



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Vilnius
University

Giovanni Menon

Telegram Community Intelligence: A Multi-Layer Framework for CTI Analysis

Master Dissertation

Master's in International Cybersecurity and Cyberintelligence

Work developed under the guidance of:

Alberto Casares Andrés

Gabriel Maciá Fernández

June 2026

Abstract

Telegram has become one of the main platforms for cybercriminals and hacktivist collectives that use it to coordinate attack campaigns, spread disinformation and share technical artifacts. However, existing approaches to Telegram threat intelligence focus on individual channels or cases, providing limited insights into how threat communities are organized, how content moves across communities, and how malicious indicators are produced and distributed across the ecosystem. This dissertation proposes the **Telegram Communities Intelligence Framework (TGIF)**, a multi-layer analytical framework for Telegram *Cyber Threat Intelligence* (CTI) that integrates user co-affiliation community detection, community-level topic modeling, forwarding and content-hash propagation analysis, and *Indicator of Compromise* (IoC) extraction and verification to analyze user community structure, information flow, and distribution of IoC artifacts. We analyze a corpus of 45.7 million Telegram messages collected over a window of fourteen months. The results reveal a robust community structure that is not reducible to language, topic, or activity timing. The information-flow analysis further shows that forwarding and content-hash propagation are strongly community-bounded, while different communities exhibit distinct forwarding behaviors. Finally, the indicator layer identifies a subset of communities and topics in which the production and distribution of malicious indicators is concentrated, while temporal analysis shows that part of the malicious-indicator space appears in the corpus before it is represented in external reputation data. Therefore, these results demonstrate the value of a multi-layer analytical pipeline that enables cross-layer findings that single-channel or single-method approaches cannot capture.

Index

1	Introduction	1
2	Literature Review	3
2.1	Telegram	3
2.1.1	Telegram's Role in Hactivist and Cybercriminal Ecosystems	3
2.1.2	Indicators of Compromise on Telegram	4
2.2	Community Detection in Social Networks	4
2.2.1	Community Detection Algorithms	5
2.3	Topic Modeling for Threat Intelligence	7
2.3.1	Topic Modeling Approaches	7
2.4	Information Flow and Diffusion	8
2.5	Existing Telegram Intelligence Frameworks and Research Gap	8
3	Methodology and Implementation	10
3.1	Framework Overview	10
3.2	Methodological Setup	11
3.2.1	Environment Setup	11
3.2.2	Evaluation Measures	11
3.3	Dataset & Preprocessing	13
3.3.1	Dataset Description	13
3.3.2	Preprocessing Pipeline	14
3.4	Methods for RQ1: Community Structure and Topical Profiling	19
3.4.1	User Co-Affiliation Graph Construction	19
3.4.2	Topic Modeling	27
3.4.3	Community Profiling	33
3.4.4	Alternative attribute-based partitions	34
3.5	Methods for RQ2: Information Flow and Broker Detection	34
3.5.1	Forward Graph and Source Attribution	35
3.5.2	Forwarding Ecosystem Analysis	36
3.5.3	Topic-Driven Sourcing	37
3.5.4	Content-Hash Propagation	37
3.6	Methods for RQ3: IoC Extraction and Community Patterns	38
3.6.1	IoC Distribution Analysis	38
3.6.2	Temporal Analysis	40

4	Results & Discussion	42
4.1	Dataset Overview and Preprocessing Outcomes	42
4.1.1	Corpus Size and Temporal Coverage	42
4.1.2	Language Distribution	43
4.1.3	BERTopic Preprocessing Outcomes	44
4.2	RQ1: Community Structure and Topical Profiling	44
4.2.1	Structural Communities	45
4.2.2	Topic Space	47
4.2.3	Community Profiles	49
4.2.4	Alternative Attribute-Based Partitions	53
4.2.5	RQ1 Answer	53
4.3	RQ2: Information Flow and Broker Detection	54
4.3.1	Forwarding Structure and Source Ecosystem	54
4.3.2	Recurrent Feeders	56
4.3.3	Topic-Driven Sourcing Across Community	57
4.3.4	Content-Hash Propagation	58
4.3.5	RQ2 Answer	60
4.4	RQ3: IoC Patterns	60
4.4.1	IoC Extraction and Verification Summary	61
4.4.2	Distribution Across Communities	62
4.4.3	Distribution Across Topics	63
4.4.4	Temporal Patterns	65
4.4.5	RQ3 Answer	67
5	Activity Plan	69
6	Conclusion	70
6.1	Limitations	70
6.2	Future Work	70
7	Bibliography	72
A	Appendix I - Preprocessing	79
B	Appendix II - Structural Layer	81
C	Appendix III - Thematic Layer	83

