* place type or narrower as a place of interest (*poi*). Place types, such as *country* and administrative area (*admin*), are too big areas and we can not use such data for analyzing situational awareness. It means that we can easily exclude such data from our dataset, but with one assumption. We exclude tweets with irrelevant place types only if these tweets do not have exact geographic coordinates. Because if the tweet contains exact geographic coordinates, we should not consider to which place the tweet is attached. So, if we exclude tweets with *country* and *admin* place types, we can reduce dataset by maximum 20.92%.

Table 6.3: Statistics of external sources of tweets.

twitter.com	instagram.com	bit.ly	swarmapp.com	goo.gl	trendinalia.com
8.46%	6.54%	1.51%	0.98%	0.47%	0.44%

Not all content published in Twitter is original. Some tweets come from other social networks or have links to external websites or services, such as Facebook, Instagram, Swarm (Foursquare) and so on. If the tweet contains several links, we consider only the last link, because usually the last link refers to the original source of information. According to Table 6.3, the main external source of tweets is twitter.com (8.46%). It means that users post tweets that contain links to other tweets. Then we have instagram.com (6.54%), bit.ly (1.51%), swarmapp.com (0.98%) and so on. We can suppose that if a user posts a tweet about what happens around him, he does not include links to external websites into the tweet, otherwise, the user likely does not really describe the situation around him. But this statement does not work if the tweet has a link to another social network, because, in this case, it could be that user posted about the situation around in Instagram but then he reposted the message to Twitter. Therefore, it is not always obvious, which data we can exclude from the dataset. It requires more detailed analysis. But some of them we can definitely exclude. For example, we can exclude tweets that refer to Foursquare, Swarm or Yelp, to social networks that have nothing with describing the situation around. Usually, tweets, which refer to Swarm, have information, such as "Hi, I am in London" or something similar to it. So, if we exclude tweets at least only from Swarm, we can reduce dataset by about 0.98%.

Table 6.4: Statistics of the most demanding geographic coordinates.

gcpe6rh4k4j9	gcpvjc9kxvpg	gcpuuqtyeztv	u120jz6zfbbe	gcpusn4djt0k
5.79%	5.75%	4.93%	4.33%	4.15%

We have 1 million tweets but only 11.53% of tweets have exact geographic coordinates, others have geographic coordinates based on an attached geographic place (the center of the place). It means that if tweets have the same attached places then they have the same coordinates. Based on our dataset, we calculated that from 1 million possible geographic coordinates, we have only 37650 unique geographic coordinates, which is about 3.76% of all possible coordinates. This percentage is less than the percentage of tweets with exact coordinates. So, we can conclude that tweets with exact geographic coordinates can have the same coordinates. Usual case for that is when tweets are reposted from other social networks. Additionally, we analyzed which geographic coordinates are the most demanding. In Table 6.4, you can find the calculated statistics. The coordinates are presented in geohash[1] form to simplify the representation. They can be easily converted back to latitude and longitude by the geohash function. According to our statistics, about 5.79% of tweets have coordinates *gcpe6rh4k4j9* (51.23513843, -0.59857568), which is in the area of the city Guildford. The next, about 5.75% of tweets have coordinates *gcpvjc9kxvpg* (51.51294588, -0.09681718), which is the center of London. By these statistics, we can see which places are the most attractive by Twitter users. Above we mentioned that about 79.08% of tweets contain attached places of the *city* type. But these places are different. If a user posts a tweet in London and he attaches the place as London, then his tweet would have coordinates of the center of London. But if he attaches some concrete district of the city then the tweet would have a more concrete location. For example, such district of London can be Barnet, Hackney, Lambeth and so on. Actually, Twitter suggests the closest and the most appropriate place when a user wants to attach the place to the tweet, and it helps more correctly determine the location for the tweet even if a user did not attach the exact geographic coordinates. Now, the question is what we can do with these statistics to support analysis of situational

[1] https://en.wikipedia.org/wiki/Geohash

awareness. We can use these statistics to design a filter. This filter would use statistics from Table 6.4 and provide filtering data from tweets, which were posted from the most demanding exact geographic coordinates. In some cases, it can facilitate visual analysis of the situation.

In Section 6.2, we mentioned that we subscribe for data from London and its neighborhood defined by the following coordinates {51.247948, -0.569042; 51.727184, 0.303813}. But Twitter usually returns not only data from the specified area but also from areas that overlap that area. For example, we receive, also, tweets with coordinates of the entire country UK. Therefore, additionally, we calculated the percentage of tweets that have coordinates inside of the specified area. The result is 82.51%, which is close to the percentage of tweets containing a *city* place type. If we exclude tweets outside the monitored area, we can reduce the amount of data by 17.49%. But this percentage will be less, if we, firstly, exclude tweets with inappropriate place types *admin* and *country*.

At the beginning of this section, we have mentioned that some tweets contain location information, such as hashtags or words of locations, and they can be geolocated. Therefore, we want to find out how many tweets in our dataset contain additional location information. In some cases, it can help to identify the location more precisely than it is specified. For example, we could have the tweet "The house is on fire in Carnaby Street" and this tweet could have an attached place as the entire London, which has coordinates of the city center. But in this case, this message would be more valuable if it would have a more concrete location. We can see this concrete location in the text message "in Carnaby Street". Therefore, we want to find out how many tweets in our dataset contain such additional location information. To find out it, we used Stanford Named Entity Recognizer (NER) [16]. And after applying this library to our dataset, we obtained that about 9.51% of tweets contain additional location information in their text content. It means that potentially 9.51% of tweets could be more accurate geolocated than it is specified.

Table 6.5: The most active Twitter users.

tegrenade	hesjkr94	trendinaliagb	orgetorix	blankiiam015	don_jide	a_rockas
1.40%	0.61%	0.50%	0.26%	0.23%	0.22%	0.22%

The next thing, in which we are interested in, is how many unique users we have. We calculated that in our dataset we have 107105 unique users, and it means that we have about 9.37 posts per user. Not all users are equally active. Therefore, we want to find out which users produce more tweets than others. In Table 6.5, you can find partial statistic results. In the first place, we have user *tegrenade*, who produced about 1.40% of all tweets from our dataset. It is about 14004 tweets for 12 days, which means 1167 tweets per day. The Twitter accounts in the next palaces produced also a huge amount of data, you can see it in Table 6.5. Some of these active users we can consider as a non-personal or spam users. It is an interesting challenge to identify it and Diansheng Guo et al. [65] presented their approach of detecting non-personal and spam users. But in the scope of this chapter, we would like to use found statistics to design a filter. And this filter would be aimed to filter data from the most active users by using statistics from Table 6.5. In some case, it can facilitate the analysis of data.

Now, we consider hashtags, user mentions, symbols, URLs and media objects in tweets, and how they are mentioned by users. We start with symbols. The symbol is the character started with *$* dollar sign and it is often used in the financial area, for example, to put price information or to include the name of the company in the stock market. Examples of symbols are AMZN, GOOG, FB, GBPUSD, where AMZN - Amazon, GOOG - Google, FB - Facebook, and GBPUSD - British Pound to Dollar. So, we can suppose that tweets with symbols have nothing with describing the situation around, therefore, we can exclude them. But they constitute just about 0.01%, so the benefit is small.

Table 6.6: The percentage of tweets that contain URLs in their text content.

has URL	1 URL	2 URLs	3 URLs	4 or more URLs
25.70%	23.96%	1.71%	0.03%	0.003%

Many tweets contain URLs. According to our statistics, about 25.70% of tweets contain URL. Mostly, tweets contain only one URL (23.96%), but some of them have more links. Details statistics you can find in Table 6.6. Such URLs could be links to other social networks or to some external websites with some news. We can suppose that if a user posts the tweet about what he sees now,

likely, he will not include a link to the external resources, but with the exception when user reposts the tweet from other social networks. Therefore, we can assume only one or maximum two links. Others tweets we can exclude from the dataset.

Some tweets contain media objects: image or video. According to our statistics, 14.24% of tweets contain media object, and 14.238% of them contain only one media object and 0.002% of them contain 2 media objects. Media-based tweets are produced by 35.81% of users. So, it is common to include one media object to the tweet, but the existence of media objects in tweets does not tell us about the relevance of the tweets from the perspective of describing the situation around. Therefore, we should not consider this parameter for filtering data.

