University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Location-based Open Source Intelligence to Infer Information in LoRa Networks

Jonathan Contreras
Zurich, Switzerland
Student ID: 18-943-993

Supervisor: Dr. Alberto Huertas Celdran, Jan von der Assen
Date of Submission: July 5, 2023

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Diese Bachelorarbeit stellt eine neue Open-Source intelligence (OSINT) Plattform vor. Das Ziel der Platform ist die Identifizierung einer primären Person und eines damit einhergehendes Event. Als Datenquelle wird hierbei ein LoRa (Long Range) Datensatz in Kombination mit öffentlich zugänglichen Daten genutzt. Daraufhin benutzt die Plattform Web Scraping, das Sprachmodell GPT-3.5 von OpenAI und einen speziell entwickelten Matching Score Algorithmus. Ziel der Arbeit ist die Erstellung eines umfangreichen Bildes der primären Person und Schlussfolgerungen darüber zu ziehen, wer noch an Zeit und Ort der erfassten LoRa Daten präsent war. Die Evaluierung dieses Ansatzes zeigt, dass 14 von 16 Teilnehmern mit dem verwendeten Datensatz wiedergefunden werden konnten. Das demonstriert die Fähigkeit der Tools, relevante Datensätze zu erstellen, mit potenziellen Teilnehmern des Events. Wenn die Genauigkeit der Einstufung (ob eine Person präsent war oder nicht) anschauen, konnte eine Präzision von 0.75 erreicht werden, jedoch zeigt der Recall Wert von 0.46 auf, dass einige präsente Teilnehmer als nicht anwesend eingestuft wurden. Die Ergebnisse reflektieren die Schwierigkeit der Identifizierung von Teilnehmern eines Events, das kaum Online Präsenz aufweist. Trotz des herausfordernden Szenarios zeigt die Plattform eine innovative Variante zur Einbindung von OSINT Praktiken in LoRa Datensätze. Zukünftige Arbeiten werden sich auf die Verbesserung der Robustheit des Tools konzentrieren, weitere soziale Medienplattformen einbinden und die Anpassungsfähigkeit für verschiedene Szenarien verbessern. Zusätzlich sollte die Integration weiterer Sprachmodelle in Betracht gezogen werden.

This thesis introduces and evaluates a novel platform that uses Open-source intelligence (OSINT) to identify a primary subject and an associated event using publicly accessible data. As a starting point, the platform utilizes LoRa (Long Range) datasets. This novel tool will make use of web scraping techniques, the power of OpenAI's large language model GPT-3.5, and a custom matching score algorithm. The objective is to collect a comprehensive image of the primary subject and infer potential participants of the specific location and time covered by the LoRa dataset. Evaluating our approach demonstrates its effectiveness in identifying 14 out of 16 actual participants, showcasing its ability to create a relevant dataset of potential participants. Looking at the accuracy, the model manages to achieve a precision score of 0.75, while the recall score of 0.46 indicates some true positives were not captured. The results reflect the difficulty in identifying participants in a private event with a limited public presence. Despite the challenging scenario, this tool represents an innovative approach to merging OSINT techniques with LoRa data. Future work will focus on enhancing the tool's robustness, expanding its coverage to additional social media platforms, improving adaptability across diverse scenarios, and exploring advanced language models.

Acknowledgments

I want to thank my supervisors, Dr. Alberto Huertas Celdran and Jan von der Assen. Their feedback and support helped me to complete this thesis.

Contents

Chapter 1

Introduction

OSINT (Open-source intelligence) has increasingly evolved into a potent tool, revealing substantial potential across a wide array of application domains [1]. Yet, current research barely scratches the surface of its capabilities, focusing primarily on data procured from Social Network Sites (SNS) and thus narrowly defining the scope of OSINT's applicability.

This concentrated focus on SNS leaves a broad field largely untouched, particularly in the domains of sensory data and low-power wide-area network (LPWAN) communication technologies. Certain sectors within the domain of OSINT remain notably under-researched, suggesting an abundance of untapped opportunities for further exploration and investigation.

Among these uncharted territories is LoRaWAN (Long Range Wide Area Network), a widely used communication protocol that is currently overlooked in the context of OSINT research. This discrepancy suggests a potential blind spot in our understanding and application of OSINT. The fusion of OSINT with LoRa network data could potentially amplify data enrichment, enhance the accuracy of existing methodologies, and pave the way for a more comprehensive interpretation of different scenarios.

This thesis aims to broaden the horizons of OSINT research, to explore the overlooked integration of OSINT with LPWAN communication technologies, specifically LoRaWAN. Exploring the under-researched intersection of LPWAN technologies with OSINT promises to reveal new viewpoints and methodologies. This could potentially enhance our understanding of OSINT's applications, leading to a more in-depth and multifaceted knowledge base.

1.1 Motivation

Open-source intelligence holds substantial promise for numerous application domains [1]. However, it appears that existing research has only begun to explore its full potential. A review of the literature reveals that most studies are primarily focused on data accessible via Social Network Sites. A noticeable void exists due to the insufficient exploration of

sensory data and low-power wide-area network communication technologies in the domain of Open-source intelligence research.

In recent years, the rapid rise of Internet of Things (IoT) devices has transformed various facets of daily life and industrial operations from healthcare to agriculture [2]–[4]. This expansion is driving increased use of LoRaWAN, one of the most prevalent communication protocols in the IoT space, due to its efficiency and wide-area network capabilities[5]. Despite its ubiquity in the growing IoT landscape, LoRaWAN remains surprisingly underexplored in the context of OSINT research. This is a significant observation considering the extensive and diverse data generated by these IoT devices, offering a largely untapped resource in the field of OSINT.

This thesis expands OSINT research by bridging the gap between extensively researched areas and underexplored domains such as LPWAN communication technologies. In doing so, it allows for a more diversified and multi-dimensional approach to OSINT, potentially yielding richer and more accurate insights. The focus of this thesis is to address this research gap, specifically exploring the potential integration of OSINT with LoRaWAN.

1.2 Thesis Outline

Chapter 2 serves as an introduction to the background knowledge of Open-Source Intelligence, the communication technology LoRa, and the specific LoRa protocol, LoRaWAN.

Chapter 3 investigates the relevant literature. Sub-chapter 3.1 outlines the search strategy used and the exclusion criteria of the literature review process. Sub-chapter 3.2 provides a summary of the literature on Geo-location-based OSINT. Sub-chapter 3.3 summarizes research regarding people-based OSINT, and sub-chapter 3.4 addresses the area of IoT-based OSINT. This chapter concludes with a brief summary and discussion of the related work, highlighting key trends and gaps in the existing literature.

Chapter 4 shifts the focus to the design of the platform. Beginning with an examination of the existing code base and its functionalities in sub-chapter 4.1.1. Building upon this, sub-chapter 4.1.2 presents the proposed design, discussing the contributions and improvements. Additionally, sub-chapter 4.1.3 moves to the architectural aspects of the platform, providing a detailed overview of its structure. Furthermore, the chapter includes a brief description of the scenario employed for measuring the performance of the model, discussed in sub-chapter 4.2.

Chapter 5 provides a detailed description of the implementation specifics, dividing it into three unique components: the LinkedIn scraping process, discussed in sub-chapter 5.2, the method of scraping employee company pages outlined in sub-chapter 5.3; and the approach used in the scraping process to infer additional event participants, explained in sub-chapter 5.4.

In Chapter 6, a detailed case study provides a demonstration of the platform's use. It offers a step-by-step guide showcasing its features with a real-world scenario from a previous Hackathon.

Chapter 7 presents an evaluation of the findings of this thesis. The performance metrics are demonstrated in sub-chapter 7.1, alongside the corresponding results in Section 7.2. Sub-chapter 7.4 discusses and interprets these findings. Additionally, sub-chapter 7.3 focuses specifically on the identified limitations. Sub-chapter 7.5 evaluates the time efficiency of the scrapers, while chapter 7.6 explores the implications derived from the findings of this thesis.

Chapter 8 concludes the thesis by providing a summary of the key findings and conclusions resulting from the thesis. It provides a brief review of the study's contributions and hints at potential directions for future research.

Subsequently, Chapter 9, as the final section of this thesis, builds on the summary chapter and presents a set of recommendations for future investigations.

Chapter 2

Background

2.1 Open Source Intelligence

Open-Source Intelligence (OSINT) is the use of free and publicly accessible sources such as social media, news articles, and reports, to deduce or gain information relevant to a specific topic[6]. The role of social media and individual interactions has become increasingly important within the OSINT framework. As more people engage on digital platforms, they contribute a wealth of user-generated data. This data can encompass everything from text-based posts and comments to shared links and even patterns of interaction. Such information, when subjected to rigorous analytical methodologies, has the potential to yield significant insights in various domains.

OSINT is used by a variety of organizations, including governments, law enforcement, businesses, and researchers [1]. Governments initially used intelligence gathering for military and national security purposes, tracing its roots back to the earliest days of this practice. However, with the advent of the internet and the growth of digital data, OSINT's use has broadened, and its applications now encompass fields as diverse as market research, competitive intelligence, fraud detection, and disaster response [7].

A key characteristic of OSINT is its ability to gather information from a wide range of publicly available sources, allowing for cross-validation through various methods. This can involve the use of geolocation data to track a person or object's location, or analyzing timestamps to determine the occurrence time of an event or a particular online post. The incorporation of social media data and people-centric analysis in OSINT further enhances its breadth and precision, leading to more effective and timely intelligence.

Geolocation data is especially useful in OSINT as it can offer significant insights into an individual or group's movements and activities [8]. Likewise, timestamps serve an integral role by providing a temporal context to the data posted online or to the occurrence of specific events. This becomes particularly essential in contexts that demand high accuracy and reliability of information, such as in legal proceedings or emergency response scenarios.

In conclusion, OSINT's significance has grown across a wide range of fields. Its ability to gather and analyze publicly available data from multiple sources makes it an invaluable

tool for decision-making and problem-solving, with the role of people and social media interactions being a noteworthy contributor to the overall richness of OSINT.

2.2 LoRa

LoRa (Long Range) is a wireless communication technology designed for low-power, wide-area networks (LPWAN) that enable long-range communication between devices with minimal battery consumption [9].

LoRa works by transmitting data over radio waves at a frequency range of 868 MHz to 915 MHz (depending on the region), using spread-spectrum modulation techniques to increase the range and reduce interference. LoRa uses a form of chirp spread spectrum (CSS) modulation, which enables it to transmit data at very low bit rates while maintaining a high level of sensitivity [9]. In this context, sensitivity refers to the ability of a receiver to detect and process weak signals.

LoRa also uses a unique network architecture, with a gateway that receives and transmits data between end devices and a cloud-based network server. The end devices can communicate with multiple gateways, which allows for redundancy and increased range. For further processing and analysis, this network server will then send the data to an application server.

Concluding, LoRa offers a cost-effective, energy-efficient, and wide-reaching communication technology. These features make LoRa well-suited for Internet of Things (IoT) applications such as smart cities, industrial automation, and environmental monitoring.

2.2.1 LoRaWAN

LoRaWAN (Long Range Wide Area Network) is a wireless communication protocol that is based on the LoRa technology. It is designed to enable long-range, low-power communication between IoT devices and gateways, and provides a standardized network architecture for deploying and managing IoT devices at scale [9].

LoRaWAN uses a star-of-stars network topology, where end devices can communicate with multiple gateways, which in turn are connected to a central network server. The network server manages the communication between the end devices and the application server where the data is processed and analyzed. This ability of end devices to communicate with multiple gateways increases network reliability and coverage.

LoRaWAN comes equipped with enhanced safety measures such as comprehensive encryption and authentication, guaranteeing that information sent over the network remains confidential and protected. Furthermore, the protocol accommodates various device classes, each with distinct power usage, latency, and data transmission needs. This flexibility makes it suitable for numerous IoT applications.

LoRaWAN is not the only LPWAN technology. There are several LPWAN technologies available on the market, including LoRaWAN, Sigfox, Ingenu, and Weightless. These technologies differ in terms of the radio spectrum used, network architecture, and data rate, among other factors [5].

In conclusion, LoRaWAN represents a key cornerstone of the IoT infrastructure. Features such as the ability to manage vast networks of devices, combined with robust security features such as end-to-end encryption made LoRaWAN one of the most popular choices for IoT applications. Despite operating among a myriad of LPWAN technologies like Sigfox, Ingenu, and Weightless, each with unique attributes, LoRaWAN manages to balance efficient power use, adequate data rates, and expansive network architecture, proving its significant role in the thriving IoT landscape.

Chapter 3

Related Work

Over the last few years, Open-Source Intelligence (OSINT) has become a popular area of research. This can be attributed to the increasing significance of open-source information in different domains. Researchers are exploring various aspects of OSINT. Areas of interest are the sources, methods, implementation, and automation tools. Also, the incorporation of artificial intelligence has become more prominent in recent years [7], [10].

This section introduces the reader to related work about the main areas of OSINT important to this thesis. The first section will provide an overview of the applied search strategy to find relevant papers. The second section will focus on the importance and benefits of location data. Subsequently, research about people-focused OSINT is represented. Here, OSINT is used to gather information about a specific person or a group of interest. Section number three concentrates on the usage of OSINT in relation to Internet of Things (IoT) devices. The chapter concludes with a summary and discussion of the findings from the literature review.

3.1 Literature Search Strategy

To learn about the current state of research, the databases from Academia, IEEE, Google Scholar, and ScienceDirect were searched for the following strings in keywords and titles: "(OSINT or Open Source Intelligence) and (Location or Geo-location or People or Social Networks or IoT)". The search was conducted in February 2023. A total of 49 papers were collected and read and classified into geo-location-based, people-based, or IoT-based papers. Papers collected from the literature search that focused on research unrelated to this paper, such as ethics, were excluded.

In a second step, a reverse search was performed on all papers that made the selection. Referenced papers with similar research were added to the collection. In total, 19 location-based, 10 people-based papers, and 9 IoT-based Papers made the selection and were read completely. From there on the following papers were excluded: (1) papers not written in English; (2) theoretical papers that did not work with a dataset; and (3) papers that mainly analyze the ethical aspect of OSINT. This left a collection of 7 location-based,

5 people-based, and 5 IoT-based Papers which are summarized and compared in the following sections.

3.2 Geolocation-Based OSINT

The availability of location data has a high value in OSINT. Analyzing whereabouts can be used to create user patterns, profile people, and even predict their future locations [11].

The paper [12] examines the possibility of localizing a social network user based on relevant OSINT and developing a toolkit to collect and analyze data from three social networks, namely Twitter, Facebook, and Instagram. Firstly, they gather social network data and extract basic features such as the user's location (if provided), relationships between users, and IP direction. The Data was collected via APIs and crawling the timelines of a user. The proposed method starts localizing users from Twitter and expands its search through Facebook and Instagram. Analyzing relationships between users is fundamental to correlating their whereabouts and estimating the location of the target. During the analysis, geotags embedded in the tweets/posts were used along with natural language tools such as CLAVIN and Geograpy to find the location from where the tweet was posted. The method tested various machine learning algorithms and found that using logistic regression, accurate user location predictions could be made at a success ratio of 77.72%.

In contrast, the relationship-based approach [13] discusses how to infer users' mobility patterns based on the content of their tweets, even if they do not explicitly attach geographical locations to their tweets. They propose a new model, called *Full*, which showed significant performance gains compared with other research. Their model organizes regions with the Nested Chinese Restaurant Franchise (nCRF) method for discovering a hierarchical tree structure with unbounded width and depth, allowing for a more fine-grained representation of locations. They also compare different methods for sampling regional language models and find that sampling from Antoniak distributions is more accurate but takes longer compared to the minimal or maximal path method. They conclude that their model is useful for behavioral targeting and online advertisements.

A related approach can be seen from [8]. To improve the effectiveness of location-aware recommendation and analytic services, the authors propose using message content to predict a user's temporal location. They develop a machine learning (ML) technique to predict whether a message contains a user's location and whether that location corresponds to their present or future location. Extracting named entities from short messages proved to be challenging due to their informal nature and use of acronyms and non-standard abbreviations. To solve this challenge, the authors developed a customized named-entity extractor based on GATE, modifying the tokenizer, gazetteer, and semantic tagger. To distinguish whether the content of the tweet is about current or future locations, different classifiers were tested. The Maximum Entropy classifier showed the best performance, with a classification accuracy of 88.2%. They apply these techniques to two real-world datasets and provide insights into users' location properties such as granularity, associated time entities, and heat maps with high accuracy.

The thesis [14] is another Tweet context-based approach. The goal of the researchers was to identify the home location of Twitter users at the level of the city, time zone, or state using an ensemble of classifiers. To determine the location, they not only analyze the heuristics of tweet content, but also classify hashtags, history of visited places, time zones, and place names. For the classification, they mainly rely on existing classifiers introduced by related research. The results show that performance is generally higher for classifiers that discriminate between fewer classes, and hierarchical classification approaches have superior performance for city prediction. Place name classification achieved the highest recall and the visiting history scored the lowest recall.

A new method called FriendlyLocation for estimating a user's location on social media was proposed [15]. FriendlyLocation takes into account the strength of the relationship between users and uses factors such as the number of followers and interactions to predict the distance between them. By training a decision tree to distinguish between users who are likely to live near each other and those who are not. Example indicators to determine proximity were, users with many followers tend to be further away and user mentioning each other in posts usually are closer together. Their model is based on The Facebook model from [16] and adapted to Twitter. The authors found that their method improved location estimation compared to previous techniques. FriendlyLocation reduced the average error distance for 80% of Twitter users from 40 miles to 21 miles simply by analyzing information from the user's friends and friends of friends, which has significant implications for location-based services.

The goal of [17] was to track mobile phone owners in their vehicles without using position-determining services such as GPS. The author developed an app to track sensory data such as gyroscope, accelerometer, and magnetometer without permission of the user. Also, cell tower pings, as well as Wi-Fi/Bluetooth address harvesting, were conducted. The application collects this data in the background and sends it to servers as soon as an internet connection is available. After simulations of specific cities, the tool had a probability of higher than 50% to deliver a set of 10 routes, of which one was the actual route taken.