Figures Index

Example of a bipartite affiliation network (a) and the relative one-mode projection (b)	5
Overview of TGIF structure	10
Preprocessing Pipeline	14
Bertopic Preprocessing Pipeline	16
IoC Preprocessing Pipeline	18
BERTopic Pipeline for G_1 and G_0	28
Forwarding visibility limitation	35
Corpus insights graphs	43
Language Distribution	43
Community size distribution	46
Daily share of the <i>Middle Eastern Conflict and Activism</i> macro-topic within G_1	48
Pairwise Jensen–Shannon distance between the G_0 macro-topic profiles of the 72 venue-level communities	50
Share of monolingual communities by community size in U	51
Language composition of top-20 communities by size	52
Hour-of-day fingerprints for the 20 highest-volume communities	52
Aggregate and per-community inbound source composition	55
Community forwarding distribution	55
Intra-community versus cross-community forwarding share across source levels	56
Distribution of recurrent feeders by the share of their forward volume sent to their single most-targeted community	57
Observed topic-concordant forwarding compared with the permutation null by flow regime	58
Flow-type composition of the 10,902 content-hash propagation edges	59
Content-hash propagation example	59
Concentration of malicious IoCs across communities	64
Observed share of malicious indicators per macro-topic against the share expected under the extracted-volume null	65
Lead–lag relationship for <i>Cybersecurity Vulnerabilities and Threats</i> (M13)	66

Tables Index

Corpus composition before and after the general cleaning pipeline	42
Corpus size at each stage of BERTopic-specific preprocessing	44
Size of the RQ1 structural corpus across the main graph-construction stages	45
Robustness comparison between the selected Leiden partition and alternative community detection algorithms	47
The two BERTopic topic spaces G_0 and G_1	48
Channel and community coverage at each stage leading to the analysis universes $U(G_0)$ and $U_1(G_1)$. . .	49
Comparison between the Leiden structural partition and attribute-based baseline partitions on U	53
IoC extraction and verification results over the whole corpus	61
Resolved and confirmed-malicious indicators by type	62
Origin and MIME-family composition of resolved and malicious MD5 indicators	63
Topic-space coverage of distinct confirmed-malicious indicators	64
Corpus-first malicious indicators by type, with MD5 hashes further decomposed by attachment MIME family .	67
Work Plan	69
Placeholder tokens used in BERTopic preprocessing	79
Sensitivity of the real-token threshold	80
Sensitivity of the language-mass threshold	80
Mapping from attachment MIME type to file-type family (<code>mime_family</code>)	80
Leiden resolution sweep summary	81
Silhouette sweep for the K -means topic baseline over the per-channel macro-topic distributions $P(m c)$.	82
Silhouette sweep for the K -means temporal baseline over the per-channel hour-of-day activity fingerprints . .	82
Final BERTopic pipeline configuration	83
HDBSCAN and UMAP selection	84
Ward macro-topic resolution sweep on the G_1 refit (223 fine topics)	84
Cross-correlation between macro-topic activity and IoC appearance, per G_1 macro-topic	85

Acronyms

ARI *Adjusted Rand Index.*

CTI *Cyber Threat Intelligence.*

HDBSCAN *Hierarchical Density-Based Spatial Clustering of Applications with Noise.*

HHI *Herfindahl–Hirschman Index.*

IoC *Indicator of Compromise.*

LLM *Large Language Model.*

NMI *Normalized Mutual Information.*

OSN *Online social network.*

TGIF *Telegram Communities Intelligence Framework.*

UMAP *Uniform Manifold Approximation and Projection.*

1 Introduction

Online social networks (OSNs) are a key part of how we communicate today. Platforms like X (formerly Twitter), Reddit, Facebook, and instant messaging services have changed how information is now exchanged. Beyond day-to-day social interaction, OSNs have become key infrastructures for political discussion, journalism, and increasingly for the coordination and information sharing among threat actors.

Among these platforms, Telegram has emerged as the primary communication channel for threat collectives. Its strong privacy protections, large group capacities, and tolerance for anonymous accounts make it a perfect ecosystem for actors looking to evade conventional monitoring. For this reason, hacktivist groups have been widely documented using Telegram to coordinate attack campaigns, recruit members, share malware tooling, and distribute IoCs [1, 2].

As the operational relevance of Telegram for threat actors continues to grow, so does the need for robust analytical methods to monitor and understand the communities that operate within it. However, existing approaches to Telegram threat intelligence focus mainly on individual channels or single cases. As a result, they provide limited insight into how threat communities are structurally connected, how content circulates between them, and how technical indicators are distributed across the ecosystem. Studies such as DarkGram [3] characterize the content of individual cybercriminal channels, while Arikkat et al. [4] in their studies extract IoCs from a curated set of CTI-focused channels, but neither examines how communities relate to each other nor how information flows between them. Conversely, the structural and flow-based methods that could recover this organization, such as forwarding-graph community detection, have largely been applied outside the CTI domain, to disinformation and political communication [5, 6].

This study addresses this gap. Rather than treating Telegram channels as independent elements, it analyzes the hacktivist Telegram ecosystem as an interconnected network of communities. Thus, it allows for cross-layer findings that single-channel or single-method studies cannot produce.

Based on this, this dissertation proposes a **Telegram Communities Intelligence Framework** (TGIF): a multi-layer analytical framework to analyze Telegram threat communities through community structure, topic modeling, information flow, and IoC.

The work is structured around three research questions, which together cover the three analytical dimensions of the framework:

RQ1 How are Telegram users organized into communities across hacktivist channels, and what topics characterize those user communities?

RQ2 How does content spread across those communities and which entities play key roles in that process?

RQ3 Do certain communities consistently produce more cyber threat indicators (such as malicious IPs, domains, or vulnerability references), and can their structure or topic focus help predict that?