Table 6.7: The percentage of tweets with user mentions in their text content.

has mention	1 mention	2 mentions	3 mentions	4 or more mentions
51.68%	36.35%	9.58%	3.12%	2.63%

Users often mention other users in their tweets to start or to keep discussion. For that, they use @ character and the account name of the user. From Table 6.7, we can see that about 51.68% of tweets contain user mentions. We could suppose that the number of user mentions can affect how the tweet describes the situation around. More user mentions in the tweet then more likely that this tweet is just a part of the discussion, but not describing the situation around. Therefore, we are interested in a filter that can filter data by the number of user mentions. For example, we could want to filter data from tweets that contain 3 or more user mentions. It can reduce dataset by 5.74% for further analysis.

Table 6.8: The percentage of tweets with hashtags in their text content.

has hashtag	1 hashtag	2 hashtags	3 hashtags	4 or more hashtags
22.71%	12.04%	5.04%	2.23%	3.40%

In Table 6.8, you can see the statistics of hashtags in tweets. According to our results, about 22.71% of tweets contain hashtags. About 12.04% of them contain only one hashtag, about 5.04% of them contain two hashtags, about 2.23% of them contain 3 hashtags and about 3.40% of them contain 4 or more hashtags. We can suppose that when a user wants quickly to post a tweet about

the situation around, he would not consider about including a huge number of hashtags into the tweet. But we can not fully rely on this statement. We should consider another additional parameter.

Table 6.9: The count of hashtags and the count of days when these hashtags appear.

12 (100%)	11 (91.66%)	10 (83.33%)	9 (75.00%)	8 (66.66%)	7 (58.33%)
799	580	548	606	720	988

In Table 6.9, we provide the statistics of the most utilized hashtags during the time periods of collected data. 799 hashtags appear every day (12 days) in our dataset, 580 hashtags appear in 11 days (91.66% of the entire date range), 548 appear in 10 days (83.33%) and so on. Some hashtags, which appear every day, could be useless hashtags from our perspective, for example, the hashtag *#happybirthday*. Whereas among these hashtags, there are hashtags that represent geographic places, for example, *#london*. The hashtag *#london* appears every day, but we assume that this hashtag can be included in the tweet to describe the situation around. Therefore, we need to consider an additional parameter that can help us to understand the value of tweets with popular hashtags. Therefore, we built the statistics of the distribution of the most popular hashtags among users. You can find this statistics in Table 6.10. From that statistics, we can see that, for example, the hashtag *#london* is used by 5.07% of users. This hashtag is used by the significant number of users and it appears every day, therefore, we can not exclude tweets with this hashtag. Whereas, the hashtag *#happybirthday*, which appears also every day, is used only by 76 users (0.071% of users). This fact could give us a guess that such hashtag does not bring any situational information into the tweet. Therefore, we can think about filtering them.

Table 6.10: Statistics of usage concrete hashtags by the percentage of users.

london	love	cbb	superbowl	uk	sb50	art	fridayfeeling
5.07%	0.71%	0.70%	0.60%	0.53%	0.52%	0.50%	0.49%

We have three parameters related to the hashtags: the number of hashtags in the tweet, the distribution of hashtags during the date period of the dataset and the distribution of hashtags among users. We want to filter tweets that do

not describe the situation around. For that, we need to use all three parameters. Therefore, our filter should assume that an invalid tweet should have more than n hashtags, at least m of these hashtags should appear every day during the monitored period d (in our case it is 12 days, but it can be customized), and less than p percents of users should use these m hashtags. We need to point out that we do not consider which values of parameters n, m, p, d to choose to obtain the best filtering results, but we have only shown the parameters that should be considered. Finding the concrete optimal values is a part of future work.

6.5.3 Design the Filter: Discussion

In the previous subsections, we analyzed the value of social georeferenced data from the perspective of providing situational awareness. We were motivated to do it because we have to work with a huge amount of invaluable data, and we wanted to reduce this amount of such data. During work on the analysis of data, we built statistics by different parameters and different compound parameters that gave valuable information about which kind of filters can exclude invalid data.

It is always important to point out that we do the analysis of publicly available social georeferenced data from the perspective of providing situational awareness. Therefore, we consider our built statistics from that perspective and in the scope of our research application (Section 6.3). It makes our research different from many existing social analytics tools focused mostly on business and marketing. For example, in the statistics of the most popular languages in Table 6.1, we tried to find out how this statistics can help us to remove partially invaluable data. And we found out that the indicator of invaluable data, from the perspective of providing situational information, can be the *undefined* language. We went further and we tried to heuristically find out how the number of different entities in the text content can affect the situational value of data. We had an assumption that if the tweet has many URLs, user mentions, hashtags and symbols then such tweet has less the situational value. Based on this assumption, we built statistics by mentioned parameters that showed us what the benefit we can obtain if we exclude too littered tweets. Another example of analysis of georeferenced data from the perspective of providing situational awareness is the statistics of

original sources of tweets. We found out that some sources of tweets can be indicators of invaluable data, but it requires manual analysis and making the list of irrelevant sources, such as Swarm and Foursquare, which do not bring situational information in their content.

To evaluate the benefit from the analysis of data and built statistics, we applied some recommendations for our research project. If we remove data with inappropriate place types (*country*, *admin*), data outside the specified monitored area, data from irrelevant services: Swarm and Foursquare, data with symbols and data with undefined languages, then we exclude 28.86% of invaluable data. Meanwhile, we can expect the higher percentage of removed invaluable data, if we apply more advanced methods for filtering based on compound parameters from built statistics.

6.6 Summary

In this chapter we presented 3 aspects of the research: (1) design and implementation of research application, (2) consideration of the use case and (3) detailed data analysis with proposals for data filtering. The application combines the implementations of proposed in the previous chapters methods for clustering and provide the platform for geospatiotemporal data analytics. This application includes also the gathering modules introduced in Section 2.3 and utilizes its power of data normalization. The developed application has a focus - providing situational awareness based on publicly available georeferenced data. Therefore, in the second part of the chapter, we considered the use case and showed its analysis by using the developed application. The third challenge came from the working analysis practise around collected data. This challenge is about filtering invaluable data. For that challenge, we analyzed the collected data, provided statistical data based on different characteristics. We discussed proposals for filters based on statistical results. Such filters are aimed to filter not relevant data from the perspective of providing situational awareness.

The proposed in this chapter application is the initial stage and it was originally developed for utilization of proposed in this thesis methods and algorithms. But the application can be improved and developed further. The general way of

improvement is including more various methods and algorithms for data analysis. Also, we can include more sources of data for analysis. Another direction for further work could be considering additional use cases. One of such use cases, we consider in the next chapter.

6. APPLICATION FOR ANALYSIS AND VISUALIZATION OF GEOSPATIOTEMPORAL DATA

Chapter 7

Use Case Study: Analysis of Data from the Twitter Account of the Berlin Police for Public Safety Awareness

7.1 Motivation and Related Work

In the previous chapter, we considered the use case - analysis and visualization of publicly available social georeferenced data to provide situational awareness. In the scope of this chapter, we consider another related use case - providing public safety awareness based on publicly available social data. Our focus is on public safety awareness of Berlin. Therefore, we gather relevant to this location publicly available data from the social network. One of the sources of such data is the Twitter account of the Berlin Police (*@PolizeiBerlin_E*). The police use it to post tweets about incidents that happen in the city. Analysis of data from this account is the next step of the research project and it is the challenge that we address in this chapter.

7.1.1 Related Work

Analysis of publicly available social (including georeferenced) data is a big topic and there are many papers and enterprise solutions, which cover different aspects

and challenges of the social data analysis. In this subsection, we provide (1) the overview of the papers related to the analysis of public safety and extreme emergency events and (2) the overview of papers that used data from the Twitter account of the Berlin Police for analysis.