The potential risk of disclosing personal information on sites such as Twitter or Instagram is demonstrated by [18]. The authors explore the possibility of an automated approach to collect personal information without the consent of a social media user. The focus here lies on anonymity. Different methods are explored to achieve access to Twitter, Instagram, and Google APIs to collect data without being traceable. Cryptocurrency payments, encrypted mailboxes, using public WI-FI, and Tor networks are used to achieve the desired outcome of complete anonymity while collecting data of a user without their consent.

3.3 People-Based OSINT

[19] proposes a method to detect negative predisposition towards law enforcement and authorities. This is meant to reduce the risk of insider threats when appointing security

staff or decision-making personnel in critical infrastructures. To achieve insight into personality traits, user-generated content on YouTube is collected via YouTube's Rest-API and analyzed with an ML approach, including a comment classification method. The analysis found that 0.6% of comments and 3.7% of videos had a negative attitude, and 37% of users indicated a negative predisposition towards law enforcement. 75% of the users with negative predisposition were males, and 55% were between 16 and 35 years old.

Similarly, [20] tries to reduce inside threats by predicting private traits and attributes such as sexual orientation, ethnicity, religious and political views, personality traits, intelligence, happiness, use of addictive substances, parental separation, age, and gender. In this study, 58000 volunteers provided their Facebook likes and profile pictures, detailed demographic profiles, and participated in several psychometric tests. A prediction model was created using logistic and linear regression, which showed promising results. The model could predict homosexuality correctly in 88% of the cases, determine a democrat or republican with 85% accuracy, the relationship status with 67% accuracy, and drug usage in 68% of the cases.

Also, [21] proposes a method to extract statistics on the stress level of Online Social Networks (OSN) users over time to reduce the possibility of an insider threat. Here, data is collected through a custom-developed program that uses Facebook's API in conjunction with voluntary participants. The participants could control what information is shared with the researchers. The collected data was then categorized using an unsupervised flat classification machine learning technique. The results produced two clusters, which were able to classify users based on medium-to-high and medium-to-low stress scores. The findings suggested a correlation between users' OSN usage patterns and BAI stress scores.

A more practical approach to assessing risk using OSINT can be seen by [22]. The authors performed a vulnerability assessment of a company operating on the U.S. electrical grid. The authors used LinkedIn as a starting point and from there also utilized various free tools such as Maltego, to collect more data and social engineer employees. They could identify not only the company's network software, hardware, and key IT personnel but also information about their families. The study shows a severe venerability in the targeted company for cyberattacks and raises awareness of this topic.

The paper [23] raises awareness that political affiliation profiling is possible using openly accessible data which violates privacy and has potential consequences, such as workplace discrimination and social prejudice. YouTube's rest-API was used to collect data in relation to a specific political topic, and Multinomial Logistic Regression showed the best accuracy and performance to classify the collected data. The collected data were classified into the following three categories: *(i)* user information such as their profile, uploaded videos, subscriptions, and playlists; *(ii)* information about the videos such as the number of likes, dislikes, and tags; and lastly *(iii)* information about the comments such as their content as well as the number of likes and dislikes they received. Results show that it is possible with openly accessible data from YouTube to classify a user into Radical, Neutral, or Conservative positions on a specific political topic.

3.4 IoT-Based OSINT

The widespread usage of IoT devices in the healthcare sector and its potential privacy concerns are addressed in a recent study by [24]. They highlighted that IoT devices usually transmit their information among unprotected networks, where this information might be collected by OSINT tools. They then propose an anonymity encryption model to improve the protection of private healthcare data collected and transmitted by IoT devices.

The paper [25] discusses the increased usage of IoT devices and the need for protection against potential attacks. The authors propose an Open-Source Intelligence tool called Eagle-Eye, which can be used to detect IoT devices by harvesting publicly available information. The proposed tool integrates a Shodan search engine that performs OSINT queries and is able to display the information in a user-friendly format. The idea of this tool is to raise awareness of all IoT devices nearby and by doing so mitigate the risks of a potential attack.

A similar topic covered by [26] discusses the importance of system security in our digital world. The usage of IoT devices rapidly increases, and so does the value and potential of OSINT. The proposed tool *ShoBeVODSDT*, performs searches and collects information from open data sources and databases. This tool is meant to be useful for organizations, testers, scientists, developers, and governments in identifying potential vulnerabilities and improving protection.

The thesis by [27] covers the importance of integrating security measures into databases. Nowadays, NoSQL databases are increasingly used. Unfortunately, these are often weakly encrypted and have a higher risk of data breaches. The article talks about their tool ShoBEVODSDT, which performs penetration testing to assess the security and vulnerability of a database.

A recent study provides a review of the literature on certificateless signature schemes for the Industrial Internet of Things (IIoT) [28]. The author provides further literature showing that current schemes have already been broken and thereby rendered useless. A novel lightweight certificateless signature scheme is introduced, that has been proven to be fully secure against most current attacks while providing better efficiency than existing schemes.

3.5 Summary and Discussion

Table 3.1 compares the approaches and endpoints used by the studies from the Geolocation- and People-based OSINT sections to collect user data. The first column refers to papers that used the official Applications Programming Interface (API) to collect their dataset. Column number two shows papers that used a crawler for automated data collection. This is often done to bypass the API's limitations in regard to the amount of data that is allowed to be collected or the type of data that can be accessed via the API. Studies in the third column mocked a dataset. The fourth column refers to studies that created a custom app to run on mobile phone devices and collect information for

Table 3.1: This Table Shows the Approaches for the Collection of Datasets used by the Mentioned Studies from the Geo-location- and People-based OSINT Sections.

	API	**Crawler**	**Mocked Dataset**	**Custom App**	**Traditional Usage**	**Profile Provided**
Twitter	7, 9, 10, 11, 14	7	8	-	-	-
Facebook	7	-	-	-	-	16, 17
Instagram	7	7	-	-	-	-
Mobile Phone	-	-	-	13	-	-
YouTube	15, 19	-	-	-	-	-
LinkedIn	-	-	-	-	18	-

Table 3.2: Specific API usage when Data was Collected from Twitter

	Twitter
Steam-API	7, 9, 10
Rest-API	10
Unspecified	11, 14

their dataset. The fifth column depicts research that used their information source in a "traditional way". This means that they went to the official website, created an account, and tinkered their way through the platform to find the information they were interested in. In the last column, papers are represented where participants agreed to provide access to the researchers to their profiles.

It can be seen that Twitter is the most commonly used endpoint to conduct research. This might be due to the fact that Twitter APIs are easy to access and less restrictive, in regard to what can be collected, compared to other social network sites (SNS) such as Facebook or Instagram. Insight into what specific Twitter API was used can be seen in Table 3.2.

YouTube is also recently becoming a frequent source for collecting user data. [19], [23] show the potential insight into user orientations and behaviors that can be gathered from YouTube, and probably more research will inspect the video platform in the future.

Most studies looking into IoT-based OSINT focus on data privacy issues. IoT-based devices seem to be a frequent source of data leaks, and hence a common issue of research to improve encryption and data security.

Most research raises concerns about privacy and ethical issues with the usage of OSINT. The use of OSINT tool is often only recommended in situations of security threats such as terrorism or when choosing people for critical decision-making positions that have significant impacts on other people's lives.

In general, it appears that there is a scarcity of studies involving the integration of data from communication technologies such as LoRa with Open Source Intelligence. The potential amalgamation of OSINT with LoRa data may provide an opportunity to expand and enhance the depth of insight achievable through OSINT.

Chapter 4

Design

The integration of open source intelligence (OSINT) into LoRa data presents a unique opportunity for enhanced situational awareness, improved data collection, and enhanced decision-making. Leveraging LoRa data, which provides valuable information about device types, application scenarios, locations, and owners, allows investigators to gain a more comprehensive understanding of the situation and uncover previously unnoticed connections [29]. Unlike traditional OSINT practices that often start with a specific person or group and may overlook relevant connections, this approach enables investigators to expand their scope and potentially enhance the effectiveness of OSINT investigations in a variety of fields, from law enforcement to marketing.

To demonstrate the potential of this opportunity, a platform is being designed and implemented to apply OSINT to LoRa data. This implementation then undergoes testing with an illustrative scenario of a real event across multiple platforms that serve as sources of open-source data.

4.1 Platform

4.1.1 Current State

This thesis builds upon and extends an established code base. The code basis was created by [29] and already provides a solid functionality of Open-source intelligence on LoRa data. This section introduces the reader to the current state of the platform and describes the code, along with its functionalities and capabilities. In addition, it provides an overview of the frameworks and libraries in use, investigates design choices, and offers an evaluation of the platform's current performance.

Description of the Platform

The platform exists as a web application that compiles and runs on a local machine. The code uses a few Linux commands, which necessitates having a Linux or a macOS operating

system to run the code. The code base resides in two repositories and has a front end that operates in combination with two separate back ends.

The web application offers a platform where users upload LoRa data stored in a JSON format. Once uploaded, the platform autonomously processes and presents several related pieces of information. It maps the location where the LoRa data was collected and displays concurrent weather conditions, such as temperature and humidity, recorded at the same location and time. In addition, the platform fetches social media content from Twitter and YouTube that coincide with the location and time of the LoRa data. From Twitter, it fetches posts that match these parameters, and from YouTube, it gathers geotagged videos, providing metadata such as the title, description, channel owner, and thumbnail image. The platform extends its search to Facebook, identifying events or activities in the area that might be related to the collected LoRa data, thereby painting a more comprehensive picture.

One feature, only available in Spain at present, is the inference of housing data. The platform utilizes the Spanish website [30] that provides information about property and land ownership, square meters, construction year, and information about the property's value. This data isn't incorporated into the front end; instead, a link with the applied location search is provided. Ultimately, the coordinates stored in the LoRa data are leveraged to execute a targeted search query on the search engine Shodan. Shodan allows users to search for internet-connected devices, such as servers, routers, cameras, and other types of hardware and provides IP addresses, open ports, and other details [31].

Regarding visual representation, users select a specific day and device from the dataset once it's uploaded. This selection lets them inspect the number of packets transmitted per hour and the nature of these messages for that chosen day and can be seen in Figure 6.3. In the context of LoRa networks, the term "nature" refers to the characteristics or content of the transmitted messages. It generally pertains to the type of information or data conveyed through the LoRa network. For instance, the nature of these messages might include sensor readings, telemetry data, status updates, or any other relevant information communicated between devices in the LoRa network. To understand the nature of the messages, the platform parses the non-encrypted fields of the LoRa protocol and decodes the binary payload first, enabling users to gain insights into the specific information contained within the transmitted messages.

Upon data processing completion, the platform generates suggestions regarding potential connections between devices and individuals related to the gathered data.

The developer [29] designed and developed this platform during a Hackathon competition in 2022 in Alicante, Spain. The intent of this web application aligns with the reason this thesis exists, showcasing the yet unexploited potential of introducing OSINT to LoRa data. It's noteworthy that the quality of the current code base doesn't meet the standards of production-developed code. This condition stems from its development in a time-restricted competition where scalability and clean code aren't the top priority.

Description of the Codebase

As mentioned in the previous section, the code base consists of a front end in combination with two back ends. The front end is written in JavaScript and uses the client web framework Vue.js [32].

The application's front end is intentionally designed as a single-page application, eliminating the need for separate routes or multiple pages. Instead, it features a clean and straightforward structure, comprising various sections within the main page. These sections serve distinct purposes and seamlessly integrate all the functionality, ranging from data upload to the display of inferred information. Among these sections, the "Home" section provides the upload functionality. The "What" section displays general information about the uploaded LoRa data such as date, bandwidth, or code rate. This section is followed by the "Where" section, showing the specific location of the data on a map from Google Maps. After that, the "Suggestions" component presents recommendations about potential relevant information to this scenario. For example, it shows information that comes from the Shodan search engine about other devices that might have been involved. And lastly, the "When" section is mounted, showing retrieved videos from YouTube, collected posts from Twitter, and the weather that matched the location and timestamps of the uploaded LoRa data. Interestingly, the functionality of searching Facebook isn't displayed nor called from the front end. The repository provides a README file with simple instructions to install dependencies and run the code on a local machine.

The first back end, informally declared as the "What" API, is responsible for the functionalities of the previously described "What" section of the application, including the dataset upload. Its central role lies in managing JSON LoRa data from the upload and supplying endpoints to pertinent information that it subsequently parses and returns. To illustrate, the endpoint: "/days" retrieves all distinct days when a LoRa device has recorded data.

Additionally, this back end is responsible for performing the traffic analysis. This is achieved by employing a set of scripts designed for this purpose, making them a critical component of successful task execution.

The "What" back end is scripted in JavaScript and compiles to Node.js, which in turn runs on your local operating system. Correspondingly, a README file is provided to facilitate the setting up and operation of this back end on a local machine.

The second back end called the "Where and When" back end, handles the earlier mentioned sections from the front end. Here are all the endpoints to retrieve the map, get the weather, collect the YouTube, Twitter, and Facebook feeds, and get suggestions about other relevant devices coded. When calling an API, there is often a helper function that parses uploaded information into the API query for another function that executes and handles the returned data. This back end is written in Python in a Jupiter Notebook, with Pandas imported for data handling alongside NumPy for computations and a few other libraries such as GeoPy, Folium, PyProj, and Flask for creating a RESTful server. The repository doesn't include a README file in this case.

Overall, it is notable that the code was produced in a short amount of time. No tests were written and the handling of API keys and access tokens is insecure since they are

often just written in the code. This leads to the assumption that this is the main reason the second back end is private while the first back end is public. Also, documentation is rarely found. In terms of maintenance, the code was written in 2022 and has not changed since then.

Framework Overview

To better understand the relationship between different components of the platform, Figure 4.1 presents a high-level overview of the existing framework. This diagram illustrates the functionalities described in the previous subsections, primarily focusing on the flow and processing of the uploaded LoRa dataset (indicated by the green lines and numbers).

The platform fundamentally consists of two interconnected components: the front end and the back end. The front end is responsible for user interactions such as the dataset upload, selection of specific devices, and the presentation of gathered data and insights.

The back end, which is split into two sections as detailed in the subsection 4.1.1, handles data acquisition and preprocessing as well as the intelligence aspect of the system. The intelligence refers to the processing and analysis of uploaded LoRa data, the connection to various online sources for gathering additional data (such as social media posts and weather conditions), and the generation of inferred information and suggestions.

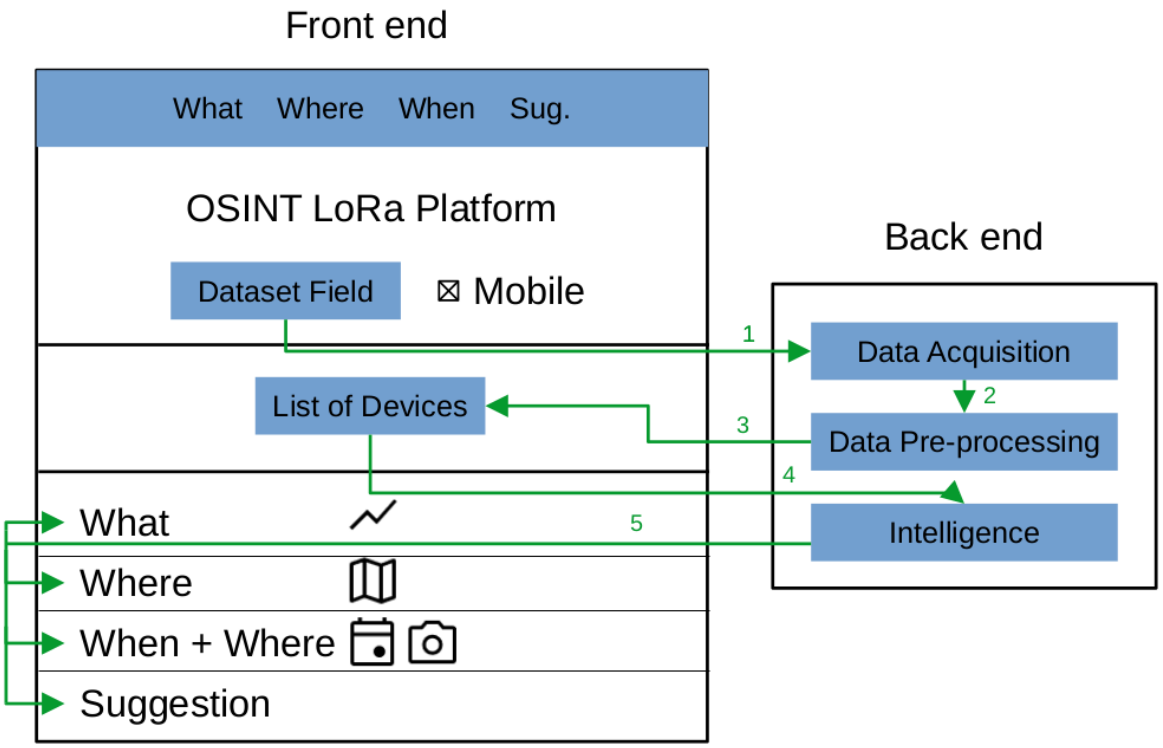


Figure 4.1: Framework Overview Current Codebase

Design Choices

In terms of design choices, the only topic of interest is the setup of two back ends. The use of two separate back ends potentially causes drawbacks in terms of understandability, maintainability, and setup complexity when running the code for the first time. The decision to implement two back ends may be an attempt to abstract the functionality, better coordinating the implementation of the code across multiple team members. This strategy might help to avoid merge conflicts and provide a better development strategy in a competitive setting where time is essential. Nonetheless, for this application, a single back end handling all requests represents a better practice in terms of software standards.

Performance Evaluation and Reproducibility

To evaluate the performance of the developed web application, a custom scenario is manually set up across Twitter and YouTube that matches with LoRa datasets. These datasets are provided in the repositories, to reproduce the findings demonstrated in a presentation video created at the 2022 competition. This is still possible and works at the time of writing. Other than that, performance evaluation is challenging. It is possible to upload a different LoRa dataset than the ones provided, to use OSINT on a real scenario, but in that case, a comparison between what is collected and what potentially is the maximum amount of openly accessible data of this scenario is hardly possible.