The remainder of this dissertation is structured as follows. Chapter 2 reviews prior work on Telegram as a cybercrime and hacktivist ecosystem and positions this work with respect to the identified gaps. Chapter 3 presents the proposed framework and the methodology of each research question. Chapter 4 presents and discusses the results for each research question. Chapter 5 present the timeline of this work. Finally, Chapter 6 summarizes the contributions, discusses the limitations of the study, and outlines directions for future research.

2 Literature Review

This chapter presents the theoretical concepts necessary for the development of this project, along with the corresponding literature review.

2.1 Telegram

Released in 2013, Telegram is one of the most widely used instant messaging platforms in the world. According to [7], in 2025, reach \$870 million in revenue with more than 500 million daily active users. It allows users to share text messages, stickers, images, audio, videos, and files up to 2 GB.

In addition to the standard one-to-one messaging, the platform allows three main communication structures relevant to this study. *Groups* enable many-to-many communication among up to 200,000 members and can be public or private. The group administrator can additionally hide the member list, preventing the enumeration of participants. *Channels* support one-to-many broadcast communication with unlimited subscribers, and only administrators can publish content. Channels do not expose subscriber identities, except by indicating the total number. *Bots* extend the platform's functionality via a programmable API, allowing automated posting and data collection.

Furthermore, Telegram exposes its functionality through MTPROTO¹, a custom binary protocol that governs all client-server communication on the platform. Telethon² is a third-party, open-source Python library built on top of MTPROTO that enables developers to interact with Telegram groups and channels programmatically. It is widely used in academic research to collect Telegram data.

2.1.1 Telegram's Role in Hactivist and Cybercriminal Ecosystems

In addition to its standard use as a messaging app, Telegram has become one of the primary platforms for cybercriminal channels and hactivist collectives. Roy et al. [3] provide the first large-scale characterization of cybercriminal activity on the platform, identifying 339 channels that trade compromised access, leaked credentials, and offensive tooling. La Morgia et al. [8] describes a larger ecosystem of more than 120,000 public channels. Ricaldi et al. [2] attribute this concentration of threat activity to the perceived privacy, user anonymity, and ease of use of the platform, and Telegram has no native reputation system tied to a stable identity. Thus, actors are able to preserve long-lasting pseudonyms, in the absence of a verifiable real-world identity.

Most relevant to this work, Telegram has become a coordination hub for distributed denial-of-service (DDoS) campaigns. Vu et al. [9] analyze the principal Telegram channel of the pro-Ukrainian "IT Army", which distributed daily lists of targets to an organized volunteer base of over 200,000 subscriber. Finally, the Telegram Bot API

¹<https://core.telegram.org/mtproto>

²<https://docs.telethon.dev/en/stable/index.html>

is repeatedly used as a stealthy command-and-control (C2) channel, allowing malware to receive commands and exfiltrate data while blending with legitimate application traffic [10].

2.1.2 Indicators of Compromise on Telegram

By definition IoCs are observable technical artifacts associated with malicious activity, such as IP addresses, domains, URLs, and file hashes. In the context of Telegram, IoCs are relevant because cybercriminal and hacktivist channels frequently share leaked data, offensive tooling, malware-related resources, phishing links, and vulnerability information. Recent Telegram CTI studies show that the platform contains a large volume of technical artifacts relevant to threat intelligence [3, 4]. Arikkat et al. [4] identify 86,509 malicious IoCs from Telegram messages by combining channel selection, BERT-based message filtering, regex-based extraction of domains, IP addresses, URLs, hashes and CVEs, and validation through VirusTotal and NVD. Roy et al. [3] reach a similar conclusion from the perspective of cybercriminal activity channel by showing that Telegram content may include phishing URLs and malicious executables. Taken together, these studies indicate that Telegram is a valuable but noisy source of CTI, where the presence of a technical artifact in a message must be distinguished from the stronger claim that the artifact is malicious.

2.2 Community Detection in Social Networks

By definition, a network is a set of nodes connected by edges. In social network analysis, nodes usually represent social entities, such as individuals, accounts, groups, or organizations, while edges represent some form of relation or interaction between them [11]. These relations can describe direct interaction, such as replies or message forwarding, but they can also describe indirect association, such as participation in the same group or activity.

A specific class of social networks is represented by *affiliation networks* or bipartite networks. In an affiliation network, the two sets of nodes have different meanings and edges exist only between nodes of different types [11, 12]. This representation is particularly suitable for Telegram. Users and channels naturally form a bipartite structure in which one node set contains users and the other contains Telegram venues, such as groups or channels. An edge connects a user to a venue when the user posts at least one message in that venue.

A second network representation relevant to the Telegram analysis is the *interaction network*. In an interaction network nodes are connected by edges that represent an interaction. In the Telegram ecosystem, the nodes correspond to Telegram entities, such as channels, groups, users, or external sources, while the directed edges represent observed forwarding relations.

Affiliation networks are often transformed into a one-mode projection to study relations among nodes of the same type. When the projection is performed onto the user set, two users are connected if they share groups, while when it is performed on the group set, the edges are between them refers to shared members [13]. Figure 1

presents an example of an affiliation network and the relative one-mode projection.