We have already mentioned several times that nowadays social media is the source of recent news and updates. Therefore, government agencies show a significant interest in the use of social networks for interaction with people on a daily basis and, especially, during extreme events, emergencies, and disasters. In 2010, Heverin et al. [68] showed how city police departments in large U.S. cities use Twitter and what type of information they share. In 2012, Bruns et al. [69] presented the detailed report that showed crisis communication on Twitter in the 2011 South East Queensland Floods. After that in 2012, Ehnis et al. [70] used the study case of the Queensland Floods to provide a general understanding how we might develop a standardized framework of adoption of social media for disaster management. They showed that the strength of social media for two-way communication and collaboration with the general public was underutilized during the floods. Later in 2013, Ehnis et al. [71] presented the paper, in which they analyzed the case of the Boston marathon bombing, in order to understand the social media behavior of an emergency service agency. In the paper, they compared obtained results with the results of their previous work about Queensland Floods [70], in order to highlight the influence of the disaster topology on social media communication.

The data from the Twitter account of the Berlin Police (*@PolizeiBerlin_E*) have been partially analyzed in some previous papers [72] [73]. Mirbabaie and Ehnis et al. [72] analyzed the communication roles in public events. For that, they analyzed the Twitter data related to the 1st May 2014 event (Labour Day) in Germany. In their analysis, the account of the Berlin Police was one of five primary roles during the 1st May event and tweets from them were most retweeted very frequently. They showed the primary role of the Berlin Police in informing users about the happening event. Later in December 2014, Ehnis and Mirbabaie et al. [73] published another paper, in which they presented their analysis of the role of social media network participants in extreme events. In that paper, they again used data from the 1st May for analysis and they analyzed

the network behavior and perception of the police role during the 1st of May event. The analysis showed the major role of the police during the 1st May event.

From the papers of previous years including mentioned in this section, we see how social data are valuable from the perspective of public safety awareness and what we can obtain from analysis of them. Also, we have seen that social network accounts of emergency agencies play a major role in informing users about incidents and, especially, during the extreme events. The Twitter account of the Berlin Police is an example of such accounts. Data from this account have been partially analyzed in previous papers, but the analysis was very specific and limited by the concrete use case. We claim that more deep analysis of these data can bring new research findings and research results in the scope of analysis of public safety in the city. Therefore, in this chapter, we provide our research analysis of data from the Twitter account of the Berlin Police. The methods proposed in this thesis earlier would help us to achieve such use case analysis.

7.2 Analysis

We skip the data gathering section because we use the same gathering tool as in the previous chapters. The workflow of the gathering tool is fully described in Section 2.3. We need to mention that we collected overall 5021 tweets and they cover the range from 29th of April 2015 until the 19th of September 2016.

Our data analysis is split into 2 sections. The first section reflects the overall analysis of the entire data. The second section with 2 subsections reflects the analysis of the *#24hPolizei*[1] marathons: the 12-13th June 2015 and the 27-28th May 2016.

7.2.1 Analysis Part 1 - Overall

The overall size of data is 5021 tweets and they cover the date range from the 29th of April 2015 until the 19th of September 2016. For these data, we apply normalization methods, in order to recognize and extract location entities and incident-related keywords. For location entity recognition, we use the Stanford NER tool [16] and dictionary-based entity recognizer (ER), which contains the

[1] https://de.wikipedia.org/wiki/24hPolizei (German)

names of S-Bahn[1] and U-Bahn[2] stations. Overall, we recognize location names for about 59.37% of tweets. Also, we apply the method for recognition and extraction of incident-related keywords. For that, we use the dictionary ER that contains the set of keywords grouped into the 4 categories: Public Disorder, Traffic Accident, Theft, Break-in. These categories reflect 4 major types of tweets posted by the police in the campaigns and marathons. Also, we have some keywords that reflect possible incidents, but they do not belong to one of the 4 predefined. For such keywords, we have the Other category. And as a result, 61.08% of tweets are categorized into incident-tweets (tweets about incidents) of different types. Additionally, because sometimes tweets about incidents have information about injured people, we calculate how many tweets have keywords that reflect the injured. In German, such words are Verletzung (English: injury) and verletzen (English: hurt). In overall data, we have 5.45% of incident-tweets that may report about injured people. In Table 7.1, you can find the described above statistics.

Table 7.1: Statistics of data.

Statistics	Overall	#24hPolizei 1	#24hPolizei 2
Days	509	1	1
Tweets	5021	1129	1070
Tweets with Location	2981 (59.37%)	1007 (89.19%)	894 (83.55%)
Tweets with Incident	3067 (61.08%)	628 (55.62%)	562 (52.52%)
Incidents with Injured	167 (5.45%)	39 (6.21%)	34 (6.04%)

We calculate the frequency of the incident-related keywords, which we use for the categorization of tweets. The top 5 utilized keywords are presented in the list below (the first three words are excluded because they are hashtags of the campaigns). In the list, you can find the German keyword, English translation, the count of tweets with this keyword, the percentage of tweets over the incident-tweets and the incident type to which the keyword belongs.

1. Tat(-er) (offender) 200 (6.52%) - Public Disorder

[1] https://en.wikipedia.org/wiki/Berlin_S-Bahn
[2] https://en.wikipedia.org/wiki/Berlin_U-Bahn

Table 7.2: Statistics of incident types.

Statistics	Overall	#24hPolizei 1	#24hPolizei 2
Theft	*797 (25.99%)*	102 (16.24%)	52 (9.25%)
Traffic Accident	776 (25.30%)	81 (12.90%)	96 (17.08%)
Break-in	581 (18.94%)	50 (7.96%)	19 (3.38%)
Public Disorder	563 (18.36%)	*256 (40.77%)*	*274 (48.76%)*
Other	350 (11.41%)	139 (22.13%)	121 (21.53%)

2. Verletzt (injured) 157 (5.12%) - Any type
3. Dieb (theft) 145 (4.73%) - Theft
4. Laut (loud) 130 (4.24%) - Public Disorder
5. Verkehrsunfall (traffic accident) 123 (4.01%) - Traffic Accident

In Table 7.2, we present the results of the tweet categorization. In the overall data, the major part of categorized tweets belongs to the Theft and Traffic Accident incident types. Together they make up more than the half of categorized tweets. Then we have Break-in and Public Disorder types of incidents. Other (tweets with no categorized keywords) is 11.41% of data.

In Table 7.3, we present the top 5 location names utilized in tweets. All location names in the table are the names of the districts in Berlin. You can see that in the first place we have Mitte 263 (8.82%), which is the city center. Then we have Charlottenburg 201 (6.74%), which is the district in the western part of the city. Next, we have Spandau, Kreuzberg, and Neukoln. Kreuzberg[1] and Neukolln[2] are known for its large number of bars and clubs. Additionally, we need to mention that for the statistics we do not count tweets that have Berlin as a location name because Berlin is the too broad area. If the tweet has several location names and one of them is Berlin then we consider only others. If the tweet has two location names and one of them is Mitte then we consider only the second one, because, in this case, the tweet usually says about Mitte (center) of some district, which is the second location name in the text.

[1] https://en.wikipedia.org/wiki/Kreuzberg
[2] https://en.wikipedia.org/wiki/Neukolln

Table 7.3: Statistics of top locations in Tweets.

Overall	#24hPolizei Marathon 1	#24hPolizei Marathon 2
Mitte 263 (8.82%)	Charlottenburg 108 (10.72%)	Mitte 89 (9.95%)
Charlottenburg 201 (6.74%)	Mitte 76 (7.54%)	Charlottenburg 60 (6.71%)
Spandau 156 (5.23%)	Kreuzberg 73 (7.24%)	Spandau 59 (6.59%)
Kreuzberg 152 (5.09%)	Neukolln 60 (5.95%)	Neukolln 52 (5.81%)
Neukolln 139 (4.66%)	Spandau 59 (5.85%)	Wedding 48 (5.36%)

Table 7.4: Statistics of incident types to locations: Part 1.