In terms of functionality of the implemented features, every feature mentioned in the description of this platform section works and shows the desired outcome with one exception. The functionality of inferring event data from Facebook is currently not functioning and is not incorporated into the application.

4.1.2 Proposed Design

Current Code Base

The existing functionality of the platform remains intact, as the majority of the implemented features continue to operate effectively and serve their intended purpose. Consequently, both back ends maintain their current form, as they were after finishing the Hackathon. The front end, however, undergoes some changes. The Vue.js-based front end receives improvements in terms of styling visualization and user experience. The incorporation of new features necessitates these enhancements to ensure all data is presented in a clear, coherent, and user-friendly manner.

Contribution

What, then, does this thesis contribute to the platform? As mentioned earlier, the current state is capable of making suggestions about who might be involved in the captured

scenario. This capability, in combination with the collected information about possible events that might have taken place at that location and time, represents the starting point of this thesis. The goal is not only to find as much information as possible about the involved person but also to analyze his/her peers and establish connections to produce insight into who else might have been attentive at that event, respectively at the collected location and time from the LoRa data."

4.1.3 Architecture

As highlighted in Section 4.1.2, the existing platform derives likely event names and potentially associated individuals based on the collected data. In the context of this tailored scenario, such information comes from various social media platforms. For example, information about a likely individual involved is obtained by extracting the name from a geotagged YouTube video. Additionally, the event name is sourced from a Twitter account that has two posts timestamped to coincide with the occurrence of the event and the available LoRa data. Figure 4.2 shows a simplistic structure from which this thesis continues. Note that the platform already extracts and displays much more data than just these two dimensions. However, as this study primarily builds upon these two facets, other features aren't extensively detailed in the architecture description.

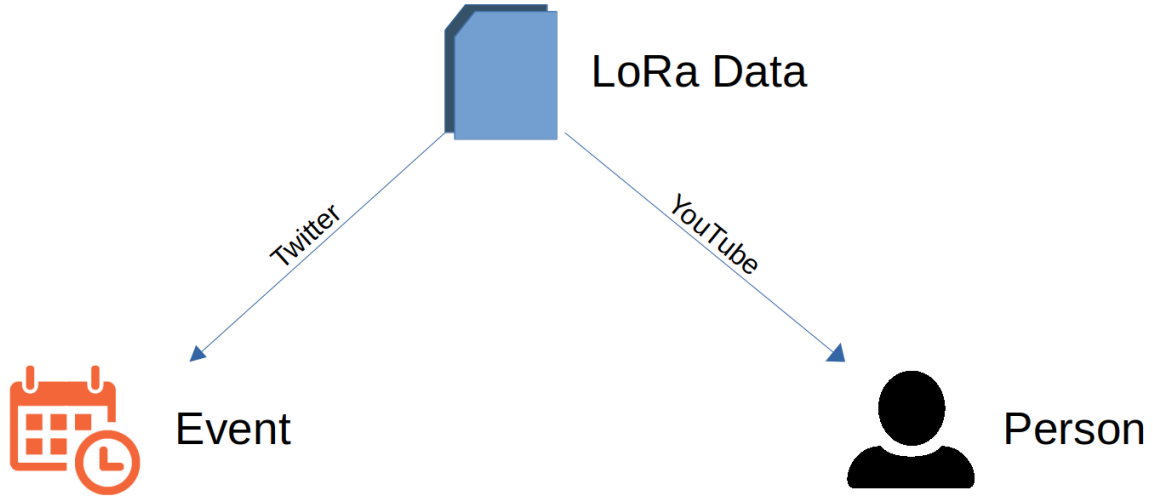


Figure 4.2: Baseline Approach of the Platform

Therefore, to deduce insights about the identified subject and to determine who else may be present, comprehensive web scraping is carried out. Specific endpoints of LinkedIn and Google Scholar are used as primary data sources, and the information gathered from them is analyzed. In addition, the search is extended to include data scraping from companies related to the subject and the profile pages of their employees.

The procedure for deriving information from the primary subject involves these actions: Verify the complete and accurate name, look up the individual's LinkedIn profile, scrape

the data from the LinkedIn page, pull as much data as possible, and deduce details about the current employment status. Using the verified name and the inferred place of employment, the company's profile page is sought out and information on collaborations and additional personal details is gathered. This process provides another connection between the primary subject and the Event via the organizer and is shown in Figure 4.3 marked with red lines.

The task of creating a dataset of potential participants is carried out by scraping data from the endpoint of Google Scholar. This process involves identifying the Google Scholar page of the main subject, scraping all listed publications, and extracting the names of all co-authors who have collaborated with the primary subject. Following this, a second round of data collection is initiated to broaden the dataset. Additional individuals are subjected to the same data-scraping process. The selection of these individuals is determined by their current employment status. If they are identified as employees of the organizer's company, they meet the criteria, and their data is also scraped. This method results in a comprehensive dataset of potential participants, which is illustrated by the green lines in Figure 4.3.

Finally, the remaining data collection involves enhancing the understanding of the event project, the organizer, and the companies and institutions involved. This is achieved by carrying out searches using specific keywords with the intent of identifying and extracting relevant documents that include these keywords. The extracted documents and text passages are subsequently analyzed using a text classifier to gather information about project names and participating entities. When projects are identified, an additional scraper is deployed to gather information on participants, publications, and the companies and universities involved. This process is depicted by a yellow line in Figure 4.3.

Once data collection is complete, the concluding analysis is conducted. During this phase, the compiled data is inputted into a closeness algorithm. This algorithm examines all the scraped parameters to establish connections between the main subject, the scraped individuals, the entities involved, the event organizer, and consequently, the event itself. All entities in the dataset receive a matching score that indicates how likely it is that this person attended the event. This process is depicted in Figure 4.3 with purple lines.

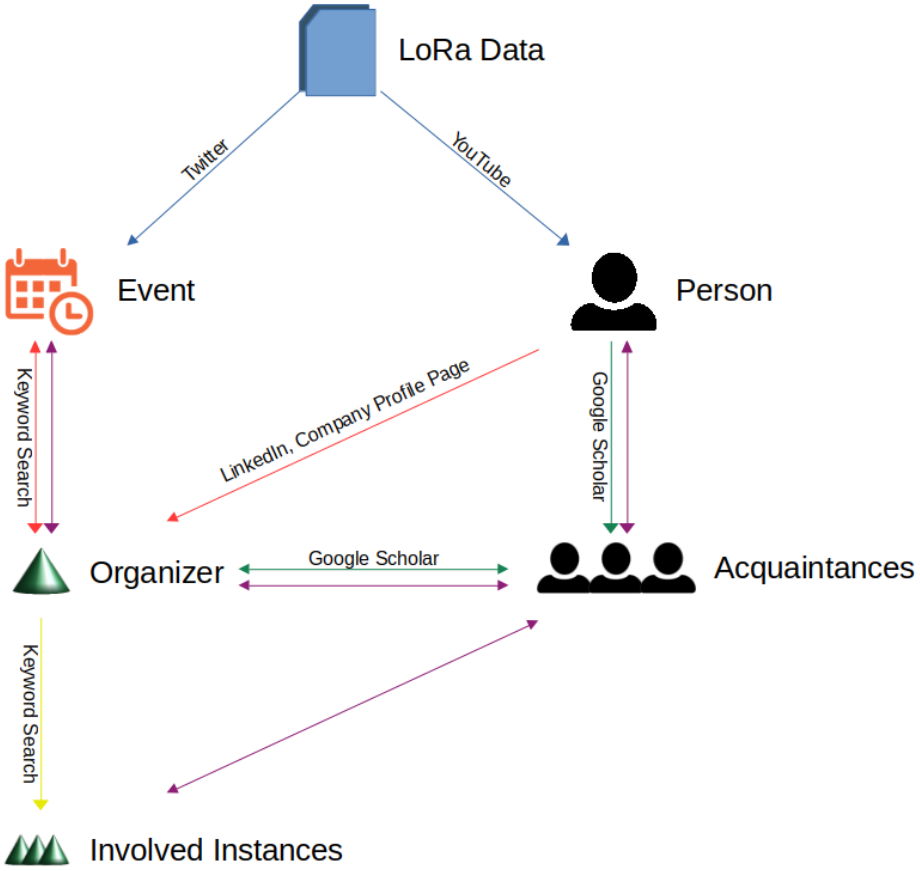


Figure 4.3: Participant Inference Approach

Detailed information on the data collection and analysis procedures is found in Chapter 5, which elaborates on the implementation specifics. Regarding the front end architecture, the features mentioned earlier are split into three sections and are implemented and displayed in the same Vue.js-based front end as the existing code base. The scraping and processes divide into the requests of scraping LinkedIn, scraping the primary subjects' employee page and performing the collection and analysis of who else attended.

4.1.4 Enhanced Framework Overview

To enhance the conceptual understanding of the contributions of this thesis, Figure 4.4 demonstrates a new flow of data and computation. This figure is an extended version of the previously presented Figure 4.1, with the additions highlighted in orange. The new back end task, designed as a large-scale data acquisition process, is illustrated in the same color. It integrates additional data from the primary subject's LinkedIn profile, the employee's company page, and a list of potential event attendees.

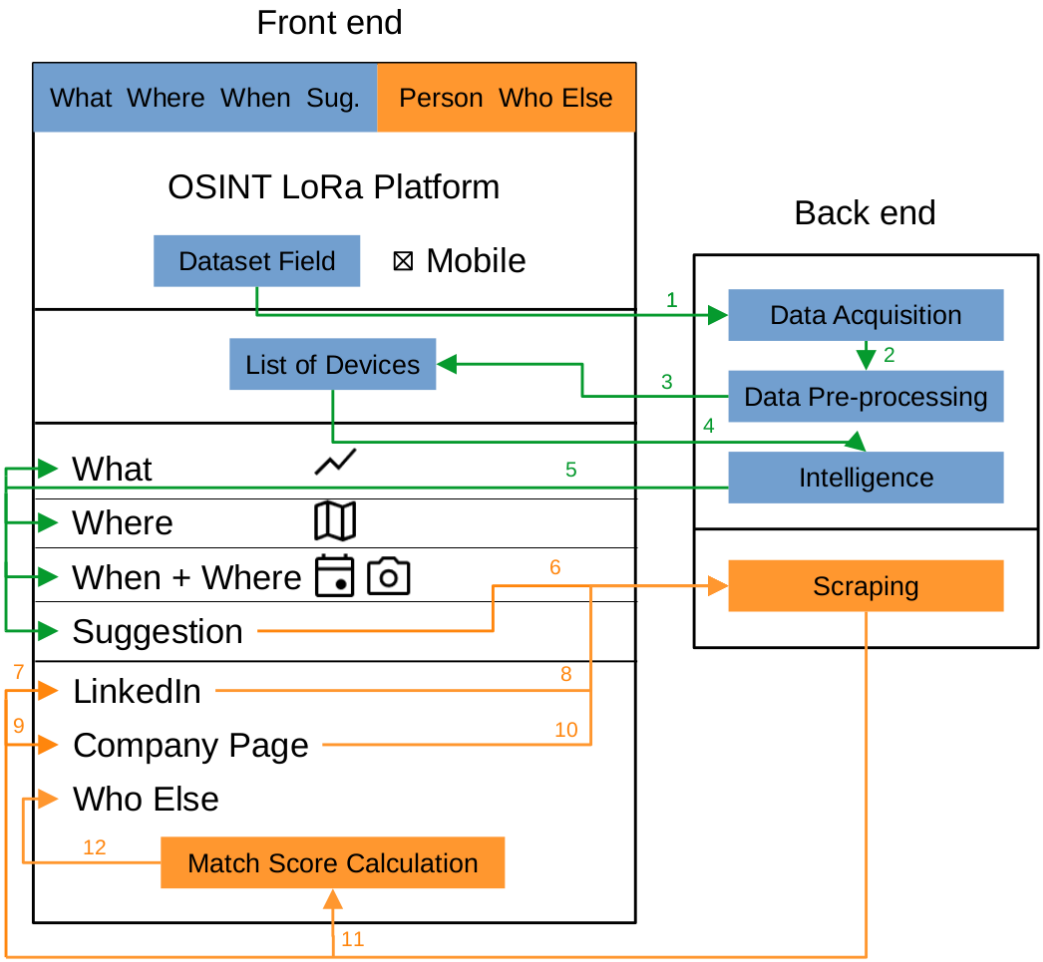


Figure 4.4: Framework Overview of the Proposed Platform

These data scraping enhancements are reflected in the front end, with three new areas displaying the information gathered. Moreover, to increase the platform's responsiveness, the calculation of the match score, which provides an indication of the likelihood of someone's attendance at the event, is performed in the front end. More detail about this design choice can be found in Section 5.4.3.

4.2 Scenario

This section aims to offer a broader understanding of the custom scenario developed for this Hackathon project. The scenario is analyzed with the existing platform and is further examined for this thesis.

The data used, referred to as the LoRa dataset, is collected between October 4 and October 5, 2022. The geographical coordinates for the data collection point are 38.263375 latitude and -0.737125 longitude, which corresponds to Alicante, Spain. To obtain the ground truth for the localization, data from a Global Positioning System (GPS) is used. This

assumption is based on the parallel progress of another team working on localization. The actual LoRa traffic is collected using the web platform provided by The Things Network [33].

Alberto Huertas is the primary subject at the location during the data collection period. He, along with his team, created posts and made public geotagged videos as a part of the scenario to test their platform. This is done in the context of a private Hackathon event organized by armasuisse, which sees the participation of 16 individuals. The identities of the participants are available and are used later in the study to evaluate the accuracy and precision of the predictions.

Chapter 5

Implementation

This section guides the reader through the specifics of implementing the additional features added to the platform, thereby representing a prototypical implementation of the architecture proposed in the previous section. The details are divided into four distinct sections. The initial section offers a broad overview of commonly used libraries and frameworks, as well as high-level structural information. The second segment concentrates on the process of data collection from LinkedIn, while the third portion emphasizes the scraping of data from the employee's company page. The fourth and most substantial section dedicates itself to deducing information about potential other visitors to the scenario.

The updated front end can be found in the repository [34], while the third added back end can be accessed in the repository [35].

5.1 General Overview

Regarding the architectural structure, this thesis proposes the addition of a new back end, rather than modifying the existing two. This provision provides a stable operating environment for the additional features and avoids the potential introduction of bugs within the current back ends. Notably, the "Where and What" back end operates on a `Jupyter notebook`, which proves to be suboptimal, leading to run-time errors during lengthier scraping processes. This novel third back end operates using the `Flask` framework in Python. The straightforward nature of `Flask` is well suited for this project, as the back end simply needs to handle a limited number of POST requests. As discussed in the design chapter, the scraping and analysis procedures separate into three individual sections, a structure that mirrors in the implementation of the POST endpoints. These endpoints receive further elaboration in the upcoming sections.

In this project, the majority of the web scraping tasks are accomplished using a collection of the same frameworks. `Selenium`, a tool primarily used for automating web applications, combined with `WebDriver`, proves effective for most use cases. This combination utilizes a `Chromium` browser, an automation framework that accepts and executes commands. The synergy between these two components enables efficient and effective web scraping across the varied tasks within this project.

5.2 LinkedIn Data Collection

LinkedIn provides detailed and accurate user information. While our primary focus was extracting data about the user's employment status and associated company or institution, we aimed to create a more comprehensive profile. Thus, our scope extended beyond this, aiming to capture all available information displayed on the LinkedIn profile page.

5.2.1 Trial Approaches

The task of collecting data from LinkedIn turns out to be more challenging than initially anticipated. In the beginning, the strategy is to use existing open-source code that could facilitate the scraping of specific LinkedIn profiles. After an exploratory phase, a particular GitHub repository is discovered [36], which offers the desired capabilities of automated extraction of profile information. Unfortunately, despite appearing to be actively maintained, the repository does not yield the expected results due to persistent failures in its fundamental functionalities. It seems that LinkedIn constantly changes the naming schemes of its web page components to avoid the usage of automated scrapers. Moreover, the rapid request frequency deployed by the tool may have triggered LinkedIn's bot detection mechanisms, leading to several accounts being blocked. When the code executes successfully, it encounters difficulties in extracting all data from a profile due to the previously mentioned alterations in naming conventions.

A revised strategy entails adapting the scraper from [36]. The modified approach seeks to navigate LinkedIn to reach the target profile page, then download the complete source code as HTML for local analysis. Here, `BeautifulSoup` is employed to facilitate the process. Unfortunately, the extracted HTML is devoid of any LinkedIn user information and appears empty. This outcome must be a consequence of LinkedIn's security measures and its method of data presentation.

Pursuing a unique strategy involves experimenting with capturing profile screenshots and utilizing image-to-text conversion tools for data extraction. The evaluation encompasses multiple libraries such as `Selenium`, `Webkit2png`, `Puppeteer`, and `imgkit`. Despite these efforts, this approach leads to similar outcomes as the HTML extraction method, with LinkedIn pages populated with placeholders instead of real content.

Another disadvantage of all the previously mentioned strategies is the necessity of using a legitimate LinkedIn account. Consequently, the user being scraped receives a notification about the profile visit from the specific account.

5.2.2 Established Technique

Following unsuccessful attempts to achieve desired results, additional investigation unfolds to explore tools capable of scraping LinkedIn. Proxycurl, developed by Nebula [37], emerges and deems suitable for the task. This tool offers an API endpoint enabling the

scraping of LinkedIn profiles and returning comprehensive data. The only prerequisite is supplying the specific URL of the profile to be scraped.

To infer the URL, a simple scraper performs a Google search for the desired name in conjunction with the keyword "LinkedIn". This follows by the extraction of the URL of the first search result 5.1. Once the URL is determined, it is then passed to the Proxycurl tool to scrape the corresponding profile. This functionality can be accessed via the POST request: http://localhost:5001/linkedIn if the code runs on your local machine.