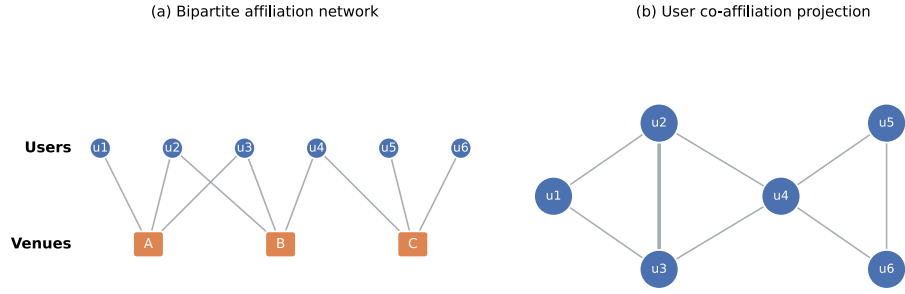


Figure 1: Example of a bipartite affiliation network (a) and the relative one-mode projection (b)

Community detection refers to the task of identifying groups of nodes that are more densely connected internally than with the rest of the network [14]. In social networks, communities are commonly interpreted as groups of users that share stronger relational, behavioral, or contextual similarity with each other than with outsiders. Communities can therefore reveal hidden groups of users who repeatedly appear in similar groups, even when the platform does not expose explicit membership lists or direct interpersonal relations.

However, projections of affiliation networks require caution as large venues can create many incidental edges simply because many users appear in the same place. Without correction, these large venues can dominate the projected graph and obscure smaller but more meaningful patterns of shared participation. For this reason, affiliation-network studies often use weighting or filtering strategies to reduce the influence of very large groups and emphasize repeated or more selective co-affiliation signals [13, 15].

2.2.1 Community Detection Algorithms

Community detection algorithms are commonly divided into two families according to the type of partition they produce. *Non-overlapping* methods assign each node to exactly one community, partitioning the network into distinct groups. *Overlapping* methods instead allow a node to belong to several communities at once.

This section focuses only on the approach most relevant to this study, thus not all community detection algorithms and categories are presented.

Non-overlapping algorithms. Following the classification of Smith et al.[14], these methods can be grouped according to their strategy they use:

1. *Divisive methods.* Divisive algorithms start from the complete graph and progressively split it into smaller communities. The most popular algorithm is the Edge-Betweenness algorithm proposed by Girvan and Newman [16], which iteratively removes edges with high betweenness because these edges are likely to

connect otherwise separate groups. This family is especially relevant when the goal is to identify structural bridges or points of separation between communities [17].

2. *Agglomerative methods.* Agglomerative algorithms follow the opposite strategy starting from individual nodes and progressively merge them into larger communities. Walktrap is based on the intuition that short random walks tend to remain within the same community, and therefore merges nodes and groups that are close under random-walk dynamics [18]. Label Propagation also follows a local logic, but instead of relying on random walks it assigns communities through an iterative consensus process in which each node adopts the most frequent label among its neighbours [19].
3. *Optimization-based methods.* Optimization-based algorithms search for the partition that optimizes a predefined objective function. The most common objective is modularity, which rewards partitions where the density of within-community edges is higher than expected under a random null model [20]. Modularity can be generalized through a resolution parameter that controls the scale of the detected communities, although it remains subject to a known resolution limit, by which communities below a scale-dependent size tend to be merged [21]. Alternative objectives, such as the Constant Potts Model, have been proposed to overcome this limit, as they are resolution-limit-free by construction [22]. The Louvain algorithm is one of the most popular algorithms [23] of this family. The Leiden algorithm refines the Louvain procedure to guarantee that detected communities are internally connected [24]. Infomap is also optimization-based, but rather than optimizing modularity, it identifies communities by modeling how a random walker moves through the network. The intuition is that if the walker tends to remain for long periods inside the same region before moving elsewhere, that region can be interpreted as a community [14, 25].

Overlapping algorithms. Overlapping algorithms are designed for settings in which a node may belong to more than one community at the same time. Examples of relevant overlapping algorithms include the following:

1. *Clique Percolation* A clique is defined as subset of nodes in which every pair of nodes is connected by an edge. The algorithm identifies communities as unions of adjacent k -cliques, cliques composed of k nodes, where two k -cliques are considered adjacent if they share $k-1$ nodes. Because a single node can belong to cliques that percolate into different communities, this method naturally produces overlapping communities [26].
2. *OSLOM* detects clusters that are statistically significant by checking whether its internal connectivity is unlikely to arise in a random network with the same degree distribution. Only clusters that are improbable under this null model are retained, and overlapping communities emerge when a node is statistically compatible with more than one significant cluster [27].
3. *BigCLAM* assigns each node a graded membership in multiple communities at once, modeling the network so that nodes sharing more communities are more likely to be connected [28].

2.3 Topic Modeling for Threat Intelligence

Topic modeling is a family of unsupervised machine learning techniques used to identify latent themes in large collections of documents. In the context of CTI, it provides a way to categorize large volumes of unstructured documents into interpretable thematic categories, supporting tasks such as characterizing what a channel or community discusses, identifying security-relevant themes, and relating topics to the actors that produce them. Topic modeling has been applied to several cyber-threat sources. Hossen et al. [29] and Moreno-Vera [30] studies show that topic modeling is useful not only for summarizing large textual corpora, but also for organizing heterogeneous CTI sources into interpretable analytical categories. The study Hossen et al. shows how topic modeling can detect latent topics such as DoS and Ransomware in hacking forums and social media, while the study of Moreno-Vera shows how topic modeling can be used to detect new emerging vulnerabilities in hacking forums.