Theft	Traffic Accident	Break-in
Mitte 40 (5.01%)	Charlottenburg 32 (4.12%)	Charlottenburg 19 (3.27%)
Charlottenburg 29 (3.63%)	Mitte 19 (2.44%)	Spandau 16 (2.75%)
Neukolln 21 (2.63%)	Spandau 18 (2.31%)	Mitte 14 (2.40%)
Spandau 17 (2.13%)	Kreuzberg 15 (1.93%)	Steglitz 13 (2.23%)
Alexanderplatz 17 (2.13%)	Reinickendorf 12 (1.54%)	Pankow 10 (1.72%)

As the next step, we analyze the correlation between locations where incidents happen and incident types. In Table 7.4 and 7.5, you can find the statistics of incident types to locations. Every cell represents the name of the location, the number of incidents of the concrete type and, additionally, the percentage of them for all incidents of this concrete type (Table 7.2). We can see that Mitte is on top of the Theft incident type. For other incident types, the top location is Charlottenburg. We can notice that for Theft, Traffic Accident, and Public Disorder in the first 4 places, we have the same set of locations as in Table 7.3. New locations are in the 4th and 5th positions of Break-in and in the 5th position of Theft, Traffic Accident, and Public Disorder. Also, we would like to notice that Alexanderplatz is the very popular place among local people and tourists in the eastern part of Berlin, therefore, there are no doubts that thefts quite often happen there, and it is shown by our statistics.

In Figure 7.1, you can see another view to the statistics of locations to incidents. The figure presents the bar chart of the top 10 locations where incidents happen. Additionally, the chart contains Alexanderplatz because this location

Table 7.5: Statistics of incident types to locations: Part 2.

Public Disorder	Other
Charlottenburg 42 (7.46%)	Mitte 26 (7.83%)
Mitte 36 (6.39%)	Kreuzberg 17 (5.12%)
Spandau 33 (5.86%)	Neukolln 16 (4.81%)
Neukolln 30 (5.32%)	Charlottenburg 14 (4.21%)
Wedding 27 (4.79%)	Spandau 14 (4.21%)

is in the top of the Theft incident type (Table 7.4 and 7.5) and Friedrichshain[1], which is known for its many bars, clubs, pubs, and cafes. In the figure, you can visually see the distribution of the incident types and the total number of incidents in locations. Firstly, we can see that Charlottenburg and Mitte are in the first places by the number of incidents. Charlottenburg dominates in 3 types of incidents. Mitte dominates in Theft over others locations (Table 7.4 and 7.5). We can see that in many locations Public Disorder dominates over other types. But meanwhile, we want to highlight that in Alexanderplatz, thefts have the significant percentage over other types of incidents (85%) in this location. Also, we can see that Friedrichshain has a small percentage of break-ins relative to other types of incidents.

7.2.2 Analysis Part 2 - #24hPolizei Marathon

The Berlin Police once per year run the Twitter *#24hPolizei* marathon. During this marathon, which goes 24 hours, the police post tweets with the hashtag *#24hPolizei* about incidents happening in the city. Many of these tweets come from the calls to the police emergency telephone number - 110. The goal of this marathon is to show the police work, motivate Twitter users to cooperative work, and as a result, provide public safety awareness for inhabitants. Overall, there were 3 *#24hPolizei* marathons, but our collected data cover only two of them: the second (in 2015) and the third (in 2016). Therefore, we split this section into 2 subsections, and in each of them, we provide the analysis of the marathon.

[1] https://en.wikipedia.org/wiki/Friedrichshain

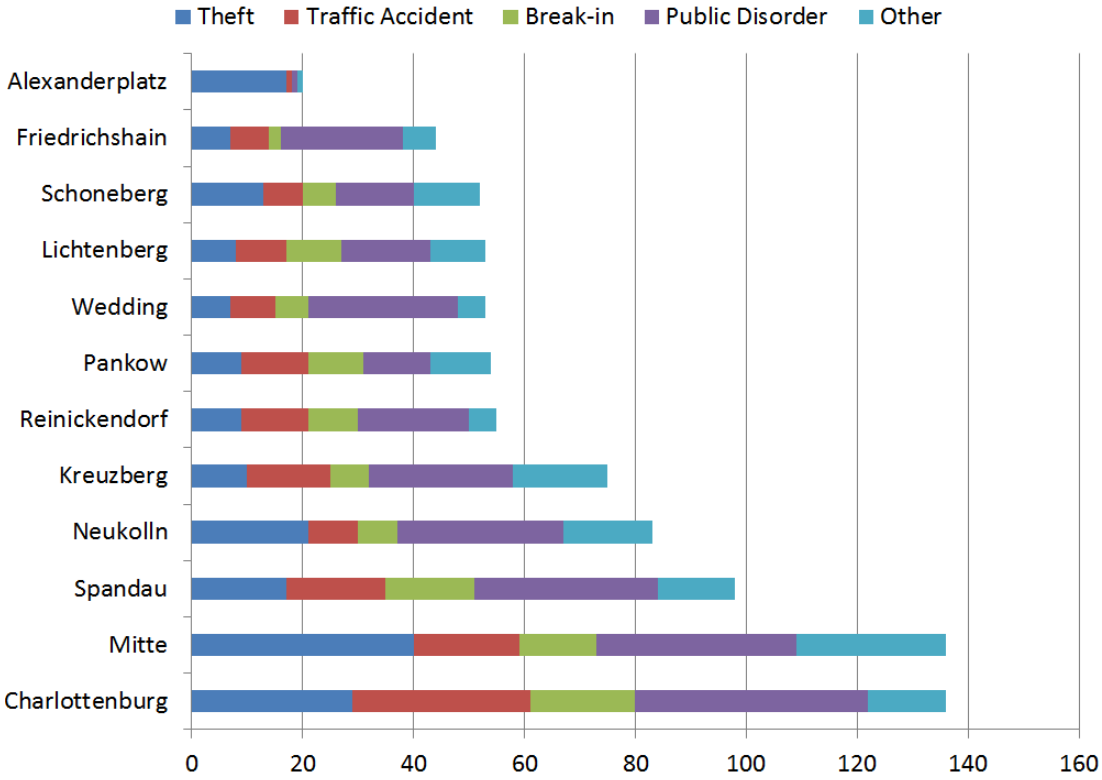


Figure 7.1: Distribution of the incident types in top locations.

7.2.2.1 Analysis Part 2.1 - #24hPolizei Marathon 1

The first Twitter *#24hPolizei* marathon in our dataset is from the 12th of June 2015 19:00 until the 13th of June 2015 19:00. Overall, we have 1129 tweets and 89.19% of them have location names recognized by the Stanford NER tool and dictionary ER. The percentage of the incident-tweets, recognized by our dictionary of keywords, equals 55.62%. The percentage of incident-tweets with possible injured people is 6.21%. This statistics is presented in Table 7.1. Also, in Table 7.2, you can find the statistics of the distribution of the incident types during the marathon. The leader is Public Disorder that differs from the Overall statistics. It takes around 40.77% of incident-tweets.

As for overall analysis in the previous section, we provide the top 5 locations, which are mentioned in tweets during the marathon. This top you can find in Table 7.3. We can see partially the same names of locations as for the overall statistics but in different positions. Charlottenburg is the first and Mitte is

the second. Kreuzberg saved its position and Neukolln and Spandau exchanged positions.

Since the marathon covers 24 hours, we have a chance to build the chart to find out the intensity of tweets by hours for different types of incidents. You can find this chart in Figure 7.2. Despite the fact that the marathon officially starts at 19:00, we have very few tweets from 19:00 until 22:00. But starting from 23:00 we have a big peak of the Public Disorder type of incidents. Then all types go down and from 5:00 until 8:00 we have the low intensity of incidents. We want to notice that from 12:00 until 13:00 there are not Break-in incidents. It is a common time for lunch and people are usually indoor. The next peak is at around 14:00. At this time all types of incidents increase. From the figure, we can see that the dominating type of incidents is Public Disorder, especially, at night and after lunch.