5.3 Scraping Company Pages

The method used to search for information about the main subject in relation to his employer is straightforward. The scraper is fed the subject's name along with the name of their current employer, extracted from LinkedIn, and a Google search conducts using these two parameters. Frequently, the first search result yields the desired company page about the subject. However, further testing reveals that LinkedIn pages or publication pages sometimes display as the top search results. To resolve this, a simple exclusion criterion creates that skips to the next search result in case one of a set of keywords is present in the link which can be seen in Listing 5.2.

Listing 5.2: Finding the First Relevant Search Result

```
search_results = driver.find_elements(By.CSS_SELECTOR, 'div.g a')
target_link = None
for link in search_results:
    href = link.get_attribute('href')
    if "linkedin" not in href and "scholar.google" not in href and
        "ieee" not in href and "researchgate" not in href:
        target_link = link
        break
```

Upon locating a search result that meets the established criteria, the scraper selects and navigates to that page. Once on this page, the driver scans for elements with specific class names, including name, email, phone, fax, bio, funding, and projects, among others. When a section with a matching class name is identified, the corresponding data is extracted. When deployed locally, the endpoint for this operation can be accessed at `http://localhost:5001/company`.

5.4 Identifying Involved Parties

Determining additional participants likely to engage with the constructed scenario emerges as the most complex task, necessitating the resolution of various challenges. The nature of the scenario and its participants amplify this complexity.

Listing 5.1: Extracting the LinkedIn URL

```
def get_linkedin_url(person_name):
    # Set up Chrome WebDriver
    options = webdriver.ChromeOptions()
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('--incognito')
    options.add_argument('--headless')
    driver = webdriver.Chrome(options=options)

    # Navigate to Google
    driver.get('https://www.google.com')

    # Wait for the cookies dialog to appear and click "I agree"
    WebDriverWait(driver, 10).until(presence_of_element_located((
       By.ID, 'L2AGLb')))
    driver.find_element(By.ID, 'L2AGLb').click()

    # Search for the LinkedIn profile
    search_query = f'{person_name} LinkedIn'
    search_box = driver.find_element(By.NAME, 'q')
    search_box.send_keys(search_query)
    time.sleep(0.5)  # Add a 500ms (0.5s) delay
    search_box.send_keys(Keys.RETURN)

    # Wait for search results to appear and get the first result's
        URL
    WebDriverWait(driver, 10).until(presence_of_element_located((
       By.CSS_SELECTOR, 'div.g a')))
    linkedin_url = driver.find_element(By.CSS_SELECTOR, 'div.g a')
       .get_attribute('href')

    # Print the first result's URL to the console
    print(f'LinkedIn profile for {person_name}: {linkedin_url}')

    # Close the browser window
    driver.quit()

    return linkedin_url
```

The initial two scrapers enable us to gather comprehensive information about the primary subject and their professional life. For the concluding scraping stage, there are three primary data points considered essential, and consequently, they are relayed to this process. These include the subject's full name, the company where the subject is employed, and the key companies with which they engage professionally.

5.4.1 Google Scholar

In this context, Google Scholar emerges as an appropriate endpoint for creating a dataset of potential participants for further analysis. The scraping process begins with a straightforward Google search, appending the term "Google Scholar" to the name of the primary subject. The first search result yields the URL to the profile of the individual of interest.

Initial Iteration

The first extraction procedure requires scraping all publications to derive the names of all co-authors. The original strategy is to gather the names from the publications displayed on the profile page. However, this proves to be inadequate as sometimes not all co-authors are listed, and only the initials of the first name are provided. Therefore, the scraper has to individually open and inspect every publication to extract the full names. In order to perform this task in a loop, the total number of publications has to be determined first. To accomplish this, the scraper has to click on the "show more" button on the profile page until all publications are loaded, and the total count is displayed at the bottom. Subsequently, a looped scraper iterates through all the publications in sets of 20. This approach is chosen because the default profile page layout only displays 20 publications at a time. Once the first 20 publications are scraped, the driver modifies the URL to start from the 21st publication, and the iterative process is restarted, as shown in Listing 5.3.

Listing 5.3: Iterating through Multiple Pages of Publications

```
try:
    # Iterate through multiple pages of publications
    for i in range(20, publications_to_iterate, 20):
        next_url = f"{url}&cstart={i}"
        driver.get(next_url)
        publications += extract_publication_details(driver)
```

After compiling a list of all publications, the collaborations get tallied and a dataset with all individuals is created. One challenge faced during the collaboration count is the inconsistency of names featuring special characters. For instance, the name 'Gregorio Martinez Perez' appears in different variants, such as 'Gregorio Martínez Pérez' and 'Gregorio Martinez Pérez'. To tackle this issue, the introduction of a helper function becomes necessary to standardize special characters as they get scraped from the publications on Google Scholar. The `Unicode` library is utilized for this purpose, as can be seen in Listing 5.4.

Listing 5.4: Normalizing the Author Names

```
# Split authors by comma and apply normalization to each author
normalized_authors = [unidecode(author.strip()) for author in
    authors.split(",")]
return ", ".join(normalized_authors)
```

At this point, this dataset contains the full name, the number of how many times the main subject collaborated with this person, and an iteration label that indicates that this person's origin is from the first Google Scholar scrape of the primary subject.

The subsequent steps involve a comprehensive scraping procedure for all co-authors. In this stage, the scraper cycles through each individual, locating their Google Scholar profiles. Within these profiles, it is possible to extract the associated academic institution or workplace, in addition to any profile picture uploaded to their page. This scraper uses the same URL extraction technique as previously illustrated in Listing 5.1. After determining the URL, the driver navigates to the profile page to harvest the specific class components encompassing both the image and the associated institution.

The scraping process proves to be time-consuming, primarily due to the iterative nature of the scraper. To expedite this procedure, a new structure is adopted, as shown in Listing 5.5. The function **scrape_collaborator_info_on_google_scholar(collaborations, iteration, top_n)** plays a crucial role in extracting information from the Google Scholar profiles of collaborators. Initially, it arranges the collaborators based on their number of collaborations in descending order. Utilizing Python's `multiprocessing` module, it employs the Manager to create a shared progress list, which aids in monitoring the progress of tasks across multiple processes. After this step, the code creates a pool of processes. The size of this pool is determined by choosing the smaller number between the total number of collaborators and the total number of CPU cores available in the system. This is done to optimize the use of system resources and to ensure efficient time performance. It then initiates the scraping process by applying the **get_collaborator_info()** function to each collaborator. This function visits each Google Scholar profile and accumulates relevant information. The purpose of incorporating the `tqdm` framework is simply to offer real-time updates on the scraper's progress.

Follow-up Iterations

As noted at the start of this chapter, the extraction of essential information such as the full name, the employer, and a key company that either provides regular funding or collaborates consistently with the main subject has already taken place. The relevance of this key company becomes apparent at this point. Given that the dataset from the initial iteration contains details about every person's current institution or company, a helper function scans all entries for matches with the key company. If an individual affiliated with the key company is identified, the same data extraction process employed in the initial iteration is implemented. However, this time, all the individuals added to the dataset get tagged as being from a secondary scraping iteration. If several individuals are associated

Listing 5.5: Extracting Information for Every Collaborator

```
def scrape_collaborator_info_on_google_scholar(collaborations, iteration, top_n):
    sorted_collaborators = sorted(collaborations, key=lambda x: x.get("Collaborations"), reverse=True)
    num_collaborators = min(top_n, len(sorted_collaborators))

    with Manager() as manager:
        progress = manager.list()  # shared between processes
        pool = Pool(processes=min(num_collaborators, cpu_count()))
        # prepare the arguments for get_collaborator_info
        to_scrape = [(col, progress) for col in sorted_collaborators[:num_collaborators] if
                     col.get("Iteration") == iteration]

        # Use a try/except block to handle exceptions during multiprocessing
        try:
            result_obj = pool.starmap_async(get_collaborator_info, to_scrape)  # returns a result object
        except Exception as e:
            print(f"An exception occurred during multiprocessing: {str(e)}")
            print(traceback.format_exc())
            pool.terminate()

        # progress bar
        with tqdm(total=len(to_scrape)) as pbar:
            # left out since it is not relevant for the core functionality

        result = result_obj.get()  # get the result
        pool.close()
        pool.join()

    # Handle remaining collaborators (if any)
    remaining = [col for col in sorted_collaborators[num_collaborators:] if col.get("Iteration") == iteration]
    for collaborator_info in remaining:
        collaborator = collaborator_info.get("Name")
        count = collaborator_info.get("Collaborations")
        result.append({"Name": collaborator, "Collaborations": count, "Institution": None, "ImageLink": None,
                       "Iteration": iteration})

    return result
```

with the key company, the scraping process is replicated for each of them. It's worth noting that this match-checking with the key company is only carried out on the original dataset to prevent an endless loop.

In order to reduce workload, another helper function checks if the person, who is being scraped in the second iteration, is already included in the first dataset before initiating the individual profile page scraping. If the function finds that the person is already included in the first dataset, this individual is eliminated from the second dataset to avoid the unnecessary task of scraping the same information twice. After each scraping iteration is completed, the datasets are combined.

Normalize Institution Name

After generating the dataset containing individuals of interest, the following step involves an initial data clean-up process concerning the institution names. While utilizing Google Scholar for web scraping, there are instances where the institution names would appear in a foreign language. The proposed solution to this issue is to ensure that the web page always loads in English.

To implement this, the Python module `urllib` is utilized. It is used to modify the URL such that the language parameter is consistently set to "en", thereby ensuring that the page always loads in English, as referenced in the Listing 5.6.

Listing 5.6: Ensuring the Page is Loaded in English

```
# Modify the language parameter to ensure English
    query_params = urllib.parse.parse_qs(parsed_url.query)
    query_params['hl'] = 'en'  # Set the language to English
    modified_query_string = urllib.parse.urlencode(query_params,
       doseq=True)
```

Unfortunately, this solution only solves the issue in certain instances. It appears that Google Scholar allows users to either choose their institution from a list of verified institutions or input it manually as a string. For the former, the institution name is language-dependent and will be translated if the page is opened in a different language. However, for the latter case, where the institution name is manually entered as a string, the language selection becomes irrelevant. Regardless of the language setting, the manually inputted string will always display as it was entered.

Addressing this issue involves incorporating the OpenAI API into the project. Each entry in the dataset is cycled through and the large language model `GPT-3.5` is queried to determine if the institution name is in English. If not, the model is asked to return the English equivalent of the word. To minimize the number of requests, a temporary list is established to keep track of all already checked institutions. If an institution that has been previously verified reappears in the iteration, the English name can be directly fetched from this list, thereby circumventing the need to make another request to `GPT-3.5`, as illustrated in Listing 5.7.

Listing 5.7: GPT-3.5 Request to Translate Institution Names to English

```
1
2  def chat_with_gpt3(question):
3      url = "https://api.openai.com/v1/chat/completions"
4
5      data = {
6          "model": "gpt-3.5-turbo",
7          "temperature": 0,
8          "messages": [
9              {"role": "system", "content": "You are a helpful
                   assistant."},
10             {"role": "user", "content": question}
11         ]
12     }
13     response = requests.post(url, headers=headers, data=json.dumps
           (data))
14     response_json = response.json()
15
16     if 'choices' in response_json:
17         return response_json['choices'][0]['message']['content']
18     else:
19         return None
20
21
22 def update_institution_names(dataset):
23     checked_names = {}
24
25     for entry in dataset:
26         institution = entry.get('Institution')
27         if institution and institution not in checked_names:
28             question = "The following name of an institution, is
                   this in English or in a different language?" +
                   institution + ". if it is in English please return
                   the same name. If it is not in English please
                   return the name of the institution in English. Can
                   you make sure to just return the name of the
                   institution and don't say anything else."
29
30             response = chat_with_gpt3(question)
31             checked_names[institution] = response
32
33         entry['Institution'] = checked_names.get(institution,
               institution)
34     return dataset
```

One challenge faced when working with `GPT-3.5` is obtaining a response that contains solely the English version of the institution name, without any additional dialogue. Following a period of experimentation, our most effective solution involves explicitly stating in our query that the response should only include the institution name and nothing more. This is combined with setting the 'temperature' parameter to zero as depicted in Listing 5.7.

The 'temperature' parameter influences the randomness and creativity of the language model's responses. By setting this to zero, it ensures that the output is focused and deterministic. With these two adjustments in place, answers that exclusively contain the name of the institution, free of any extraneous content, are consistently obtained.

5.4.2 Company Reports

The following step entails gathering more details about the event and its affiliated partners. Given that a key company has already been identified, and the initial code base extracts the event's name from a Twitter post, the decision is made to conduct a keyword search on Google. This search includes the event's name, the key company's name, and the year, as obtained from the collected data.

It becomes evident that the search often yields annual reports, typically stored in PDF format which presents challenges for traditional web scraping methods. To access the information within these PDFs, the scraper initially checks the link. If the search result is a PDF, the entire file is downloaded. To effectively carry out a keyword search and extract relevant sections, the PDF is converted into HTML, a process shown in Listing 5.8. This step is crucial for identifying the relevant start and end points of a paragraph once a keyword search hits a match.

The transformation from PDF to HTML is accomplished using the `PDFMiner` library. Initially, the 'utf-8' codec is set, ensuring the avoidance of special characters in the resultant HTML file that might cause errors in subsequent analysis. An instance of the `LAParams` class, where 'LA' stands for "Layout Analysis", is also established. This object assists in fine-tuning the layout analysis for PDF files, adjusting parameters such as line margin or word margin among others.

In the next step, an `HTMLConverter` object is created with the aid of the `PDFResourceManager`. This resource manager object is crucial for managing shared resources like fonts and images that are a part of the PDF file. Following this, the function employs a `PDFPageInterpreter` to process and translate the PDF pages into an HTML format. This is accomplished by opening the PDF file in binary read mode, iterating over each page, and processing them via the interpreter.

Lastly, it's essential to close the `HTMLConverter` instance (referred to as a 'device') upon completion of the processing. Doing so ensures that all the output has been correctly written to the HTML file, and all resources used in the conversion, like memory or file handles, are properly released. Therefore, this function successfully transforms the PDF file into a structured HTML format, paving the way for subsequent data examination.

Listing 5.8: Converting a PDF File to an HTML File

```
def pdf_to_html(pdf_file, html_file):
    rsrcmgr = PDFResourceManager()
    codec = 'utf-8'
    laparams = LAParams()

    with open(html_file, 'wb') as output_file:
        device = HTMLConverter(rsrcmgr, output_file, codec=codec,
           laparams=laparams)
        interpreter = PDFPageInterpreter(rsrcmgr, device)

        with open(pdf_file, 'rb') as input_file:
            for page in PDFPage.get_pages(input_file):
                interpreter.process_page(page)

        device.close()
```

With the company report converted into a more manageable format, a two-fold analysis is initiated. Firstly, the reports are examined for instances of names present in the dataset. Often, these company reports incorporate a section on publications, wherein a helper function is utilized to identify these names and calculate the number of mentions for each. Subsequently, a new field gets added to every entry in the dataset, populated with an integer that represents the number of times that name is mentioned in the report.

Subsequently, a keyword search is conducted to extract more information about the event. Keywords such as "Hackathon", "LoRa", or "OSINT" are examined in the company reports. If a match is detected, the entire paragraph is extracted for further analysis.

Analysis of Extracted Paragraphs

Upon extracting paragraphs from the report that correspond to certain keywords, the following task involves sifting through these paragraphs to pinpoint information of relevance, such as project names or company names. To facilitate this, an initial attempt is made using the open-source text classifier `spaCy` [38]. Experiments are conducted with their pre-trained pipelines - en_core_web_sm, en_core_web_trf, and en_core_web_md, however, their effectiveness falls short when it comes to extracting the targeted keywords. Consequently, we turn to `GPT-3.5` to carry out the same task.

Similar to the previous utilization of OpenAI's large language model, the temperature parameter is set to zero. The model is instructed to parse the given paragraph and extract company and project names, asking it to return a comma-separated value (CSV) list of these keywords of interest and nothing more. Interestingly, an occasional occurrence of additional formal chat is encountered from this task, as well as an occasional inability on the model's part to locate a keyword of interest. This leads to instances where no comma-separated value list is returned at all. To overcome this problem, the request is programmed to repeat for three iterations in the event of a failure, when an out-of-bounds

error occurs due to the absence of a returned list. These issues are encountered rather rarely, hence the sufficiency of giving it three tries to return a CSV list. This approach can be seen in Listing 5.9.

Listing 5.9: Analyzes Extracted Text with GPT-3.5

```
for attempt in range(3):
    try:
        response = requests.post(url, headers=headers, data=json.
           dumps(data))
        response_json = response.json()

        print("Tried with GPT-3.5")

        content = response_json['choices'][0]['message']['content'
           ]
        extracted_strings = re.findall(r'"([^"]*)"', content)

        return extracted_strings[0]
    except IndexError:
        print(f"GPT-3.5 failure on attempt {attempt + 1}, retrying
           ...")

    return
        print("GPT-3.5 failure")
```

With the additional company or project names identified, the subsequent scraping operation aims to harvest information related to publications and partner companies. The scraper performs a web search, landing on the websites of the identified projects or companies, and hunts for sections such as ”Publications” or ”Partners”.

For the ”Publications” section, if any are identified, the names associated with these publications are extracted and compared to our initial dataset collected from Google Scholar. Each entry in the dataset receives an additional field indicating whether the individual has co-authored a paper with any of the companies or projects identified. If an individual's name is not found within the dataset, it is added.

In the case of partner companies, websites commonly contain sections that list such associations, which our scraper is programmed to find. Project websites often denote partner companies via links to their respective web pages. The scraper extracts these links and stores them in a separate dataset. This dataset is then passed to `GPT-3.5`, employing the same parameters used earlier, instructing the model to return a list of company names corresponding to the inputted links. This process, as illustrated in Listing 5.10, executes without any glitches and provides the desired outcome.