2.3.1 Topic Modeling Approaches

Several families of topic-modeling algorithms are commonly used, and the choice among them influences the type of thematic structure that can be recovered. This section focuses only on the model most relevant to this study, thus not all topic modeling model and categories are presented.

Classical bag-of-words methods such as LDA [31] and NMF [32] model topics as weighted lists of words extracted from term co-occurrence. Although their main advantage is interpretability, they depend heavily on lexical overlap and can struggle with sparse, short, and noisy text [33]. However, they remain widely used in CTI settings. Hossen et al. [29] and Moreno-Vera [30] both apply them to hacker-forum analysis and vulnerability-exploitation discussions.

Embedding-based models move from word counts to dense semantic representations. Top2Vec [34] derives topics from the spatial distribution of document and word vectors, while BERTopic [35] clusters sentence-transformer embeddings and describes each cluster with a class-based TF-IDF representation. Both approaches have been proposed in the CTI context [36, 37].

Prompt-based models represent a newer direction. TopicGPT [38] delegates topic generation and assignment to a large language model, producing natural-language topic labels and descriptions rather than only bags of words. These methods can improve interpretability, but they require repeated model inference and therefore scale less easily to very large corpora.

Applying topic modeling to common messages remains challenging because they are short, noisy, multilingual, and often mixed with technical artifacts such as URLs, hashes, domains, and vulnerability identifiers. Short-text topic modeling is known to suffer from sparsity because individual documents contain too few word occurrences to estimate stable topics [33].

2.4 Information Flow and Diffusion

Information diffusion describes how content propagates from node to node across a network, generating a branching structure commonly called a *cascade* [39]. Reconstructing a cascade requires observing every individual propagation step. However, on Telegram diffusion is only partially observable. The main diffusion signal is the message *forward*, which records the original source of a message but does not necessarily expose all intermediate channels through which the content has traveled. For this reason, Telegram forwarding data is better interpreted as source-to-destination attribution rather than as a complete reconstruction of multi-hop cascades.

In diffusion research one main question is whether content remains inside community boundaries or is transmitted across them. Theories of structural holes and brokerage suggest that actors connecting otherwise weakly linked groups can mediate access to information and influence its circulation [40].

Because explicit forwarding metadata may be incomplete or may not capture copied content, diffusion can also be studied through the recurrence of the same artifact across communities. Zannettou et al. [41] use a content-centric approach, relying on fingerprints or hashes to identify the reuse of identical or near-identical material across platforms and groups.

2.5 Existing Telegram Intelligence Frameworks and Research Gap

The previous sections show that Telegram is not only a communication platform, but also an operational space for cybercriminal channels, hacktivist collectives, and threat-oriented communities. Prior work documents its use for trading compromised credentials, sharing offensive tooling, coordinating campaigns, distributing targets, and circulating technical indicators [3, 9, 10]. This makes Telegram relevant for cyber threat intelligence, but also creates an analytical challenge since threat activity does not necessarily occur inside isolated channels. For this reason, analyzing channels as independent feeds can miss how actors are structurally connected, how content moves between groups, and which communities are associated with the production or amplification of indicators.

The majority of Telegram CTI work to date has been content-centric and channel-bounded. Roy et al. [3] and Arikkat et al. [4] analyze cybercriminal channels as individual sources or as a curated pool of feeds, while Ricaldi et al. [42] focus on discovering and assembling such channels at scale through reference-driven snowballing. These works collect and characterize CTI-relevant channels, but they do not reconstruct the structure connecting those channels, nor examine how threat communities relate to each other or how content moves between them. In contrast, studies that explicitly reconstruct relations between channels have largely developed outside the CTI domain. Forwarding graphs, community detection, and bridge-node analysis have been used to map disinformation, conspiracy, and political communication on Telegram [5, 6, 43]. These works establish that Telegram channels form structured ecosystems, but they do not connect the recovered structure to the operational CTI environments.

This dissertation addresses this gap by proposing TGIF, a single pipeline for community-level Telegram CTI. TGIF integrates four different layers that bring structural, flow-based and indicator-level perspectives together into a unified, community-centered framework. By analyzing threat actor collectives as communities instead of isolated channels, TGIF makes it possible to answer questions that single-channel or single-method approaches cannot address directly.

3 Methodology and Implementation

This chapter presents the methodological framework adopted in this dissertation and the implementation choices. Section 3.1 presents the proposed framework, while Section 3.2 describes the environment setup and the evaluation measure. Section 3.3 describes the dataset and the preprocessing steps applied to prepare the corpus for analysis. Section 3.4 presents the methods used to detect and to characterize the community structure of the Telegram ecosystem. Section 3.5 describes the methodology used to analyze the information flow between communities and identify broker channels. Section 3.6 studies the relationship between technical indicators, topics, and community structure.