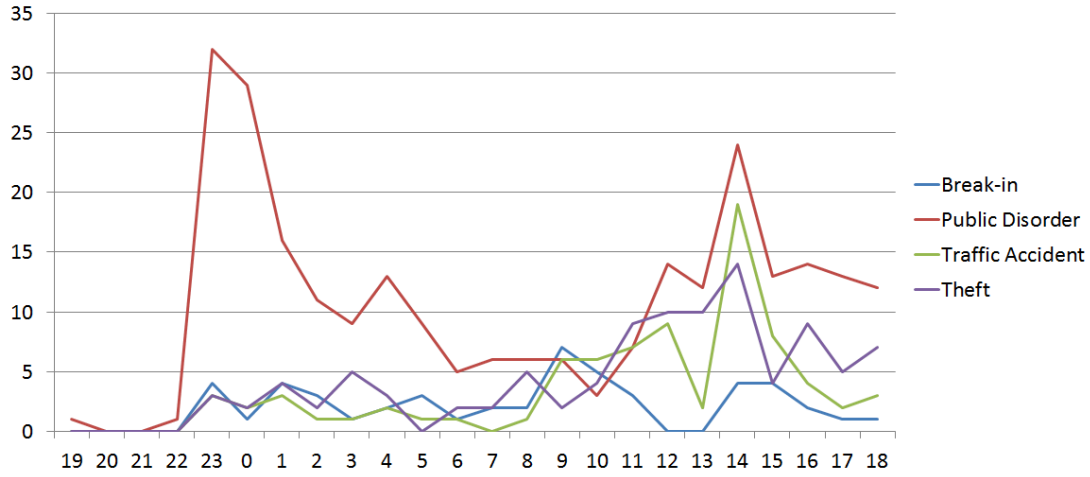


Figure 7.2: The 1st *#24hPolizei* marathon: distribution of incidents over the day.

7.2.2.2 Analysis Part 2.2 - #24hPolizei Marathon 2

The second Twitter *#24hPolizei* marathon in our dataset is the 27-28th of May 2016 from 19:00 until 19:00 of the next day. Overall, we have 1070 tweets and 83.55% of them have location names recognized by the Stanford NER tool and dictionary ER. The percentage of the incident-tweets, recognized by our dictionary of keywords, equals 52.52%. The percentage of incident-tweets with possible

injured people equals 6.04%. This statistics is presented in Table 7.1. Also, in Table 7.2, we present the statistics of the distribution of the incident types during the marathon. As for the first marathon, the leader is Public Disorder, it takes almost half (48.76%) of the incident-tweets. As for previous analysis, in Table 7.3, we present the top 5 districts mentioned in tweets. We have almost the same set of locations, except for one new location - Wedding instead of Kreuzberg.

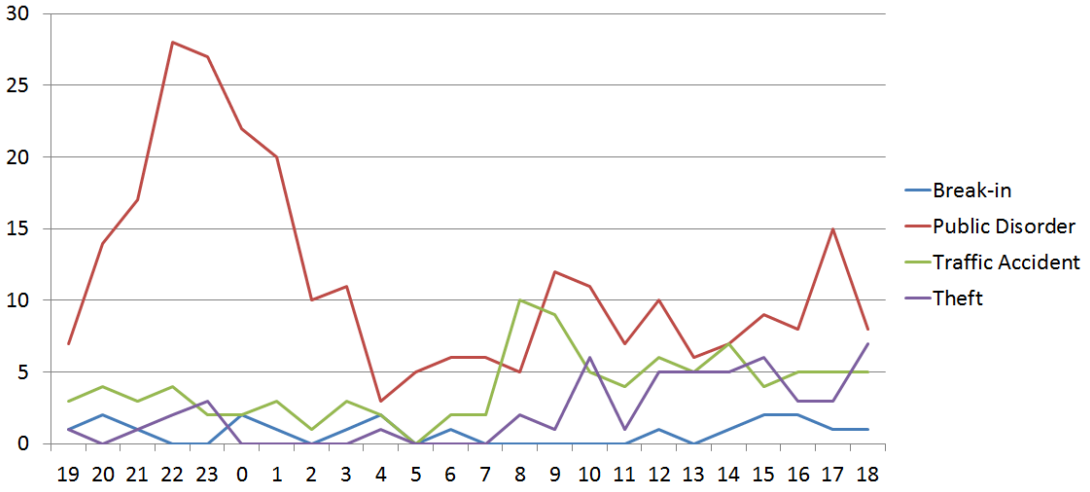


Figure 7.3: The 2nd *#24hPolizei* marathon: distribution of incidents over the day.

As for the first *#24hPolizei* marathon, for the second marathon, we also provide the chart of the intensity of incidents per hours. You can find this chart in Figure 7.3. Unlike the first marathon, we have data starting from 19:00, which is the start time of the marathon. And immediately, we can see the dramatic increase of Public Disorder. This type of the incidents again dominates over others. We have the high intensity of incidents during the entire night and it goes down only closer to the morning after 3:00. During this marathon, we have the low number of break-ins and from 7 until 11 there are not Break-in tweets at all. Then we can see that on the next day with coming the evening, Public Disorder starts to increase again.

7.3 Discussion

The Berlin Police Department provides relevant and valuable data. Analysis of these data helps to understand the situation in Berlin and provide public safety awareness for inhabitants. Some results of the analysis are obvious for locals of the city because when you live in the city several years, you can probably intuitively guess at what time and at what neighborhood you should not go to avoid, for example, thefts. As in many cities, the city center (German: Mitte) is the place where thefts and traffic accidents happen quite often. The first because the crowded places attract criminals and the second because of the big number of cars. Besides the city center (Mitte), Charlottenburg is also a very lively place, and it is presented in many statistics in this chapter. Additionally, we highlighted places Neukolln and Kreuzberg. These places also quite lively and popular among young people because of a lot of bar and clubs.

Our data analysis highlights the native guess that most crimes and public disorder incidents happen in the evening and at night. We have seen characteristic peaks at that time in Figures 7.2 and 7.3. These results are strengthened by the fact that the *#24hPolizei* marathons were performed from Friday to Saturday when people usually have a rest after working week and drink alcohol that can force to public disorders.

We categorized tweets into different types of incidents, and we have seen how they are distributed among the districts of the city. We noticed that the dominating incident type in the entire dataset is Theft, but meanwhile, in both *#24hPolizei* marathons, Public Disorder dominates. And if we look at the top 10 locations of the entire data, we can see that Public Disorder dominates in almost all locations (Figure 7.1). Also, the interesting finding is that in Alexanderplatz there are a lot of thefts in comparison to other types of incidents. And it is reasonable because this place is always full of local people and tourists. Another interesting finding is that Friedrichshain has a very small percentage of break-ins in comparison to other types of incidents.

The Berlin Police do a good job, but data, which they provide, are limited and it brings some limitation to analysis and statistics. There are no doubts that the police have more data for analysis, statistics and even for prediction

potential incidents and crimes. But from our side we have only publicly available data and, in the scope of our research project, we aim to provide situational and public safety awareness for inhabitants based on that data. This situational information is limited because of the limitation of publicly available data, but it is still valuable. The evolution of social networks and more tight integration them into the real lives of millions of people will level this limitation. Users will become a part of the entire system of providing real-time situational and public safety awareness.

7.4 Summary

Our analysis gives an initial overview of the safety situation in Berlin based on data from the Twitter account of the Berlin Police. The result of this research study is a part of our research project in the area of geospatiotemporal data analytics and one of the steps towards achieving one of the use case goals introduced in Chapter 6 - providing situational and public safety awareness. We applied normalization tasks to recognize and extract location entities and incident-related keywords. Such normalization gave us the possibility to analyze data from different perspectives and provide public safety situation awareness in Berlin. We found the top places from where tweets come. Then we used the extracted incident-related keywords to categorize tweets into one of four predefined incident types. Once we identified the locations of tweets and the incident types, we managed to build the statistics of the distribution of incident types in data. Additionally, we analyzed the public safety situation in the concrete districts of Berlin. We showed the distribution of different types of incidents among different locations. Besides the overall analysis of data, we provided the analysis of the *#24hPolizei* marathons. Since during these marathons, the police continuously publish tweets about incidents for 24 hours, we analyzed how the public safety situation changes in the city during the day. For that, we provided the chart of the distribution of incidents per hour during the day of the marathon. From that, we have found the characteristic increase of incidents in the evening time for both marathons.

As with every study, there are limitations. Our analysis of data is based on the normalization task and how we recognize and extract location entities,

incident keywords and how we categorize tweets. For recognition and extraction, we used the Stanford NER tool and dictionary-based entity recognition. We tried to normalize as much as possible tweets. In order to achieve it, we applied the preprocessing methods for cleaning the text of tweets. If we can improve the entity recognition process, we can have a more detailed and accurate analysis of data. Therefore, it must be one of the future works. For example, we would propose to use DBpedia[1] as the third additional method for annotation and recognition of entities.