Having generated a list of companies or institutions involved with the event, our next step is to traverse the dataset of individuals to determine if any individual is associated with a company or institution on that list. Since the association of every person is already being collected from Google Scholar, this becomes a simple comparison task. Each entry in the

Listing 5.10: Inferring Company Names from Links

```
def chat_with_gpt3(question):
    url = "https://api.openai.com/v1/chat/completions"

    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {api_key}"
    }
    data = {
        "model": "gpt-3.5-turbo",
        "temperature": 0,
        "messages": [
            {"role": "system", "content": "You are a helpful
               assistant."},
            {"role": "user", "content": question}
        ]
    }
    response = requests.post(url, headers=headers, data=json.dumps
       (data))
    response_json = response.json()

    # Extract the assistant's reply
    assistant_reply = response_json['choices'][0]['message']['
       content']

    return response_json


question = "I will give you a list of URLs. Can you infer the name
    of the company or university with these URLs?\n\n"
question += "URLs:\n"
for link in links:
    question += f"- {link}\n"
question += "\nPlease only return the names in a CSV format and no
    explanations."

response = chat_with_gpt3(question)
```

dataset gets augmented with a new field, indicating whether a match with a company or institution can be identified.

Scraping GitHub

The final scraping process targets GitHub, given that the names of projects or companies have previously been identified. Particularly, projects often maintain public repositories or what's known as 'organizations' on GitHub, which include a list of contributors. Consequently, our scraper performs a keyword search, pairing the derived name with GitHub. Once a repository is located, the scraper navigates to the 'contributors' section and retrieves a list of names.

This list serves as the final addition to the dataset of individuals and operates in a manner similar to previous checks. Each entry in the dataset is supplemented with a new field, indicating whether this individual is identified as a contributor to a GitHub repository or organization associated with the project or company name derived from the report.

5.4.3 Match Score Calculation

At this stage, the dataset has been compiled, as outlined in Listing 5.11. With the completion of the scraping process, the focus shifts to the calculation of a matching score. This score indicates the probability of an individual's participation in the event of the scenario. A total of seven parameters are considered for this computation: the presence and frequency of collaboration with the primary subject, the presence and frequency of collaboration with the key company during the year of the event, whether their current institution affiliation (as per Google Scholar) matches the key company or a partner institution involved in the event, the presence of their name in a GitHub repository linked to the event, and their contribution to a publication related to a project associated with the event.

Listing 5.11: Structure of the Final Dataset

```
[{
'Name': 'Full name of the person,
'Collaborations': Number of collaborations with the main subject,
'Institution': 'Name of the institution currently employed',
'ImageLink': 'Link to an image of the person,
'Iteration': 'Indicates the origin of this entry,
'MentionCount': 'number of mentions in the company report,
'InstitutionMatch': 'Name of matching institution',
'InstitutionMatchStatus': 'Boolean value to indicate a match',
'publication_ElectoSense': 'Boolean value to indicate a match',
'GitHub_Match': 'Boolean value to indicate a match',
}]
```

Upon making a request to `http://localhost:5001/full` from the front end and the successful completion of the scrapers, a dataset structured as per Listing 5.11 is returned. The `calculateMatchScores()` function 5.12 is responsible for calculating the "match score" for each individual in the dataset, based on various criteria. It uses cloned data from the original dataset to avoid direct modification.

Listing 5.12: Calculating Match Score

```
calculateMatchScores() {
      // Clone the original data so we don't modify it directly
      this.data = _cloneDeep(this.originalData);
      const maxCollab = Math.max(...this.data.map(d => d.
         Collaborations));
      const maxMention = Math.max(...this.data.map(d => d.
         MentionCount));

      this.data.forEach(d => {
        const collabExistsScore = (d.Collaborations > 0 ? 1 : 0) *
            this.collabExistsWeight;
        const collabCountScore = (d.Collaborations / maxCollab ||
           0) * this.collabCountWeight;
        const mentionExistsScore = (d.MentionCount > 0 ? 1 : 0) *
           this.mentionExistsWeight;
        const mentionCountScore = (d.MentionCount / maxMention ||
           0) * this.mentionCountWeight;
        const institutionMatchScore = (d.InstitutionMatchStatus ?
           1 : 0) * this.institutionMatchWeight;
        const githubMatchScore = (d.GitHub_Match ? 1 : 0) * this.
           githubMatchWeight;
        const publicationElectoSenseScore = (d.
           publication_ElectoSense ? 1 : 0) * this.
           publicationElectoSenseWeight;

        d.match_score = collabExistsScore + collabCountScore +
           mentionExistsScore + mentionCountScore +
           institutionMatchScore + githubMatchScore +
           publicationElectoSenseScore;
      });

      // Sort the data in descending order of match score
      this.data.sort((a, b) => b.match_score - a.match_score);
    },
```

The function initially determines the maximum number of collaborations and mentions in the dataset. These values serve for normalizing the collaboration and mention scores of each individual, ensuring the scores are proportionate and comparable. Following this, the function iterates over each individual in the data, calculating scores of the seven parameters mentioned earlier, each multiplied by a corresponding weight.

This front end calculation and storage design decision facilitates user modification of weights, allowing the recalculation of the match score based on user input, as depicted in 6.7. The user has the possibility to manipulate the weight and therefore the importance of each parameter and by pressing the recalculation button, the list is accordingly updated. Given that the data is stored on the front end and each computation does not manipulate the original data, this process is executed instantaneously, eliminating the need for the scrapers to repeat their operations. The computed results are presented beneath the sliders used for weight calculation, arranged in descending order of the match score. An illustrative depiction of how individuals are represented in this list can be found in Figure 6.7.

Chapter 6

Case Study

This chapter presents a complete walkthrough demonstrating the platform's intended use and visually presenting the capabilities enabled by the platform. In this participatory case study, the author of the platform uses the guiding platform to execute the pipeline proposed in this thesis. The resulting analysis and visual representation are developed based on a real-world case of a previous Hackathon with actual persons involved.

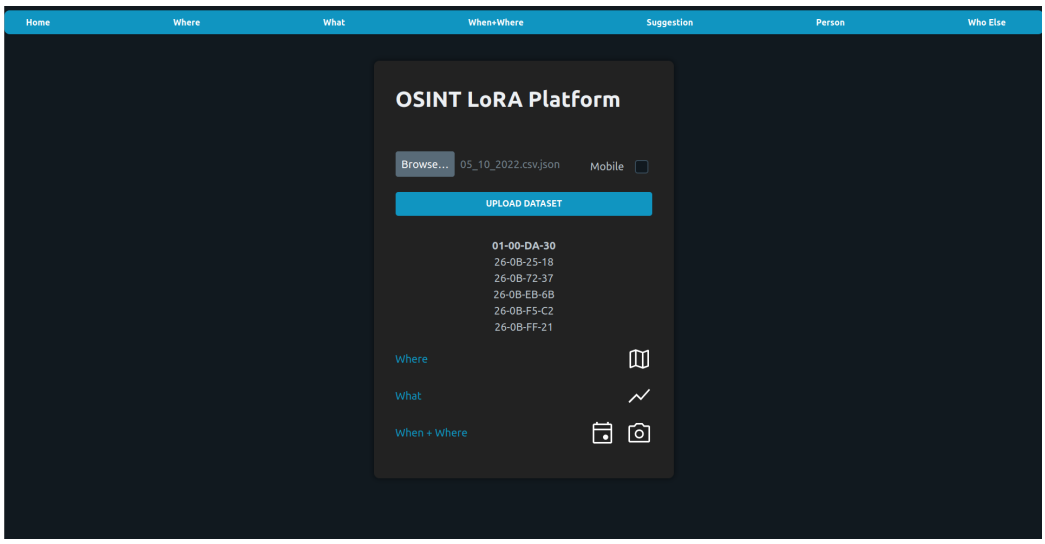


Figure 6.1: Successful Upload of the LoRa Dataset

Following the successful setup of the front end and three back ends as guided, users access the locally deployed web application via their browser. As the front end is built, they are welcomed by the user interface showcased in Figure 6.1, where they select and upload a LoRa dataset in JSON format. Once upload, the informally called "What" back end provides the front end with a list of devices, a list of days, alongside a traffic analysis. The specific dataset used here was exported from the *The Things Network* platform and contains LoRa traffic of multiple sensors. Upon dataset upload, users can pinpoint a specific device from a list located beneath the upload button. Once a device is chosen, so called "Where and When" back end is employed. This back end provides a map, displaying

the location of the LoRa device as illustrated in Figure 6.2. Additionally, this back end is responsible for inferring the social media posts, the weather data, and the results from the search engine Shodan.

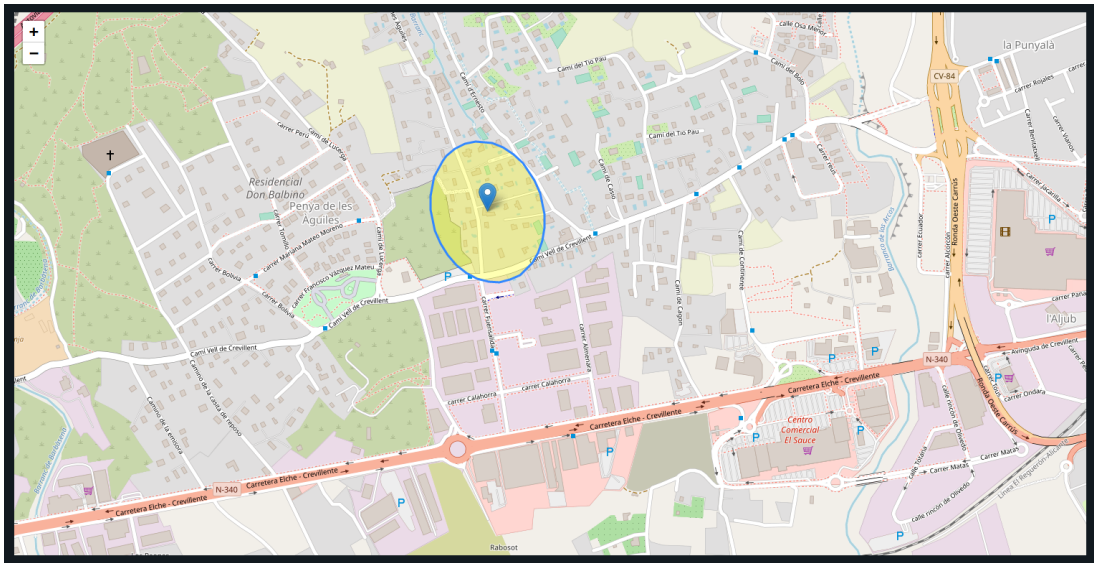


Figure 6.2: Visual Representation of LoRa Data Points Across Geographic Locations

Beneath the map lies a section presenting the overview of the retrieved data from the "What" back end. Here, users can select specific days and devices to review the traffic analysis. They are provided with an overview of hourly packet data, message types (*i.e.*, mainly uplink traffic in this example), and other relevant metadata upon selection. This information is key for the subsequent heuristics that are applied based on the traffic. For example, a device that mainly sends uplink traffic will be categorized differently as a device receiving downlink messages. A more detailed explanation of this feature can be found in Section 4.1.1 and is visually represented in Figure 6.3.

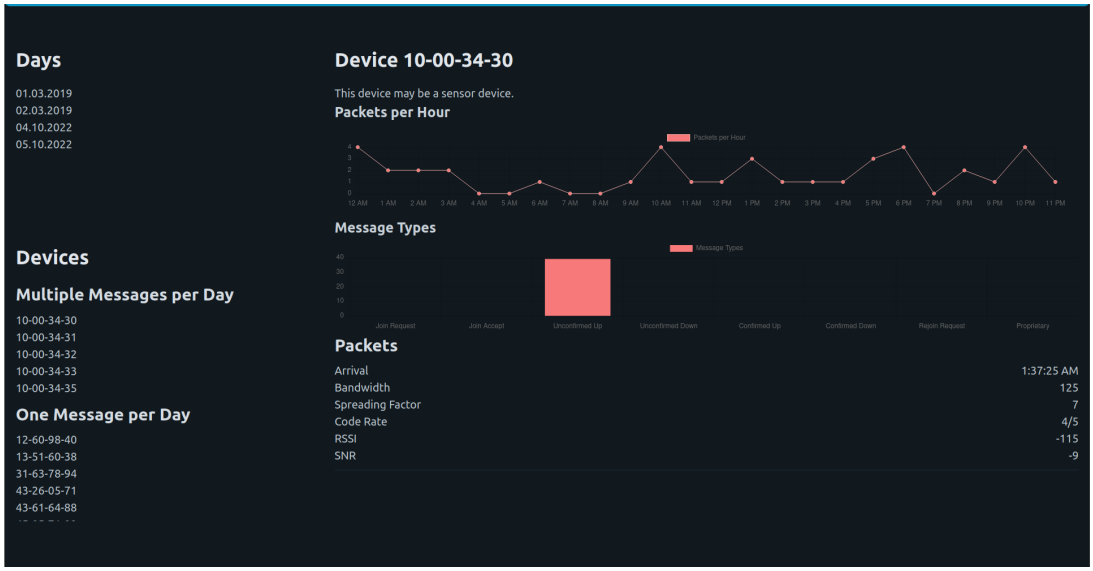


Figure 6.3: Visualizing Hourly Packet Traffic and Facilitating Device Selection

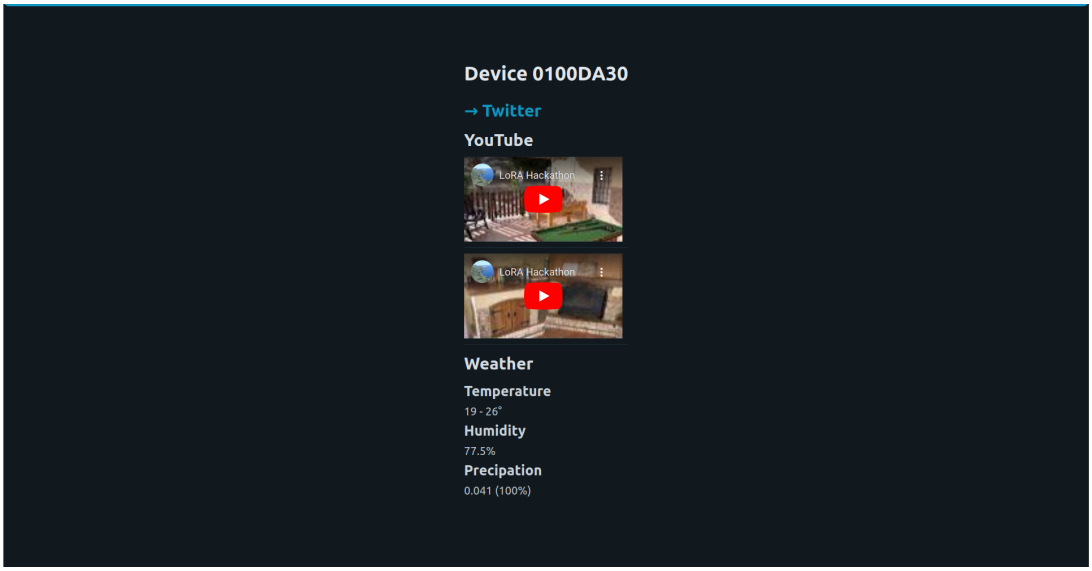


Figure 6.4: Scraped YouTube Videos, Links to Twitter Posts, and Weather Information

Based on the extracted location, the first information section presents video insights extracted from YouTube, a link to a specific Twitter search aligned with the location and timestamps. Additionally, pertinent weather data from the specified time of the LoRa data is displayed. This section is represented in Figure 6.4. Specifically, two related videos were associated with the previously extracted metadata, which already provides a first indication of persons that may potentially be involved.