3.1 Framework Overview

This section presents the TGIF, the analytical framework proposed in this dissertation. Instead of treating Telegram channels as independent sources, the framework TGIF analyzes the ecosystem as an interconnected set of communities and studies how content and technical indicators are organized and circulate within it.

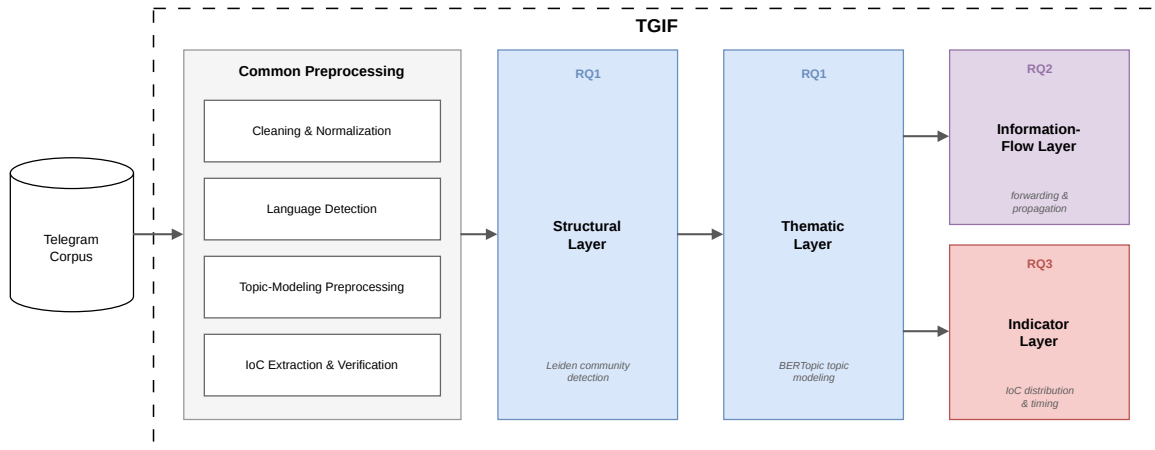


Figure 2: Overview of TGIF structure

As shown in Figure 2, the framework integrates four analytical layers into a single community-centered pipeline built on top of a common preprocessing stage (Section 3.3). Together, the four layers address the three research questions of this work.

1. **Structural layer (RQ1).** In this layer we recover the hidden structure of the ecosystem from the co-affiliation signal. The detail and process are described in Section 3.4. The output of this layer are the community partition that all other layers reuse as structural reference.
2. **Thematic layer (RQ1).** This layer characterizes what each communities discuss through topic modeling. Together with the structural layer, it answers RQ1 by describing communities along their topical, linguistic, and temporal profiles (Section 3.4).

3. **Information flow layer (RQ2).** This layer analyzes how content moves between communities, whether it remains within community boundaries, and which sources feed the ecosystem. This layer answers RQ2 (Section 3.5).
4. **Indicator layer (RQ3).** This layer contextualizes each technical indicator within the communities and topics defined by the previous layers. It examines whether these artifacts are spread across the ecosystem or concentrated in specific communities, and analyzes their temporal appearance. It answers RQ3 (Section 3.6).

It should be noted that these layers are not independent but a connected pipeline in which each layer utilizes the output of the previous ones. The framework itself is corpus-independent and the specific results it produces, reported in Chapter 4, are properties of the analyzed corpus.

3.2 Methodological Setup

This section presents the environment setup used in this study and formally define the evaluation measures used in our experiment.

3.2.1 Environment Setup

All analyses are conducted in a controlled environment provided by Constella Intelligence. The corpus is provided from the organization’s internal Solr database and is further described in Section 3.3.1. All private information is obscured and censored to prevent any disclosure of users and channels. The pipeline is built in Python, using Polars library for data handling operations and DuckDB for SQL aggregation. Statistical testing relies on SciPy and scikit-learn, while python-igraph is used for graph construction and community detection process. The language identification pipeline is described in detail in Section 3.3.2. Topic modeling is performed in a separate Google Colab machine equipped with an A100 GPU (160 GB RAM, 80 GB VRAM). Within this stage, *Large Language Model* (LLM)-assisted steps based on GPT-4o mini are used for topic domain labeling and for the natural-language representation of topics. Finally, IoC verification is performed by querying the VirusTotal API v3.

3.2.2 Evaluation Measures

This section defines the measures used in the dissertation. Measures are divided into categories according to their scope. We describe each measure at the operational level and refer to the original sources for their formal definition.

Concentration. Concentration measures assess how unevenly a single quantity is distributed across a set of categories. The *Gini coefficient* summarizes the inequality of a distribution in a single value between 0 and 1,

where a value of 0 indicates a perfectly even distribution and values close to 1 indicate that the quantity is concentrated in a small minority of categories. In our work, we apply it to the distribution of indicators across many communities or topics, where its normalized scale facilitates comparisons across distributions of different sizes [44]. The *Herfindahl–Hirschman Index* (HHI) also measures concentration but through the sum of squared shares, which makes it especially sensitive to a single dominant share. We apply it to the targeting behavior of an individual actor, to measure whether an orphan feeder concentrates its forwarding on one destination community or spreads it across several.