For analysis, we used data from only one source - the Twitter account of the Berlin Police. For further work, we would like to gather data from other emergency service agencies. Among possible accounts, we can consider the Twitter accounts of the Berlin Fire Department (*@Berliner_Fw*), Kriminalreport (*@Kriminalreport*), transport companies of Berlin (*@BVG_Ubahn*, *@SBahnBerlin*) and so on. Besides social network accounts, we consider gathering data from the official websites of the Berlin Police, Verkehrsregelungszentrale - VKRZ (Traffic control center) and other relevant sources of data. Some papers [44] showed the possibility for detecting local geospatial events based on publicly available social georeferenced data, therefore, additionally, we are interested in integrating user-created georeferenced data from social networks to improve situational analysis.

The classification of tweets is based on the dictionary of keywords and it is a quite straightforward solution. Meanwhile, for the considered use case, it is applicable because all tweets are posted from the same account and they are written in the same style and use the same keywords to describe incidents. But, as we mentioned earlier, we plan to use additional sources of data to provide more detailed and valuable analysis. Therefore, we consider applying machine learning approaches for classification of tweets. These approaches can potentially decrease the percentage of not categorized tweets.

In our calculation of statistics, we consider the location names. But there are cases when two different tweets have different location names, which are are the names of streets. These streets can be located in the same district or these streets can cross each other. In our current implementation, these two streets are two different locations and we calculate statistics based on this assumption.

[1] http://wiki.dbpedia.org/

It would make sense to consider geogprahic coordiantes instead of steet names. Therefore, as future work, we propose to apply the method for geocoding (Section 2.1) of tweets: identification of latitude and longitude coordinates based on location names. For that, we plan to use Google Maps API[1]. This method gives the possibility to build more flexible statistics and do analysis based on the geographic coordinates of the districts or any other areas, in which we are interested. We have already used this approach in our previous work for analysis of social georeferenced data. This approach was used in the published papers [4] [5] and in several chapters of this thesis.

Within our analysis and study, we showed the possibility to analyze and provide public safety awareness based on only publicly available data without breaching any privacy of users. We plan to continue work and integrate used in the chapter approaches and statistics into our developed research platform for visualization and analysis of publicly available social georeferenced data to provide situational and public safety awareness [4] (Chapter 6).

[1] https://developers.google.com/maps/documentation/geocoding/intro

Chapter 8

Conclusion and Outlook

The benefits of utilizing georeferenced and geospatiotemporal data manifold. The most important that they have objective information, such as location and time. It brings many opportunities in data analysis and visualization that increase the value of data knowledge discovery. In particular, with georeferenced data, we can visualize the distribution of data on the online map, find the correlation between events, cluster data and detect anomalies. When we consider also the temporal aspect of data, we can objectively tackle an additional analytic questions regarding the temporal distribution of data - detection of the intensest area in temporal space.

Not all data that we work are georeferenced or geospatiotemporal, but they can be converted to such data by different methods of entity recognition and data normalization. Such transformation gives as new opportunities in the analysis that we do not have with raw non-normalized data. Particularly, we can combine subjective features from original raw data and objective features from normalized data, and as a result, we can build combined analytical methods that support could the data analysis and knowledge discovery.

Georeferenced and geospatiotemporal data brings much valuable information but they bring also challenges around working with such data and developing new methods and frameworks for geospatiotemporal data analytics. We have addressed such challenges in this thesis. We presented several contributions aimed to utilize and explore knowledge of geospatiotemporal data jointly with contributions in analysis and normalization of such data. Overall, based on our proposed

methods and frameworks we developed applications and use cases that utilized our methods on real data. It helped us to explore valuable data patterns that we discussed in corresponding chapters.

In Chapter 2, we have introduced means for data normalization. We have considered several steps of normalization that contain applying a method for entity recognition, using external tools and services, and their combination. The part of data normalization is methods for data geolocation that extract valuable for our research information about the location. Location information makes our data as georeferenced data. This chapter showed the initial steps and challenges in processing, working and storing georeferenced data and our solutions for that. At the end of the chapter we discussed possible future research challenges around geospatial data processing.

We used the knowledge and outcome derived from Chapter 2 for developing our method for clustering georeferenced data in Chapter 3. In that chapter, we illustrated different approaches for clustering georeferenced data. We provided detailed explanations of clustering approaches and identified advantages and disadvantages of them. Based on our evaluation and motivation, we introduced the requirements that utilize the combination of most advantages of existing approaches, in order to propose our solution for clustering georeferenced data. We proposed the clustering method that meets introduced in the chapter requirements: (1) online algorithm, (2) distance-based clustering, (3) density-based cluster center, (4) applicability for map-reduce clustering. We designed and explained methods and algorithms, and afterwards, we demonstrated the results of work based on collected georeferenced data. Based on discussion around the obtained results, we proposed the mechanism for configuration of clustering, in order to have flexible possibility to trade off between performance and accuracy depending on needs. After implementation and successful usage, several extensions came to mind. We described them in the summary section, and one of them, we addressed in the next chapter.

In Chapter 4, we looked at the proposed earlier method for clustering in terms of extension it for one more additional aspect. We proposed and extended the algorithm by bringing the temporal aspect into the clustering. We kept the earlier defined requirement but with inclusion additional one - keeping the cluster's

center in the intensest area of temporal space. We designed and illustrated the method workflow jointly with the pseudo code. We defined also a possible application for such clustering - detection of anomalies in spatial and temporal spaces. Such anomalies could be results of analysis of real data from social networks or traffic sensors. In the case of analysis of georeferenced data from social networks, it could help to detect local geospatial events and support the research proposed and presented by Walther et al. in 2013 [44]. We defined the possible extension of the algorithm that would include the clustering with concerning the content (textural or type) of data events. Such textual and type information of data would help to classify clustered anomalies.

In Chapter 5, we tackled the challenge of designing the framework that can convert non-spatial data into spatiotemporal data and apply pattern mining algorithms to them. The proposed framework could utilize the power of existing data mining algorithms in the area of spatiotemporal data mining. The framework contains several software architectural modules and we provided the full description of them jointly with the pseudo code for the most crucial parts. To evaluate and to test an use case, we applied the developed framework for publicly available crime data. It gave us interesting findings in the scope of correlation of different types of crimes in spatial and temporal aspects. From our point of view, the framework has the potentiality for extension and application. We described our thoughts about that at the end of the chapter.

In the next chapters, we concentrated on applications and use cases, that could utilize earlier proposed methods and frameworks, in order to provide data analysis jointly with data visualization. We designed and implemented the application that collects publicly available georeferenced data from Twitter. Using methods for data normalization, clustering, and visualization, we were aimed to provide situational awareness. We demonstrated the concrete use case based on the collected data. We showed, how with our application we can find the reflection of events on an online map and identify happenings. The application was developed around the goal - providing situational awareness. In this goal scope, we identified an additional challenge - filtering data from the perspective of situational awareness. We analyzed collected data, and based on the results, we proposed criteria for filtering irrelevant data. Excluding irrelevant data could

improve the analysis and visualization of data, and as the result, it improves providing situation awareness. Every part of the application has the potentiality for improvement and extension. We discussed such possible improvements at the end of the chapter.

The next interesting use case, which we addressed, is providing public safety awareness. For this challenge, we collected relevant data from the Twitter account of Berlin Police. We analyzed and visualized collected and normalized data. The results of the analysis explored knowledge about public safety situation in the city. Such information is available also from publicly available reports of Berlin Police, but the value of our results is that this information was explored through data normalization and based on only publicly available data from social networks. The value of this information could be increased with increasing the number of sources that provide publicly available information. And it was one of the proposals for future work.

In summary, in this thesis, we addressed multiple challenges in georeferenced and geospatiotemporal data through data normalization, design and proposal methods for clustering data in spatial and temporal spaces, frameworks for spatiotemporal sequential rule mining, implementation of applications and tools, analysis of data and exploring use cases with discussions of findings.