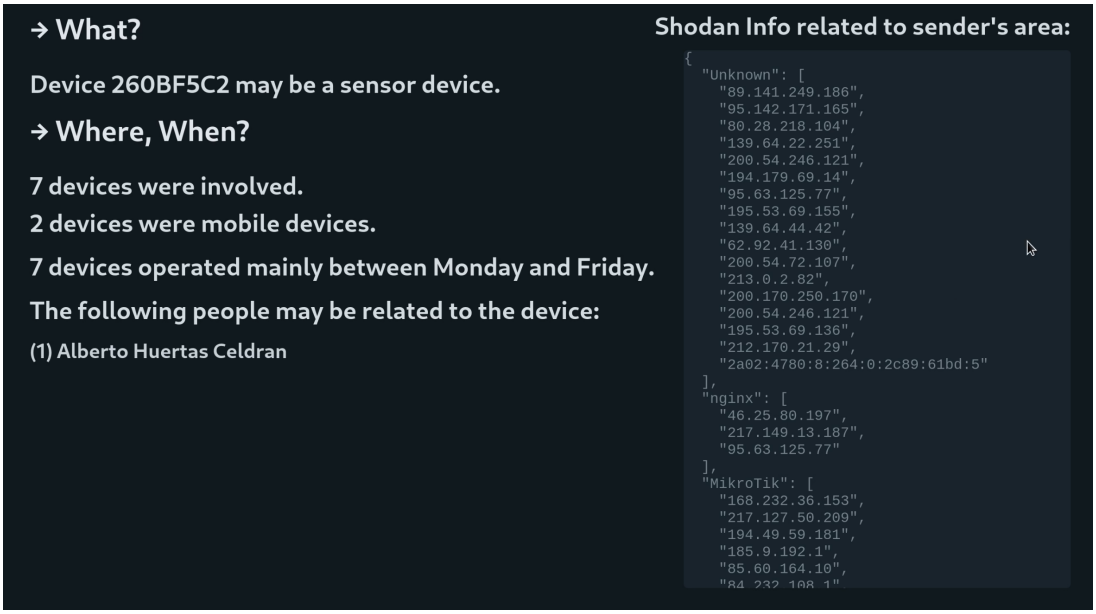


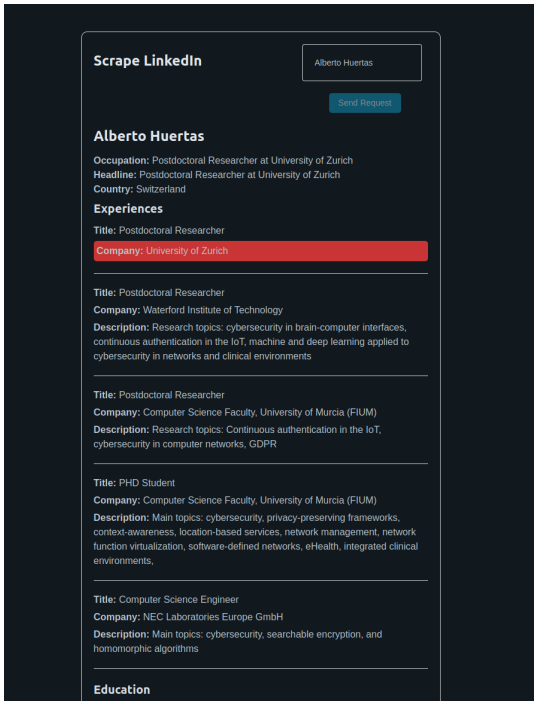
Figure 6.5: Summarized Suggestions

Subsequently, a section dedicated to suggestions is presented. It comprises the results from the Shodan search engine, detailing the involved devices. Furthermore, it includes a directory of individuals who may potentially be involved. People that are listed here are

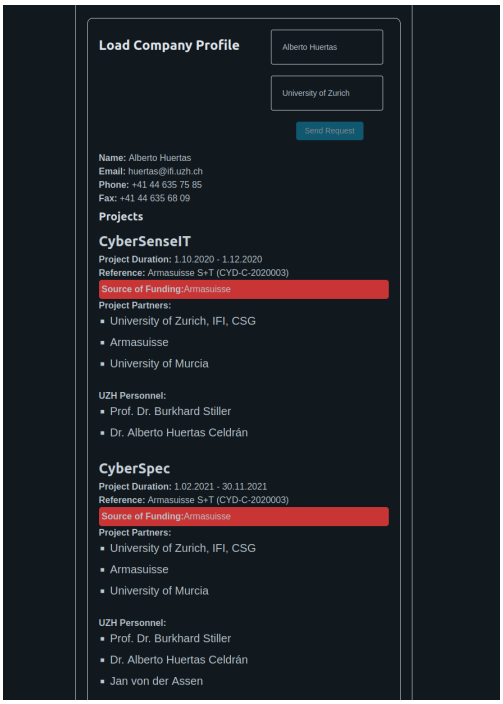
referred to as primary subjects and will serve as input parameters for scraping processes introduced by this thesis. As shown in Figure 6.5, in this scenario, 7 devices were identified, and, based on heuristics, two of them were classified as mobile devices. Furthermore, it is shown that the devices only operate on weekdays. Most importantly, *Alberto Huertas Celdrán*, the name of a person associated with the localized network traffic, is displayed in the user interface.

Progressing to the first key feature of this thesis, which is illustrated in Figure 6.6 (a), there is an input field that shows the primary person of interest as determined by the platform. Once a name populates this field, a request can be initiated to the new back end. As mentioned in sub-chapter 5.2.2, the scraper infers the URL of the specific LinkedIn profile and then collects all data represented from said site via the service of ProxyCurl. Usually this data contains information about the current employment, past employer, education, and publications. This varies depending on what a person has shared on his profile page. After the process completes, the front end presents all data sourced from LinkedIn, with critical points emphasized in red. In our case we are mainly interested where the primary subjects is currently employed.

Once the platform identifies the current employer, it automatically pre-populates the subsequent field with this information, thereby preparing the next data request. Following the same procedure, once the request is sent, the front end displays all the acquired data. This process is visible in Figure 6.6 (b). A successful scraping process retrieves additional information about the primary subject, such as email address, office address, phone number, current projects, and sources of funding.



(a) Scraped LinkedIn Data

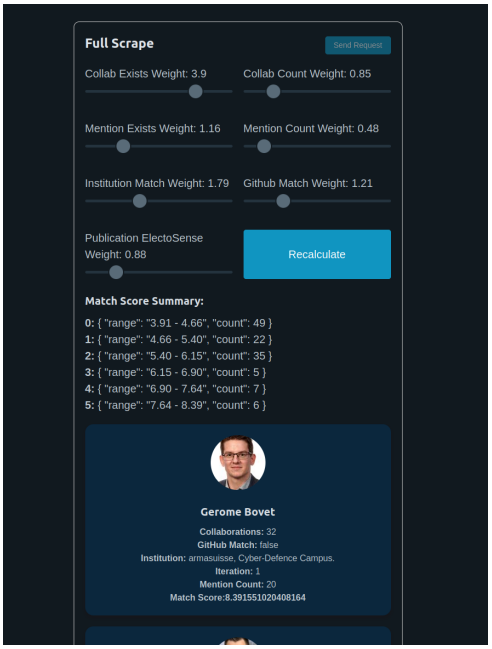


(b) Scraped Employees Company Page

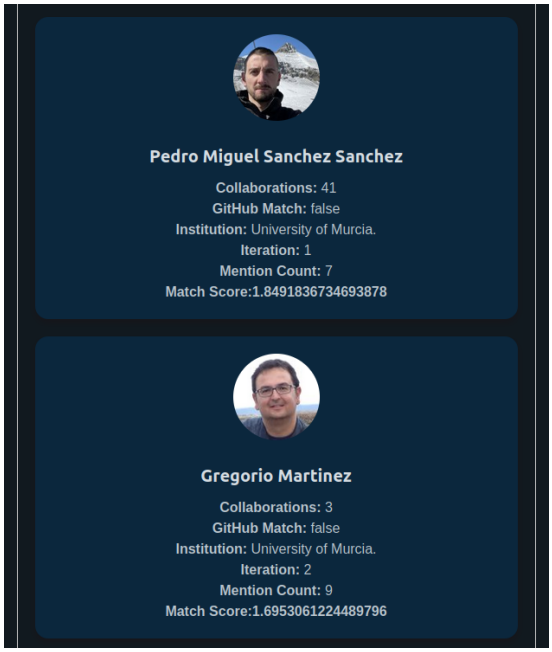
Figure 6.6: Scraping Results

We now transition to the final data scraping phase. As seen in Figure 6.7 (a), this is where the last data request is initiated. At this point, the user chooses to wait until the data is fetched before adjusting the weights, or he/she can preset the weights before making the request. Once the scraping process concludes, a list of all deduced individuals is displayed in descending order based on their match scores, as demonstrated in Figure 6.7 (b). As presented, for each individuals, the number of collaborations, affiliations, and other dimensions are rendered. Finally, the match score used for the ordering is also displayed to the user.

The match score can be recalculated by simply adjusting the sliders and clicking the 'recalculate' button, which instantly refreshes the list. Beneath the sliders, a brief summary of the match scores is provided. This summary reveals the range of calculated match scores based on your weight configuration. It also displays the population distribution for each range of match scores. For this, the match score is divided into six equally sized ranges.



(a) Recalculate the Match Score and Seeing the Resulting List Below



(b) Potential Participants Alongside their Respective Match Scores and Personal Information

Figure 6.7: Match Score Results

Chapter 7

Evaluation

In this chapter, a multi-dimensional evaluation of the approach's performance is undertaken, spotlighting its accuracy, speed, potential limitations, and resultant findings. The assessment is grounded on two primary metrics: the extent of coverage of actual event participants and the precision of the matching score calculations, which collectively offer insights into the effectiveness and accuracy of the data collection and matching algorithm.

Alongside this, the time efficiency of the scrapers is examined, and the model's limitations are delved into, stemming from the scraper design and the chosen scenario. This acknowledgment of potential constraints and bottlenecks assists in identifying areas for future refinement, paving the way for improvements.

The results of this work are presented and interpreted within the broader context of the project and its overarching objectives. This chapter emphasizes the real-world application of location-based LoRa data in scenarios with minimal online presence and explores new theoretical avenues in network analysis. The conclusion outlines the specific implications derived from this study in the context of applying Open-source intelligence (OSINT) to LoRa networks.

7.1 Performance Metrics

In Open-source intelligence projects, measuring performance and accuracy can be particularly challenging due to the uncertain nature of the data - it's often impossible to define the total amount of data that can be collected for a given scenario. However, in this project, the evaluation process is somewhat simplified due to a well-defined real-world scenario: custom-set posts on social media and the availability of a comprehensive list of all participants at the event, which constitutes the ground truth.

Given this scenario, the model's performance can be measured using two primary metrics. The first metric concerns the coverage of the actual participants in the event - in other words, how many of the real-life participants have been successfully identified and included in the collected dataset. This provides us with a quantitative measure of how

comprehensive the data collection methods presented in this thesis were in capturing the relevant individuals.

The second evaluation metric focuses on the precision of the matching score computations. Using the list of actual event participants as a reference, the scores given to these participants are compared with those assigned to individuals who didn't attend. This comparison allows for assessing the extent to which the score computation can accurately differentiate between attendees and non-attendees. Effectively, this precision measurement provides insights into the accuracy and predictive capacity of the matching score algorithm. A comprehensive understanding of the model's performance is obtained by examining these two metrics.

Next, efforts were dedicated to identifying an efficient weight allocation for the seven parameters in the model to optimize the precision metric. The investigation yielded the following optimal weights:

- `collab_exists_weight` = 3.9
- `collab_count_weight` = 0.85
- `mention_exists_weight` = 1.16
- `mention_count_weight` = 0.48
- `institution_match_weight` = 1.79
- `github_match_weight` = 1.21
- `publication_electosense_weight` = 0.88

These weights are primarily designed to minimize false positives. They are determined based on the known list of event attendees, hence, it reflects an ideal weight assignment for this specific context. In realistic settings where information about attendees is unknown, it's expected that the performance of the predictions will be reduced.

Three critical statistics are used in the analysis: precision, recall, and the F1 score. Precision refers to the ratio of true positives to all positive results—both true and false positives. Recall (or sensitivity) is the ratio of true positives to all actual positives, including both true and false negatives. The F1 score, on the other hand, measures test accuracy by considering both precision and recall. It can be seen as a weighted average of these two values, with the best score being 1 and the worst being 0. The F1 score offers a balanced view of model performance, especially in situations with uneven class distributions.

7.2 Results

Examining the first performance metric, the inclusion of participants, reveals interesting findings. Specifically, there were 16 individuals present at the armasuisse Hackathon organized in Alicante, Spain, from 4-5 October 2022. The proposed data collection method captured 14 out of these 16 participants in the generated dataset, indicating a high coverage rate.

Before evaluating the second performance metric, a cutoff point has to be established. This cutoff point sets the threshold within the matching score range above which a person is classified as present. To illustrate this concept, consider an example: if the matching scores range from 0 to 1 (with 1 being the highest possible score), and a cutoff point is set at 0.8, everyone with a score higher than 0.8 is classified as present at the event, while those below the threshold of 0.8 are categorized as not present. For this particular dataset and given weights from sub-chapter 7.1, cutoff values from 0.55 to 0.95 were explored to identify the optimal threshold. Results show that a cutoff point of 0.75% of the matching score range yields the highest precision and the best F1 score, as shown in Figure 7.2.

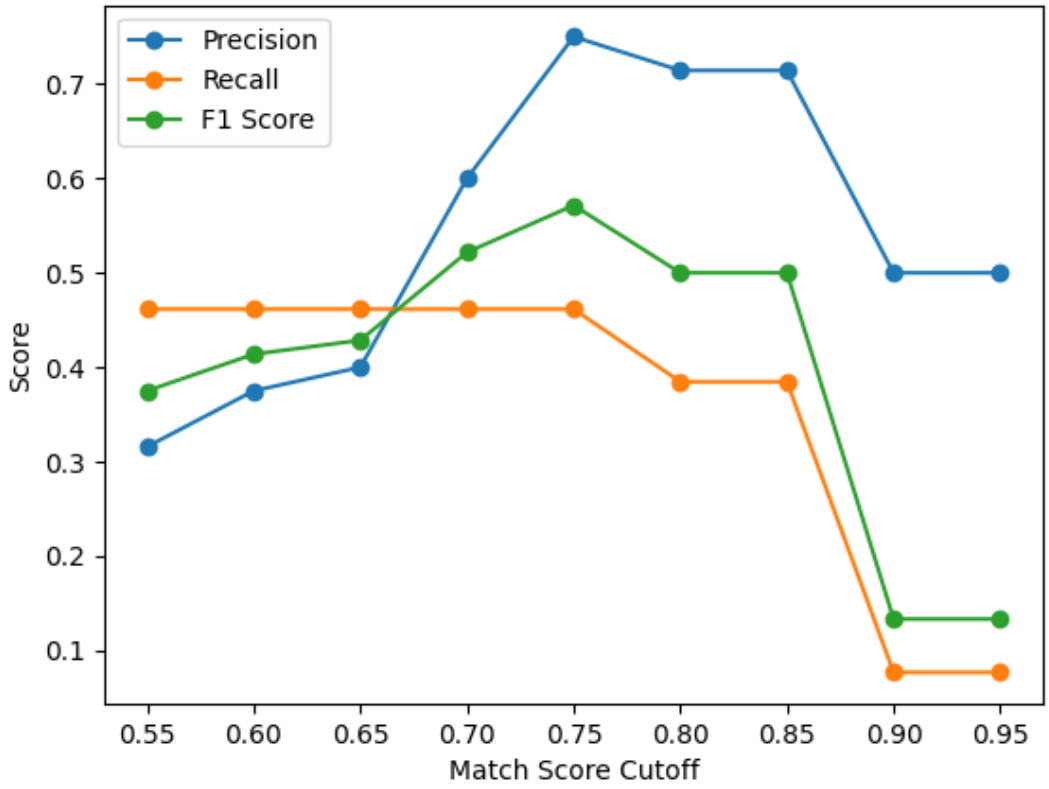


Figure 7.1: Precision, Recall, and F1 Score at Different Match Score Cutoffs

In the selected context, a match score cutoff point of 0.75% of the range proved to be the most effective. The corresponding performance metrics are illustrated in Figure 7.2. The approach proposed in this thesis achieved a precision of 0.75, indicative of a strong correlation between predictions and actual presence at the event. However, the recall score was relatively lower at 0.46, suggesting some true positives were not identified. The

balance between precision and recall, encapsulated in the F1 score, was measured at 0.57, reflecting the overall effectiveness of our method.

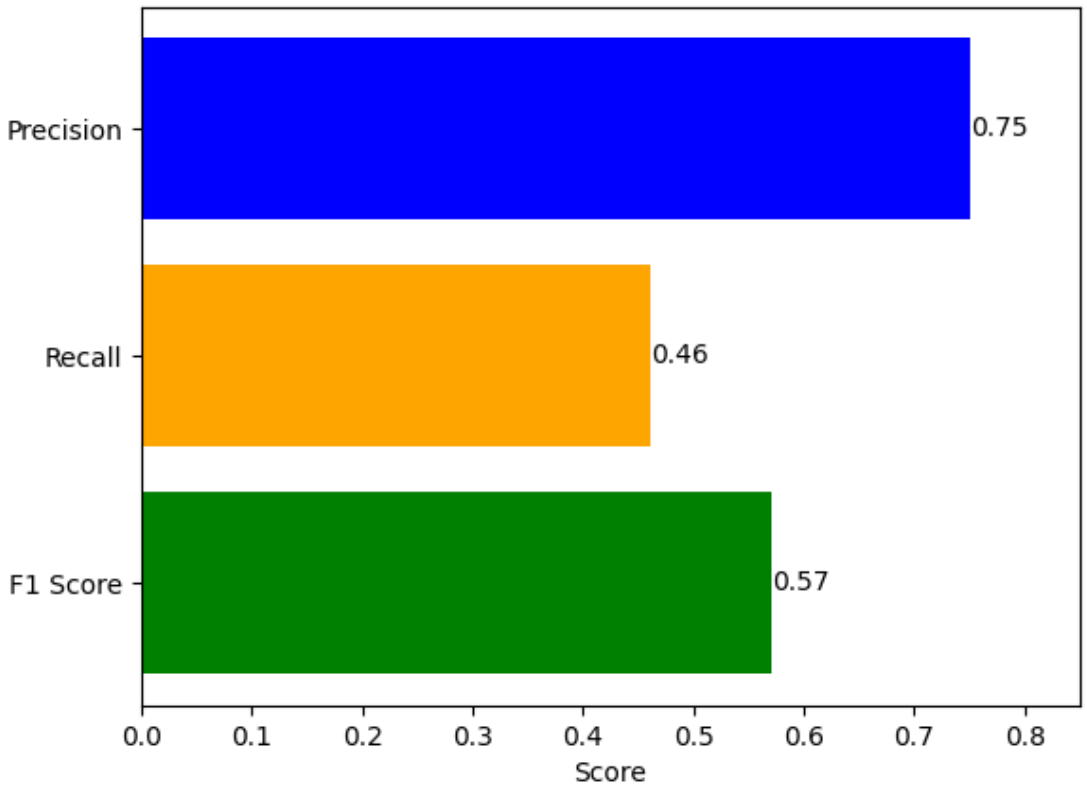


Figure 7.2: Metrics at Match Score Cutoff: 75%

7.3 Limitations

7.3.1 Scarpers

Utilizing scrapers in a replicable and scientific manner inherently poses challenges. Scrapers often rely on both the structure of the site and specific textual cues. Any update or modification by the site owner could disrupt the extraction of the desired information. Additionally, as bot detection mechanisms are continually enhanced, there is a possibility of encountering rate-limited server responses (*i.e.*, HTTP response code 429), which block the queries.

The tool's applicability varies across different scenarios due to the reliance on specific information extraction patterns. It is designed to locate key data points, such as an "employee information" section on a company's profile page or a specific company report through a keyword search. However, the scraping process will run into issues if these sections or documents are not found, which inevitably limits the application scope of this tool.

Moreover, the scraper's performance is also contingent on the availability and presentation of data on an individual's institutional affiliation pages. For example, extracting parameters like project funding might prove challenging if the institution lacks a profile page or presents data differently across companies.