The *normalized Shannon entropy* describes how evenly a probability distribution is spread over its support, on a $[0, 1]$ scale obtained by normalizing by the maximum entropy. A value close to 0 identifies a focused distribution dominated by a few categories, while a value close to 1 identifies a near-uniform one [45]. We use it to distinguish thematically focused communities from thematically diverse ones, and linguistically homogeneous communities from multilingual ones.

The *Pearson χ^2 goodness-of-fit test* assesses whether an observed distribution departs from the distribution expected under a null model. A large statistic, with an associated p -value below the chosen significance level, leads to rejecting the null [46]. In our work, we use it to verify whether the observed concentration of malicious indicators exceeds what would be expected based on a reference value for the activity, a significant result indicates that this concentration cannot be explained by the volume of the activity.

Partition Agreement. Partition agreement measures assess how strongly two partitions of the same set of items coincide. We use them to validate the community partition against alternative detection algorithms and to test the how strongly a single channel attribute can reduce the structural partition. Following the community-detection literature adopted in this work [14, 17], the *Adjusted Rand Index* (ARI) and the *Normalized Mutual Information* (NMI) are used as the primary agreement measures. The ARI values of 1 indicates identical partitions and is centered on 0 for the agreement expected from random labelings, having been corrected for chance [47]. The NMI measures the information shared between two partitions on a scale between 0 and 1 and remains informative even when the partitions differ in granularity. This makes ARI and NMI complementary in our study. Furthermore, in our work we implemented *homogeneity* and *completeness* measures that give different perspective of any disagreement [48]. Homogeneity is high when each detected cluster maps to a single reference group, while completeness when each reference group stays within a single cluster. Together, these two measure clarify the source of disagreement, since high homogeneity with lower completeness indicates that one partition subdivides the other rather than merging unrelated groups.

Association. Association measures assess how strongly two variables are related, or how far an observed quantity departs from a specified baseline. *Cramer’s V* measures the association between two categorical variables on a $[0, 1]$ scale, where 0 indicates no association and 1 a perfect one [49]. We use it to examine how strongly an alternative

partition is associated with the structural partition. The *lift* compares an observed quantity with the quantity expected under a baseline. A value above 1 indicates over-representation, while below 1 under-representation. The measure is generic, and each analysis specifies its own expected term, namely the topic frequency outside a community in community profiling (Section 4.2.1), a permutation-based null in the forwarding analysis (Section 3.5.3), and the extracted-indicator share in the IoC analysis (Section 3.6.1). Finally, the *Pearson correlation coefficient* r measures the linear association between two series, ranging between -1 and 1 . In our study, we use it in our temporal analysis to evaluate r between one series and the other shifted by a temporal lag.

Similarity and distance. The *Jensen–Shannon distance* measures the dissimilarity between two probability distributions. A value of 0 indicates identical distributions, while 1 indicates two distribution totally with no element in common [50]. In our study, we used the *Jensen–Shannon distance* to compare the dissimilarity between the macro-topic distributions of communities, capturing differences in their overall thematic profile beyond individual high-lift topics.

3.3 Dataset & Preprocessing

This section is divided into Section 3.3.1 and Section 3.3.2. Section 3.3.1 analyzes the composition, origin and limitation of the dataset, while Section 3.3.2 presents the pipeline used to convert raw data into inputs for our study.

3.3.1 Dataset Description

The input dataset of this study is provided by the Constella Intelligence organization ³. The corpus contains messages and attachments captured by the organization in both public and private groups. These groups are related to cyber criminal activities such as hacktivism, breach-data, infostealer system, and war propaganda.

The raw data amounts to approximately 10 GB of uncompressed JSON ⁴. Each record corresponds to a Telegram message and contains its metadata. These metadata always include the sender, channel, date, and text content. When available, it also contains attachment information, replies, and additional metadata for forwarding ⁵.

The dataset contains messages and attachments captured from 01 January 2025 to 25 March 2026. We selected this time frame to focus the study on the most recent material and active threats.

a) Dataset Limitations

Several limitations of the dataset should be acknowledged.

³<https://constella.ai/>

⁴This dataset is only part of Constella Organization Telegram dataset.

⁵The metadata fields described here are not exhaustive, but represent only a subset of the fields captured by Constella Intelligence.

1. The Constella organization continuously adds channels and backfills their historical data only when is possible, so the dataset is not temporally homogeneous in terms of coverage. As a result, the raw daily counts increase as the number of channels observed grows, not necessarily because of increased activity. Therefore, all temporal analysis of this dissertation required normalization.
2. Some metadata fields are not available across the corpus because the organization collection pipeline evolved over time. As a result, the data set is not temporally homogeneous with respect to the metadata coverage. In particular, metadata related to forwards and replies was only captured from February 2026 onward.
3. Although the organization only collects channels related to the cybercriminal and threat-related ecosystem, the monitored dataset still includes sources in which automated activity is dominant. In some channels, bots and spam accounts account for the majority of observed users and messages.

3.3.2 Preprocessing Pipeline

The preprocessing pipeline is divided into three sequential stages, summarized in Figure 3. The pipeline takes as input two raw tables, *Raw Messages* and *Raw Attachments*, which are first combined into a single unified dataset.