References

[1] Amirkhanyan, A., Cheng, F., Meinel, C.: Real-time clustering of massive geodata for online maps to improve visual analysis. In: Innovations in Information Technology (IIT), 2015 11th International Conference on. pp. 308–313 (Nov 2015) 6, 56

[2] Amirkhanyan, A., Meinel, C.: Density and intensity-based spatiotemporal clustering with fixed distance and time radius. In: Kar, A.K., Ilavarasan, P.V., Gupta, M., Dwivedi, Y.K., Mäntymäki, M., Janssen, M., Simintiras, A., Al-Sharhan, S. (eds.) Digital Nations – Smart Cities, Innovation, and Sustainability. pp. 313–324. Springer International Publishing, Cham (2017) 6

[3] Amirkhanyan, A., Meinel, C.: The framework for spatiotemporal sequential rule mining: Crime data case study. In: 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA). pp. 34–38 (Oct 2017) 6

[4] Amirkhanyan, A., Meinel, C.: Visualization and analysis of public social geodata to provide situational awareness. In: 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI). pp. 68–73 (Feb 2016) 7, 90, 118

[5] Amirkhanyan, A., Meinel, C.: Analysis of the value of public geotagged data from twitter from the perspective of providing situational awareness. In: Dwivedi, Y.K., Mäntymäki, M., Ravishankar, M., Janssen, M., Clement, M., Slade, E.L., Rana, N.P., Al-Sharhan, S., Simintiras, A.C. (eds.) Social

Media: The Good, the Bad, and the Ugly. pp. 545–556. Springer International Publishing, Cham (2016) 7, 93, 118

[6] Amirkhanyan, A., Meinel, C.: Analysis of data from the twitter account of the berlin police for public safety awareness. In: 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD). pp. 209–214 (April 2017) 7

[7] Amirkhanyan, A.: Towards analysis of public social data to improve situational awareness. In: Proceedings of the 9th Ph.D. Retreat of the HPI Research School on service-oriented systems engineering (Universitätsverlag Potsdam 2016) (2016) 7

[8] Amirkhanyan, A.: One working day of the berlin police analysis of data from the #24hpolizei twitter marathon. In: Proceedings of the 10th Ph.D. Retreat of the HPI Research School on service-oriented systems engineering (Universitätsverlag Potsdam 2017) (2017) 7

[9] Amirkhanyan, A.: Testbed automation for network security and security analytics. In: Proceedings of the 8th Ph.D. Retreat of the HPI Research School on service-oriented systems engineering (Universitätsverlag Potsdam 2015) (2015) 7

[10] Amirkhanyan, A., Sapegin, A., Gawron, M., Cheng, F., Meinel, C.: Simulation user behavior on a security testbed using user behavior states graph. In: Proceedings of the 8th International Conference on Security of Information and Networks. pp. 217–223. SIN '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2799979.2799985 7

[11] Sapegin, A., Amirkhanyan, A., Gawron, M., Cheng, F., Meinel, C.: Poisson-based anomaly detection for identifying malicious user behaviour. In: Selected Papers of the First International Conference on Mobile, Secure, and Programmable Networking - Volume 9395. pp. 134–150. MSPN 2015, Springer-Verlag, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-319-25744-0_12 8

[12] Geocoding definition: Wikipedia. https://en.wikipedia.org/wiki/Geocoding, last visited on 28.10.2018 9

[13] ArcGIS Geocoding Process. http://desktop.arcgis.com/en/arcmap/10.3/guide-books/geocoding/the-geocoding-process.htm, last visited on 28.10.2018 10

[14] Google Geocoding API. https://developers.google.com/maps/documentation/geocoding/intro, last visited on 28.10.2018 11, 14, 68, 69

[15] Geolocation. https://en.wikipedia.org/wiki/Geolocation, last visited on 28.10.2018 13

[16] Stanford named entity recognizer (ner). http://nlp.stanford.edu/software/CRF-NER.shtml, last visited on 28.10.2018 14, 54, 97, 107

[17] Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 363–370. ACL '05, Association for Computational Linguistics, Stroudsburg, PA, USA (2005), https://doi.org/10.3115/1219840.1219885 14, 54

[18] Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web. pp. 591–600. WWW '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1772690.1772751 15, 82

[19] Twitter API Documentation. https://dev.twitter.com/overview/api, last visited on 28.10.2018 15, 17, 18, 84, 94

[20] Bright Planet. http://www.brightplanet.com/2013/06/twitter-firehose-vs-twitter-api-whats-the-difference-and-why/-should-you-care), last visited on 28.10.2018 15, 83

[21] Morstatter, F., Pfeffer, J., Liu, H., Carley, K.M.: Is the sample good enough? comparing data from twitter's streaming API with twitter's firehose. CoRR abs/1306.5204 (2013), http://arxiv.org/abs/1306.5204 15

[22] The world's most active Twitter cities. http://www.forbes.com/sites/victorlipman/2012/12/30/the-worlds-most-active-twitter-city-you-wont-guess-it/, last visited on 28.10.2018 15, 83, 93

[23] Geographic midpoint. http://www.geomidpoint.com/calculation.html, last visited on 28.10.2018 20, 39, 60

[24] Geohash. https://en.wikipedia.org/wiki/Geohash, last visited on 28.10.2018 21, 28

[25] Database definition: Wikipedia. https://en.wikipedia.org/wiki/Database, last visited on 28.10.2018 22

[26] Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Yormark [77], pp. 47–57, http://doi.acm.org/10.1145/602259.602266 23

[27] MongoDB Documentation. https://docs.mongodb.com, last visited on 28.10.2018 23

[28] RFC7946. https://tools.ietf.org/html/rfc7946, last visited on 28.10.2018 23

[29] MongoDB GeoSpatial Queries. https://docs.mongodb.com/manual/geospatial-queries, last visited on 28.10.2018 23

[30] WGS84. http://www.unoosa.org/pdf/icg/2012/template/WGS_84.pdf, last visited on 28.10.2018 23

[31] Google's recommendations for clustering geodata. https://developers.google.com/maps/articles/toomanymarkers, last visited on 28.10.2018 28, 30

[32] M.Parimala, Lopez, D., Senthilkumar, N.: A survey on density based clustering algorithms for mining large spatial databases (2011), http://www.sersc.org/journals/IJAST/vol31/5.pdf 31

[33] Ester, M., peter Kriegel, H., S, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. pp. 226–231. AAAI Press (1996) 31

[34] Ankerst, M., Breunig, M.M., peter Kriegel, H., Sander, J.: Optics: Ordering points to identify the clustering structure. pp. 49–60. ACM Press (1999) 31

[35] Tamura, K., Ichimura, T.: Density-based spatiotemporal clustering algorithm for extracting bursty areas from georeferenced documents. In: Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on. pp. 2079–2084 (Oct 2013) 31

[36] Sakai, A., Tamura, K., Kitakami, H.: A new density-based spatial clustering algorithm for extracting attractive local regions in georeferenced documents. In: Proceedings of the International MultiConference of Engineers and Computer Scientists. pp. 360–365. IMECS '14, Newswood Limited, Hong Kong (2014), http://www.iaeng.org/publication/IMECS2014/IMECS2014_pp360-365.pdf 31

[37] Kisilevich, S., Mansmann, F., Keim, D.: P-dbscan: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In: Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application. pp. 38:1–38:4. COM.Geo '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1823854.1823897 31

[38] Varghese, B., Unnikrishnan, A., Jacob, K.: Spatial clustering algorithms- an overview 3 (01 2014) 31

[39] Neethu C V, S.S.: Review of spatial clustering methods (Jun 2013) 31

[40] Liu, Q., Deng, M., Shi, Y., Wang, J.: A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity.