7.3.2 GPT-3.5

Within this project, OpenAI's large language model, `GPT-3.5`, was utilized on three separate occasions. Using `GPT-3.5` comes with its advantages and disadvantages. On the one hand, it could solve complex issues like determining if an institution's name was in English or translating it otherwise, which could be conveniently handled with a single API query. On the other hand, `GPT-3.5` sometimes proved to be a vulnerable point in the automation process, as the responses occasionally included small talk or similar elements that made it challenging to extract essential keywords automatically for subsequent scrapers.

Regarding text classification, `GPT-3.5` showed itself to be superior to other classifiers. However, achieving consistently repeatable results was problematic, which is a crucial aspect of our tool.

7.3.3 Scenario

This research focuses on a unique scenario from a privately-organized Hackathon event. The distinctive nature of this event posed several challenges in terms of data availability. Given its private nature, the event had no internet visibility or publicity. Furthermore, attendees appeared highly reserved regarding the information they shared online. Apart from some custom-made posts, no mention or visual evidence of the event was found from any participant on any social media or open-access platform.

This lack of online data led to the absence of definitive parameters to confirm an individual's presence at the event. Contrastingly, in a situation where an event has a broader online presence and attendees are more open to sharing images or statuses indicating their participation, more accurate predictions may be achievable. In the presented case, a single participant was confirmed as a starting point for all inferences. However, if a scenario permitted the identification of multiple confirmed attendees, the prediction model would vastly improve precision, recall, and F1 score.

7.4 Interpretation and Discussion of the Results

In this section, the focus shifts to interpreting the results and their implications in relation to the project. It's crucial to note that this evaluation is anchored in a real-world scenario, featuring manually inserted posts, which possesses its own peculiarities and limitations. Therefore, the interpretations should be perceived in the light of this particular context.

Beginning with an examination of our primary performance metric (the extent of coverage of the actual event participants), our analysis revealed that our model successfully detected 14 out of the 16 individuals who were in attendance at the event. This high level of coverage underscores the efficacy of the data collection methods in pinpointing and incorporating the relevant individuals into our dataset. By being able to detect the majority of the actual participants, our model has demonstrated a significant capability to extract and assimilate key participant data, primarily from open-source platforms such as Google Scholar.

Next, attention is directed towards the second performance metric (the precision of the matching score computations). By using the list of actual event participants as a reference, we could determine how accurately the matching score algorithm could distinguish attendees from non-attendees.

Examining the precision, recall, and F1 score results, it is observed that the approach achieved a precision score of 0.75. This indicates a high degree of accuracy, with three out of four predictions being correct. However, the recall score was slightly lower at 0.46, suggesting that there were instances where individuals who attended the event were not identified. The balance between these two metrics, the F1 score, is crucial in providing a holistic view of the model's performance. An F1 score of 0.57, although not perfect, indicates a reasonable balance between precision and recall considering the complexity of the problem and the limitations of the available data.

The cutoff point of 0.75% of our matching scores range proved to be the most effective in terms of optimizing both precision and the F1 score. This selection underscores the importance of carefully calibrating the decision threshold in predictive modeling to achieve a suitable balance between minimizing false positives and maximizing true positives.

In terms of limitations, our model was largely constrained by the inherent challenges of web scraping and the use of the`GPT-3.5` model. While the former raised issues about the replicability of the process and the sensitivity to website structures, the latter introduced challenges with consistency and repeatability. Another key limitation stemmed from the scenario itself - the nature of the privately-organized event limited the amount of online data available for our analysis.

The achievements and limitations of our model are tied to the unique scenario in which it was developed and tested. While our model performed well in this particular context, it's important to note that different scenarios could present distinct challenges and opportunities which likely cause some steps of the scraping to fail.

In essence, the results demonstrate that our model possesses considerable potential in the field of OSINT, particularly for event participant data gathering and analysis. However, the limitations also highlight the need for ongoing development and refinement, particularly in relation to improving the recall metric, ensuring replicability, and adapting to different scenarios.

7.5 Time Efficiency Evaluation of Scrapers

Evaluating the time efficiency of scrapers can be quite challenging due to the numerous factors that affect this metric. Key elements such as CPU performance, network speed, and response time of external API providers significantly influence the time required for the scrapers to complete their job.

In the presented scenario, a stable home WiFi network with an average speed of 140 Mbps was used. The hardware used was a mobile laptop equipped with an Intel Core i7-8550U CPU, with 4 cores and a highest boost clock of 4 GHz. The system also included 16 GB of DDR4 memory running at 3200 MHz in a single-channel configuration.

To assess the average performance of the scraping procedures, requests were sent to three distinct endpoints: LinkedIn scraping, the employee's company page scraper, and the scrapers used to infer additional participants. The duration from the request being sent to the receipt of the response data was measured. This process was repeated five times for each endpoint, allowing for the calculation of the average time required for each scraping procedure 7.3.

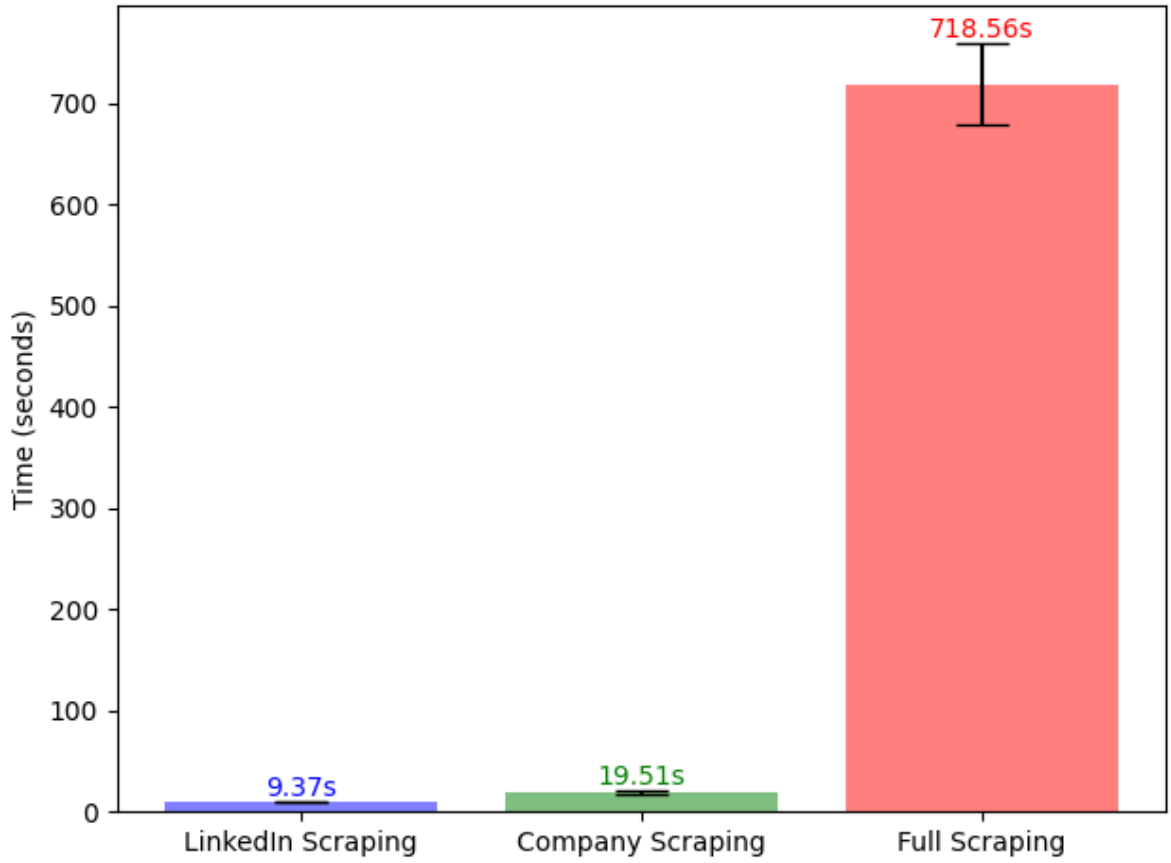


Figure 7.3: Average Time to Complete all Scraping Processes for the Three Main Endpoints

As depicted in Figure 7.3, the scraping processes for LinkedIn and the employees' company

pages are comparatively quick. However, the full scraping process, which involves inferring other participants, tends to be a much longer process, averaging around 12 minutes. The phase that collects additional information from Google Scholar about each participant is the most time-consuming, initially taking nearly 18 minutes before it was optimized using the multiprocessing library, as discussed in Chapter 5.4.1. This optimization significantly reduced the time required and prompted us to consider similar enhancements for other parts of the scraping processes.

An attempt was made to apply the multiprocessing approach to infer the Google Scholar page URLs for each individual and scrape all publications for a subject. However, challenges were encountered in both instances. Often, Google and Google Scholar may block the search requests, returning an HTTP 429 server response, indicating too many requests. In some cases, the user may be prompted to complete bot-detection tasks, such as selecting all images of a car, as an additional measure to limit automated activity.

Random waiting periods were experimented with in an effort to circumvent these issues, but the success rate showed to be too unpredictable for integration into the process. Consequently, there was no choice but to maintain other processes in an iterative manner.

7.5.1 System Utilization

Upon examining system utilization, it is clear that the performance is rarely bottlenecked due to high demands on system resources.

The LinkedIn scraping process demonstrated in Figure 7.4, highlights that the CPU and memory usage remain relatively steady in the initial 7 seconds, at around 20% and 35% respectively. It's important to note these percentages correspond to the specific hardware setup detailed in Section 7.5. Beyond this initial phase, from about the 7th to the 11th second, the system is predominantly idle, awaiting the response from the proxyCurl API.

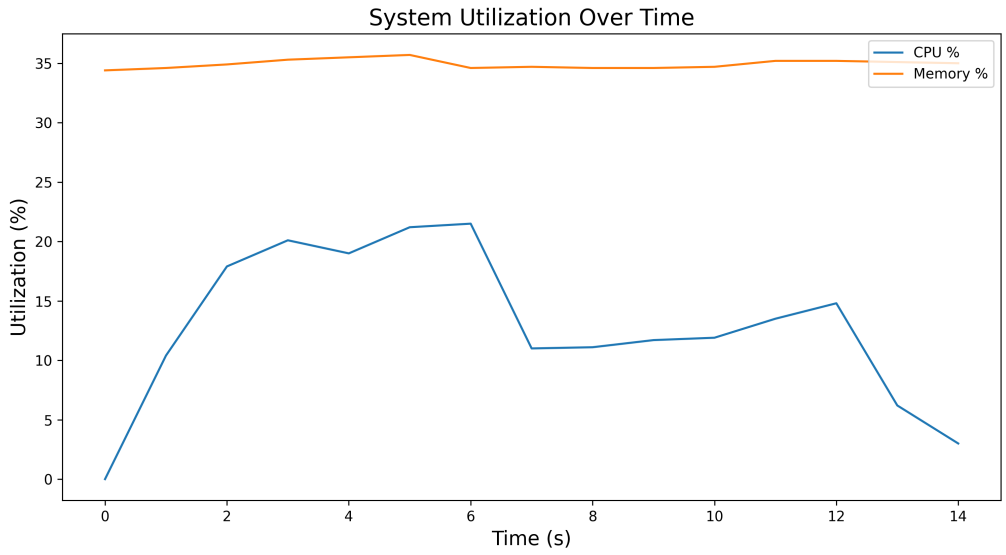


Figure 7.4: System Resource Utilization During LinkedIn Data Scraping Process

Turning our attention to the scraping of the company's employee pages shown in Figure 7.5, the CPU load is generally around 25%. The variations in CPU utilization can be attributed to the loading times of the requested web pages.

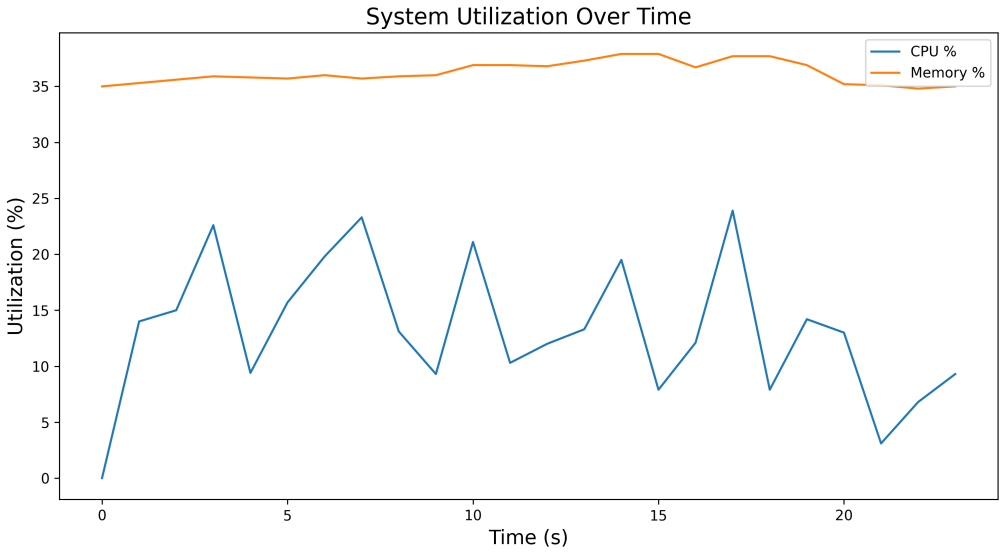


Figure 7.5: System Resource Utilization During Company Data Scraping Process

Figure 7.6 reveals the most notable observations, demonstrating the initial scraping iteration of the main subjects' co-authors, and the subsequent iteration that scrapes co-authors from all people of interest discovered in the initial iteration. In these two sections, the multiprocessing approach is employed as demonstrated in Listing 5.5. It can be observed that the CPU load averages around 78% and peaks at 97% load.

The time period from approximately the 400th to the 680th second consists primarily of requests made to OpenAI's large language model. The majority of this time is occupied by the institution normalization requests, as referenced in Listing 5.7.

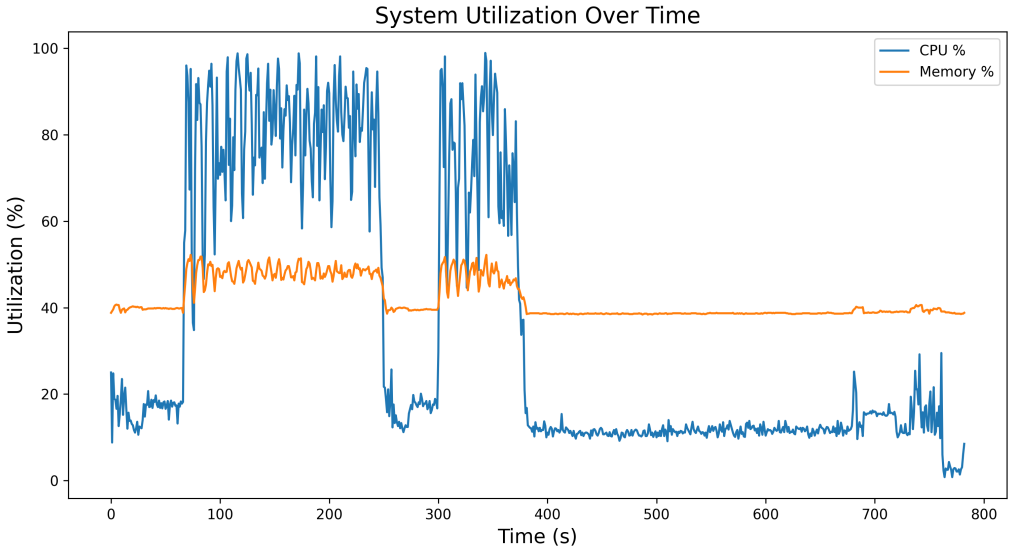


Figure 7.6: System Resource Utilization During the Scraping Process for Inferring Additional Attendees

The analysis of system utilization reveals that our scrapers operate effectively in most areas. However, one aspect that requires further optimization is the process of institution name normalization. During this phase, a significant number of iterative requests are sent to `GPT-3.5`, which affects overall efficiency. Although we did not experiment with parallelizing requests to `GPT-3.5` in this study, it is strongly recommended for future investigation.

7.6 Implications

The findings from our study have several implications, both practical and theoretical, for the implementation of OSINT in LoRa networks. On a practical level, we demonstrated the potential of location-based LoRa data in scenarios where there is minimal or no online presence of an event or the people involved.

The ability to make accurate predictions even under constrained circumstances signifies that LoRa data could be utilized more extensively for various purposes, from security measures to social network analyses. These results create a pathway for similar applications in various domains where information retrieval and processing play an important role.

On a theoretical level, the thesis contributes to the existing literature that seeks to understand the potential of LoRa networks for data scraping purposes. The ability to extract an entire network of individuals from the inferred presence of a single individual unveils new perspectives in network analysis and demonstrates the power of OSINT.

Based on the findings, a strong recommendation is made for the integration of OSINT techniques into LoRa networks. The comprehensive insights gained through this approach prove to be of high value. Not only could this enhance the efficiency of data scraping, but it could also facilitate the understanding and mapping of complex social networks.

Chapter 8

Summary and Conclusions

This thesis has presented an innovative exploration into the realm of Open-source intelligence (OSINT). The focus was not only on integrating OSINT with LoRaWAN to enhance information and analysis of event participants but also to infer as much information as possible about the main subject involved. The objective was to enhance the existing platform and extend its capabilities by incorporating data from LinkedIn, employee company profile pages, and various other sources to create a more comprehensive and accurate representation of the event scenario.

Our model successfully included a significant majority of actual event participants in the gathered dataset, providing a high coverage rate and demonstrating the efficiency of our data collection methods. It was further able to differentiate between attendees and non-attendees with a reasonable degree of accuracy, as indicated by a precision score of 0.75. The F1 score, reflecting the balance between precision and recall, stood at 0.57, a respectable figure given the complexity of the task and the limitations of available data.