Comput. Geosci. 46, 296–309 (Sep 2012), http://dx.doi.org/10.1016/j.cageo.2011.12.017 31

[41] Distance between geo coordinates. http://www.movable-type.co.uk/scripts/latlong.html, last visited on 28.10.2018 39, 60

[42] Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: Real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web. pp. 851–860. WWW '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1772690.1772777 51, 80

[43] De Longueville, B., Smith, R.S., Luraschi, G.: Omg, from here, i can see the flames!": A use case of mining location based social networks to acquire spatio-temporal data on forest fires. In: Proceedings of the 2009 International Workshop on Location Based Social Networks. pp. 73–80. LBSN '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1629890.1629907 51, 80

[44] Walther, M., Kaisser, M.: Geo-spatial event detection in the twitter stream. In: Proceedings of the 35th European Conference on Advances in Information Retrieval. pp. 356–367. ECIR'13, Springer-Verlag, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-36973-5_30 52, 80, 87, 117, 121

[45] Birant, D., Kut, A.: St-dbscan: An algorithm for clustering spatial-temporal data. Data Knowl. Eng. 60(1), 208–221 (Jan 2007), http://dx.doi.org/10.1016/j.datak.2006.01.013 52, 53

[46] Kisilevich, S., Mansmann, F., Nanni, M., Rinzivillo, S.: Spatio-temporal clustering, pp. 855–874. Springer US, Boston, MA (2010), http://dx.doi.org/10.1007/978-0-387-09823-4_44 52

[47] Chen, H., Chung, W., Xu, J.J., Wang, G., Qin, Y., Chau, M.: Crime data mining: a general framework and some examples. Computer 37(4), 50–56 (April 2004) 65

[48] Nath, S.V.: Crime pattern detection using data mining. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 41–44. WI-IATW '06, IEEE Computer Society, Washington, DC, USA (2006), http://dx.doi.org/10.1109/WI-IATW.2006.55 65

[49] Mohler, G.O., Short, M.B., Brantingham, P.J., Schoenberg, F.P., Tita, G.E.: Self-exciting point process modeling of crime. Schoenberg F P and Tita G E (2011) 66

[50] Wang, T., Rudin, C., Wagner, D., Sevieri, R.: Learning to Detect Patterns of Crime, pp. 515–530. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40994-3_33 66

[51] Wang, T., Rudin, C., Wagner, D., Sevieri, R.: Detecting patterns of crime with series finder. In: Proceedings of the 17th AAAI Conference on Late-Breaking Developments in the Field of Artificial Intelligence. pp. 140–142. AAAIWS'13-17, AAAI Press (2013), http://dl.acm.org/citation.cfm?id=2908286.2908333 66

[52] Fournier-Viger, P.: An introduction to sequential rule mining http://data-mining.philippe-fournier-viger.com/introduction-to-sequential-rule-mining/, last visited on 28.10.2018 67

[53] Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining (2017), http://www.philippe-fournier-viger.com/dspr-paper5.pdf, last visited on 28.10.2018 67

[54] Crime Data: Cambridge PD 2009-2016. https://data.world/data-society/cambridge-crime-data-2009-2016, last visited on 28.10.2018 74, 76

[55] Crime Data: Chicago PD 2001-2017. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2/data, last visited on 28.10.2018 74, 76

[56] Fournier-Viger, P., Gueniche, T., Zida, S., Tseng, V.S.: Erminer: Sequential rule mining using equivalence classes. In: Blockeel, H., van Leeuwen, M., Vinciotti, V. (eds.) Advances in Intelligent Data Analysis XIII. pp. 108–119. Springer International Publishing, Cham (2014) 77

[57] Top ways researchers can use social data. http://discovertext.com/2015/09/06/top-ways-researchers-can-use-social-data, last visited on 28.10.2018 79

[58] Zheng, Y., Chen, Y., Xie, X., Ma, W.Y.: Geolife2.0: A location-based social networking service. In: Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on. pp. 357–358 (May 2009) 80

[59] Miller, H.J., Han, J.: Geographic Data Mining and Knowledge Discovery. Taylor & Francis, Inc., Bristol, PA, USA (2001) 80

[60] Preoţiuc-Pietro, D., Cohn, T.: Mining user behaviours: A study of check-in patterns in location based social networks. In: Proceedings of the 5th Annual ACM Web Science Conference. pp. 306–315. WebSci '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2464464.2464479 80

[61] Ye, M., Shou, D., Lee, W.C., Yin, P., Janowicz, K.: On the semantic annotation of places in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 520–528. KDD '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2020408.2020491 80

[62] Scellato, S., Noulas, A., Mascolo, C.: Exploiting place features in link prediction on location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1046–1054. KDD '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2020408.2020575 80

[63] Leetaru, K., Wang, S., Cao, G., Padmanabhan, A., Shook, E.: Mapping the global twitter heartbeat: The geography of twitter. First Mon-

day 18(5) (2013), http://firstmonday.org/ojs/index.php/fm/article/view/4366 80

[64] Adnan, M., Longley, P.A., Khan, S.M.: Social dynamics of twitter usage in london, paris, and new york city. First Monday 19(5) (2014), http://firstmonday.org/ojs/index.php/fm/article/view/4820 80

[65] Guo, D., Chen, C.: Detecting non-personal and spam users on geo-tagged twitter network. T. GIS 18(3), 370–384 (2014), http://dx.doi.org/10.1111/tgis.12101 81, 98

[66] Pavalanathan, U., Eisenstein, J.: Confounds and consequences in geotagged twitter data. CoRR abs/1506.02275 (2015), http://arxiv.org/abs/1506.02275 81

[67] Twitter's statistics. http://www.internetlivestats.com/, last visited on 28.10.2018 82

[68] Heverin, T., Zach, L.: Twitter for city police department information sharing. In: Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem - Volume 47. pp. 41:1–41:7. ASIS&T '10, American Society for Information Science, Silver Springs, MD, USA (2010), http://dl.acm.org/citation.cfm?id=1920331.1920390 106

[69] Bruns, A., E Burgess, J., Crawford, K., Shaw, F.: #qldfloods and @qpsmedia: Crisis communication on twitter in the 2011 south east queensland floods (01 2012) 106

[70] Ehnis, C., Bunker, D.: Social media in disaster response: Queensland police service-public engagement during the 2011 floods. In: ACIS 2012: Location, location, location: Proceedings of the 23rd Australasian Conference on Information Systems 2012. pp. 1–10. ACIS (2012) 106

[71] Ehnis, C., Bunker, D.: The impact of disaster typology on social media use by emergency services agencies: the case of the boston marathon bombing. In: 24th Australasian Conference on Information Systems (ACIS). pp. 1–12. RMIT University (2013) 106

[72] Mirbabaie, M., Ehnis, C., Stieglitz, S., Bunker, D.: Communication Roles in Public Events, pp. 207–218. Springer Berlin Heidelberg, Berlin, Heidelberg (2014), http://dx.doi.org/10.1007/978-3-662-45708-5_13 106

[73] Ehnis, C., Mirbabaie, M., Bunker, D., Stieglitz, S.: The role of social media network participants in extreme events. In: Proceedings of the 25^{th} Australasian Conference on Information Systems. Auckland, New Zealand (2014), publication status: Published 106

[74] Reverse geocoding definition: Wikipedia. https://en.wikipedia.org/wiki/Reverse_geocoding, last visited on 28.10.2018

[75] Geospatial Query Operators. https://docs.mongodb.com/manual/reference/operator/query-geospatial, last visited on 28.10.2018

[76] The number of monthly active Twitter users worldwide. http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/, last visited on 28.10.2018

[77] Yormark, B. (ed.): SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984. ACM Press (1984) 126

[78] Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: Spmf: A java open-source pattern mining library. J. Mach. Learn. Res. 15(1), 3389–3393 (Jan 2014), http://dl.acm.org/citation.cfm?id=2627435.2750353

[79] Geohash Clustering. http://blog.trifork.com/2013/08/01/server-side-clustering-of-geo-points-on-a-map-using/-elasticsearch, last visited on 28.10.2018

[80] Geocluster: Server-side clustering for mapping in Drupal based on Geohash. http://dasjo.at/files/geocluster-thesis-dabernig.pdf, last visited on 28.10.2018

[81] Sakai, T., Tamura, K., Kitakami, H.: Density-based adaptive spatial clustering algorithm for identifying local high-density areas in georeferenced documents. In: Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on. pp. 513–518 (Oct 2014)

[82] Ahern, S., Naaman, M., Nair, R., Yang, J.H.I.: World explorer: Visualizing aggregate data from unstructured text in geo-referenced collections. In: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries. pp. 1–10. JCDL '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1255175.1255177

[83] Duan, L., Xiong, D., Lee, J., Guo, F.: A local density based spatial clustering algorithm with noise. In: Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on. vol. 5, pp. 4061–4066 (Oct 2006)

[84] Ram, A., Jalal, S., Jalal, A.S., Kumar, M.: A density based algorithm for discovering density varied clusters in large spatial databases (2010)

[85] Singh, S.: Spatial temporal analysis of social media data http://129.187.45.33/CartoMasterNew/fileadmin/user_upload/Smita_Presentation.pdf