The contribution of this research lies not only in the results achieved but also in the methodological advancement it represents in the OSINT field. This project has provided a demonstration of how OSINT, in combination with LoRaWAN, can yield valuable insights. It also showcases the potential for further research and development in this field.

Despite the promising outcomes, the research faced limitations stemming primarily from the nature of web scraping, the use of the large language model `GPT-3.5`, and the specific event scenario itself. These constraints posed challenges related to the replicability of the process, sensitivity to website structures, consistency, and repeatability, as well as the limited online visibility of the private event. Each of these limitations provides a pathway for future work, suggesting areas for refinement and improvement in subsequent iterations of the model.

Moving forward, it is evident that additional refinement and improvement are required to mitigate the model's limitations. There is potential for refining the recall metric, ensuring replicability, and adapting the model to different scenarios. Additionally, future research should aim to develop robust methodologies that can account for website updates or modifications and advances in bot detection mechanisms.

In conclusion, this research adds to the existing body of knowledge by applying OSINT within the context of LoRaWAN. It serves as a starting point for future exploration and broadens the scope for continued progress in this promising field. The employment of Open-source intelligence unveils a multitude of pathways to explore, catalyzing potential advancements in data collection and analysis. Such developments could facilitate fresh insights and progress across various applications.

Chapter 9

Future Work

Future research should consider several possible improvements and extensions to the work presented in this thesis. These recommendations are geared towards enhancing the overall robustness and applicability of our OSINT tool while mitigating the identified limitations.

9.1 Upgrading the Language Model

As part of the project, OpenAI's `GPT-3.5` was utilized for the task of translating institutions to English, inferring company names from website links, and extracting project or company names from collected text passages. While the language model demonstrated impressive capabilities, there were issues with repeatability and consistency in the responses. Future work could explore upgrading to a more advanced version of the `GPT` series, assuming that OpenAI continues its trajectory of releasing increasingly sophisticated models.

A more evolved version of `GPT` could potentially provide more consistent and context-aware responses, thereby improving the reliability and precision of the data extraction process. It's also worth considering other language models as they become available or even experimenting with training a custom language model specifically for the task at hand.

9.2 Enhancing Scraper Versatility

The developed scrapers are highly reliant on the structure of the web pages they extract data from. This dependency introduces challenges in terms of their robustness and versatility, as any changes to the structure of a website can potentially disrupt the scraper's functionality. Consequently, it is recommended to explore techniques that can enhance the resilience of the scrapers to such changes.

Machine learning could potentially be utilized to improve scraper flexibility. For instance, the use of more advanced techniques, such as computer vision or reinforcement learning, might facilitate the development of scrapers capable of adapting to changes in website structures. With the rapid pace of advancements in these fields, this approach may be feasible and beneficial in the near future.

9.3 Robustness Across Scenarios

Another aspect of our work that requires attention is the model's generalizability to other scenarios. Our project was designed and evaluated in a very specific context, which might not match other real-world situations. To be truly useful, our tool should be able to handle a wider range of scenarios and deliver consistent results.

Addressing this issue could involve refining the weight allocation process for the matching score parameters based on different scenarios. Another approach could be the development of machine learning models capable of learning optimal weight distributions from past event data. Such improvements could potentially enhance the tool's adaptability and accuracy across a broader range of use cases.

9.4 Expanding Social Media Coverage

Our project focused primarily on Google Scholar and LinkedIn for information extraction. However, there are numerous other social media platforms where valuable data about event participants could be found. These platforms include Facebook, Twitter, Instagram, and others, each hosting a wealth of user-generated content that could provide insights into event participation.

Future work should consider extending the capabilities of the tool to include these platforms. While our scenario limited the utility of these platforms, their inclusion would likely enhance the tool's performance in scenarios where participants are more active on these platforms.

Bibliography

[1] J. Pastor-Galindo, P. Nespoli, F. Gómez Mármol, and G. Martínez Pérez, "The not yet exploited goldmine of OSINT: Opportunities, open challenges and future trends", *IEEE Access*, vol. 8, pp. 10 282–10 304, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2965257.

[2] S. Shah, R. Rutherford, and S. Menon, "Emerging Technologies of IoT Usage in Global Logistics", in *2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, Jan. 2020, pp. 251–257. DOI: 10.1109/ICCAKM46823.2020.9051530.

[3] A. Rejeb, K. Rejeb, H. Treiblmaier, *et al.*, "The Internet of Things (IoT) in healthcare: Taking stock and moving forward", en, *Internet of Things*, vol. 22, p. 100 721, Jul. 2023, ISSN: 2542-6605. DOI: 10.1016/j.iot.2023.100721. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660523000446 (visited on 07/02/2023).

[4] "Internet of Things (IoT) in Agriculture - Selected Aspects", eng, *AGRIS on-line Papers in Economics and Informatics*, 8th ser., M. Stočes, J. Vaněk, J. Masner, and J. Pavlík, Eds., 2016, Num Pages: 6. DOI: 10.22004/ag.econ.233969.

[5] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia, and T. Watteyne, "Understanding the limits of LoRaWAN", *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017, ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600613. arXiv: 1607.08011[cs]. [Online]. Available: http://arxiv.org/abs/1607.08011 (visited on 02/12/2023).

[6] M. Glassman and M. J. Kang, "Intelligence in the internet age: The emergence and evolution of Open Source Intelligence (OSINT)", en, *Computers in Human Behavior*, vol. 28, no. 2, pp. 673–682, Mar. 2012, ISSN: 0747-5632. DOI: 10.1016/j.chb.2011.11.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563211002585 (visited on 02/06/2023).

[7] F. Schaurer and J. Störger, "The evolution of open source intelligence", ETH Zurich, Oct. 2010, Artwork Size: 10 p. Medium: application/pdf, 10 p. DOI: 10.3929/ETHZ-A-006251404. [Online]. Available: http://hdl.handle.net/20.500.11850/25221 (visited on 02/14/2023).

[8] A. Jaiswal, W. Peng, and T. Sun, "Predicting time-sensitive user locations from social media", in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '13, New York, NY, USA: Association for Computing Machinery, Aug. 25, 2013, pp. 870–877, ISBN: 978-1-4503-2240-9. DOI: 10.1145/2492517.2500229. [Online]. Available: https://doi.org/10.1145/2492517.2500229 (visited on 02/17/2023).

[9] R. Sanchez-Iborra, J. Sanchez-Gomez, J. Ballesta-Viñas, M.-D. Cano, and A. F. Skarmeta, "Performance evaluation of LoRa considering scenario conditions", *Sensors (Basel, Switzerland)*, vol. 18, no. 3, p. 772, Mar. 3, 2018, ISSN: 1424-8220. DOI: 10.3390/s18030772. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5876541/ (visited on 02/12/2023).

[10] J. R. G. Evangelista, R. J. Sassi, M. Romero, and D. Napolitano, "Systematic literature review to investigate the application of open source intelligence (OSINT) with artificial intelligence", *Journal of Applied Security Research*, vol. 16, no. 3, pp. 345–369, Jul. 3, 2021, Publisher: Routledge _eprint: https://doi.org/10.1080/19361610.2020.1761737, ISSN: 1936-1610. DOI: 10.1080/19361610.2020.1761737. [Online]. Available: https://doi.org/10.1080/19361610.2020.1761737 (visited on 02/19/2023).

[11] Gritzalis, Kandias, Stavrou, and Mitrou, "History of information : The case of privacy and security in social media", 2014. [Online]. Available: https://www.semanticscholar.org/paper/History-of-Information-%3A-The-case-of-Privacy-and-in-Gritzalis-Kandias/f22142c17182e53e03bd3391bae5b6cadafce591 (visited on 02/24/2023).

[12] H. Pellet, S. Shiaeles, and S. Stavrou, "Localising social network users and profiling their movement", *Computers & Security*, vol. 81, pp. 49–57, Mar. 1, 2019, ISSN: 0167-4048. DOI: 10.1016/j.cose.2018.10.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818301524 (visited on 02/01/2023).

[13] A. Ahmed, L. Hong, and A. J. Smola, "Hierarchical geographical modeling of user locations from social media posts", in *Proceedings of the 22nd international conference on World Wide Web*, ser. WWW '13, New York, NY, USA: Association for Computing Machinery, May 13, 2013, pp. 25–36, ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488392. [Online]. Available: https://doi.org/10.1145/2488388.2488392 (visited on 02/17/2023).

[14] J. Mahmud, J. Nichols, and C. Drews, "Where is this tweet from? inferring home locations of twitter users", *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 6, no. 1, pp. 511–514, 2012, Number: 1, ISSN: 2334-0770. DOI: 10.1609/icwsm.v6i1.14299. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14299 (visited on 02/17/2023).

[15] J. McGee, J. Caverlee, and Z. Cheng, "Location prediction in social media based on tie strength", in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ser. CIKM '13, New York, NY, USA: Association for Computing Machinery, Oct. 27, 2013, pp. 459–468, ISBN: 978-1-4503-2263-8. DOI: 10.1145/2505515.2505544. [Online]. Available: https://doi.org/10.1145/2505515.2505544 (visited on 02/19/2023).

[16] L. Backstrom, E. Sun, and C. Marlow, *Find me if you can: Improving geographical prediction with social and spatial proximity*. Apr. 26, 2010, 61 pp., Journal Abbreviation: Proceedings of the 19th International Conference on World Wide Web, WWW '10 Pages: 70 Publication Title: Proceedings of the 19th International Conference on World Wide Web, WWW '10. DOI: 10.1145/1772690.1772698.

[17] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring user routes and locations using zero-permission mobile sensors", in *2016 IEEE Symposium on Security and Privacy (SP)*, ISSN: 2375-1207, May 2016, pp. 397–413. DOI: 10.1109/SP.2016.31.

[18] V. Kanakaris, K. Tzovelekis, and D. Bandekas, "Geo-location twitter and instagram based on OSINT techniques: A case study", *International Journal of Advanced Research*, vol. 6, Feb. 6, 2018. DOI: 10.21474/IJAR01/6283.

[19] M. Kandias, V. Stavrou, N. Bozovic, L. Mitrou, and D. Gritzalis, "Can we trust this user? predicting insider's attitude via YouTube usage profiling", in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Dec. 2013, pp. 347–354. DOI: 10.1109/UIC-ATC.2013.12.

[20] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior", *Proceedings of the National Academy of Sciences*, vol. 110, no. 15, pp. 5802–5805, Apr. 9, 2013, Publisher: Proceedings of the National Academy of Sciences. DOI: 10.1073/pnas.1218772110. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1218772110 (visited on 02/23/2023).

[21] M. Kandias, D. Gritzalis, V. Stavrou, and K. Nikoloulis, "Stress level detection via OSN usage pattern and chronicity analysis: An OSINT threat intelligence module", *Computers & Security*, Security Data Science and Cyber Threat Management, vol. 69, pp. 3–17, Aug. 1, 2017, ISSN: 0167-4048. DOI: 10.1016/j.cose.2016.12.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404816301742 (visited on 02/06/2023).

[22] D. R. Hayes and F. Cappa, "Open-source intelligence for risk assessment", *Business Horizons*, vol. 61, no. 5, pp. 689–697, Sep. 1, 2018, ISSN: 0007-6813. DOI: 10.1016/j.bushor.2018.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007681318300296 (visited on 02/06/2023).

[23] M. Kandias, L. Mitrou, V. Stavrou, and D. Gritzalis, "Which side are you on? a new panopticon vs. privacy", in *2013 International Conference on Security and Cryptography (SECRYPT)*, Jul. 2013, pp. 1–13.

[24] H. Li, F. Guo, W. Zhang, J. Wang, and J. Xing, "(a,k)-anonymous scheme for privacy-preserving data collection in IoT-based healthcare services systems", *Journal of Medical Systems*, vol. 42, no. 3, p. 56, Feb. 14, 2018, ISSN: 1573-689X. DOI: 10.1007/s10916-018-0896-7. [Online]. Available: https://doi.org/10.1007/s10916-018-0896-7 (visited on 03/09/2023).

[25] Y. A. Mahmeed, W. Elmedany, and M. S. Sharif, "Eagle-eye: Open-source intelligence tool for IoT devices detection", in *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, ISSN: 2770-7466, Nov. 2022, pp. 526–530. DOI: 10.1109/3ICT56508.2022.9990658.

[26] A. Daskevics and A. Nikiforova, "ShoBeVODSDT: Shodan and binary edge based vulnerable open data sources detection tool or what internet of things search engines know about you", in *2021 Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, Nov. 2021, pp. 38–45. DOI: 10.1109/IDSTA53674.2021.9660818.

[27] A. Nikiforova, A. Daskevics, and O. Azeroual, "NoSQL security: Can my data-driven decision-making be influenced from outside?", in Jan. 30, 2023, pp. 59–73. DOI: 10.1108/978-1-80382-551-920231005. arXiv: 2206.11787[cs]. [Online]. Available: http://arxiv.org/abs/2206.11787 (visited on 03/09/2023).

[28] F. Rezaeibagha, Y. Mu, X. Huang, W. Yang, and K. Huang, "Fully secure lightweight certificateless signature scheme for IIoT", *IEEE Access*, vol. 7, pp. 144 433–144 443, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2944631.

[29] Cyber-Tracer, *Lora-hackathon*, 2022. [Online]. Available: https://github.com/Cyber-Tracer/lora-hackathon.

[30] *Sede electrónica del catastro*, Accessed: July 3, 2023, 2023. [Online]. Available: https://www1.sedecatastro.gob.es/OVCInicio.aspx.

[31] Shodan, *Shodan*, [Accessed on: June 22, 2023], 2023. [Online]. Available: https://www.shodan.io/.

[32] Vue.js, *Vue.js*, [Accessed on: June 22, 2023], 2023. [Online]. Available: https://vuejs.org/.

[33] The Things Network, *The Things Network*, [Accessed on: June 23, 2023], 2023. [Online]. Available: https://www.thethingsnetwork.org/.

[34] J. Contreras, *Lora-hackathon*, 2023. [Online]. Available: https://github.com/JonathanContrerasM/lora-hackathon.

[35] J. Contreras, *Osint-lora-scraper*, 2023. [Online]. Available: https://github.com/JonathanContrerasM/OSINT-LoRA-Scraper.

[36] joeyism, *Linkedin_scraper*, 2017. [Online]. Available: https://github.com/joeyism/linkedin_scraper.

[37] Nubela, *Proxycurl*, [Accessed on: June 12, 2023], 2023. [Online]. Available: https://nubela.co/proxycurl/.

[38] Explosion, *Spacy: Industrial-strength natural language processing*, [Accessed on: June 15, 2023], 2023. [Online]. Available: https://spacy.io/.

Abbreviations

API	Applications Programming Interface
CSV	comma-separated value
CSS	Chirp Spread Spectrum
GPS	Global Positioning System
IIoT	Industrial Internet of Things
IoT	Internet of Things
LPWAN	Low-Power-Wide-Area-Network
ML	Machine Learning
OSINT	Open Source Intelligence
OSN	Online Social Networks
nCRF	Nested Chinese Restaurant Franchise
SNS	Social Network Sites

List of Figures

List of Tables

Listings

Chapter 10

Installation Guidelines

The Scraper tool is a Python and Flask-based web scraper that was developed and tested on a Linux operating system. To set up the environment for running this tool, the following software needs to be installed:

1. **Python:** The Scraper tool is a Python-based application developed using Python 3.10.6. Python can be downloaded from the official Python website.
2. **Chromium:** The application uses Selenium WebDriver along with the Chromium browser. The Scraper tool was tested with Chromium version 114.0.5735.106, which can be downloaded from the official Chromium website.
3. **ChromeDriver:** This is a WebDriver for Chromium and is necessary for the Scraper tool to interact with the Chromium browser. The tool was tested with ChromeDriver version 113.0.5672.63. ChromeDriver can be downloaded from the ChromeDriver downloads page.

After the necessary software is installed, the environment needs to be set up as follows:

1. **Add ChromeDriver to your PATH:** After downloading ChromeDriver, it needs to be added to the system's PATH. This can be done by extracting the downloaded ChromeDriver archive to obtain the 'chromedriver' executable. This executable can then be moved to a directory like '/usr/local/bin', which is included in the system's PATH by default on most Linux systems. This can be done with a command in the terminal, replacing '/path/to/chromedriver' with the path where the 'chromedriver' executable was extracted. Once moved, the 'chromedriver' can be confirmed to be in the PATH by opening a new terminal window and running the 'chromedriver –version' command, which should display the version number if the 'chromedriver' was correctly added to the PATH.
2. **Install Python Dependencies:** The Scraper tool also depends on several Python packages, which can be installed using Pip. These dependencies can be installed by navigating to the 'Scraper/src' directory in the terminal and running the 'pip install -r requirements.txt' command.

10.1 Setting Up API Keys

This project uses the Proxycurl API and the OpenAI API. To obtain your API keys, accounts must be created on these platforms.

1. Create an account with Proxycurl at https://nubela.co/proxycurl. After signing up, your Proxycurl API key will be available in your dashboard.
2. Sign up for an account with OpenAI at https://openai.com/blog/openai-api. After signing up, your OpenAI API key can be found in the API section.
3. Once you have your API keys, they need to be added to a ‘.env‘ file in the root directory of the project. This file is used for securely storing sensitive information such as API keys. If the ‘.env‘ file does not exist, create it.
4. Open the ‘.env‘ file in a text editor and include the following lines:

   ```
   PROXYCURL_API_KEY='<your-proxycurl-api-key>'
   OPEN_AI_API_KEY='<your-openai-api-key>'
   ```

   Replace ‘<your-proxycurl-api-key>‘ and ‘<your-openai-api-key>‘ with your actual API keys.
5. Save and close the ‘.env‘ file. Your keys are now ready to be used by your application.

Important: Never share your API keys or commit them to your source code repository. The ‘.env‘ file is typically added to the ‘.gitignore‘ file to prevent it from being tracked by Git.

After the environment is set up, the API keys are configured, and the necessary dependencies are installed, the Flask server can be started by running the 'main.py' script located in the 'Scraper/src' directory with the command 'python main.py'. After running this command, the Flask server should start, and the Scraper tool will begin operating.