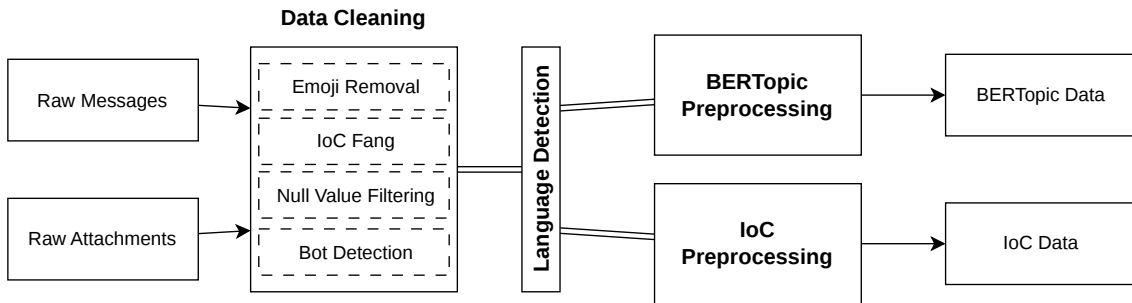


Figure 3: Preprocessing Pipeline

In the first stage (a), the text is normalized by removing emoji characters, reversing IoC defanging patterns, and by removing records with null content. In the *Language Detection* stage (b), a language label is assigned to each channel. The third stage diverges into two different preprocessing branches. The *BERTopic Preprocessing* (c) prepares the corpus for future topic modeling. The *IoC Preprocessing* (d) described the procedure used to extract and verify IoCs.

a) Data Cleaning

Due to the nature of messaging apps, the raw data contains significant noise and irrelevant information. Therefore, to avoid complications in subsequent stages of our analysis, the data must be filtered and cleaned before processing.

1. *Emoji Removal*. Unicode emoji characters are removed using a block-range filter according to the Unicode Character Database ⁶. Preserving such messages would increase the vocabulary size and, in the case of messages consisting solely of emojis, result in embedding vectors that contain no information. Therefore, messages consisting only of emojis are discarded during the Data Cleaning Phase.
2. *IoC Fanging*. Common defang patterns are reversed using a set of regular expressions before any further processing. This step precedes both IoCs extraction and placeholder tokenization and ensures that all IoCs have the same pattern.

Due to the absence of a standard, the following rules are derived from the common practices observed and cataloged in the *ioc-fang* open source project ⁷.

- **Protocol obfuscation:** `hxxp` and `hxxps` are converted to `http` and `https`.
- **Dot replacement:** bracketed or altered dots such as `[.]`, `(.)`, or `{.}` are restored to `"."`.
- **At-symbol obfuscation:** patterns like `[@]` or `[at]` are converted back to `@` in email addresses.
- **Colon obfuscation:** `[:]` is restored to `:` in URLs.

These transformations cover the most frequently observed defanging conventions and are applied sequentially using deterministic regular expression that ensure reproducibility.

3. *Empty Record Filtering*. Messages that contain neither textual content nor associated technical artifacts are removed from the dataset because they do not provide information for our analysis. Furthermore, two categories of records are discarded:

- **Content-free messages:** records whose content is null or empty and which have no attachment.
- **Telegram service messages:** records automatically generated by the Telegram platform (e.g. channel creation notices, pinned-message notifications) that do not provide behavioral topic signal.

4. *Bot Detection*. The `senderBot` attribute exposed by the Telegram API is used to identify and flag bot-generated content during preprocessing. Bots are not removed from the corpus but only flagged. For the purpose of community detection, they are treated as noise. However, for topic modeling, bot messages are retained as they often act as content distributors.

The `senderBot` flag is only available when the sender metadata has been successfully resolved and stored by the collection pipeline. As a result, some automated accounts may remain unflagged in the corpus, producing possible false negatives.

⁶<https://www.unicode.org/ucd/>

⁷<https://ioc-fang.github.io/>

b) Language Detection.

Language detection is conducted at the *channel level* instead of the message level. This choice is justified by the fact that Telegram channels are mostly monolingual, and the dominant language in a sample of messages is representative for the channel as a whole.

Furthermore, it is also motivated by the fact that classifying each message independently would be computationally unfeasible given the corpus's size.

Detection is performed using the `lid.176.bin` model distributed with the FastText library [51, 52].

For each channel, we sample $N = 300$ messages that contain at least 25 character. In this way messages that are too short to provide a trustworthy linguistic signal are filtered from the language detection.

For each message sampled, we apply the FastText classifier, which returns a predicted language label and a confidence score. The dominant language of the channel is then assigned to the language label that is most frequently classified among the sampled messages.

Channels that do not contain enough textual content for reliable language identification are treated as language-unknown in all future analyzes.

c) BERTopic Preprocessing.

Telegram messages are short by nature, making them not suitable document units for topic modeling. Therefore, we aggregate all messages posted within the same channel on the same day into a single channel-day document. This enable to retains both channel-level context and the temporal structure of the corpus.

Furthermore, after the general cleaning stage described in Section 3.3, the corpus undergoes an additional preprocessing stage specifically designed for BERTopic. The goal of this stage is to reduce vocabulary sparsity, remove high-cardinality artifacts, and retain only channel-day documents with sufficient natural-language content for our scope. The preprocessing stages are summarized in Figure 4.

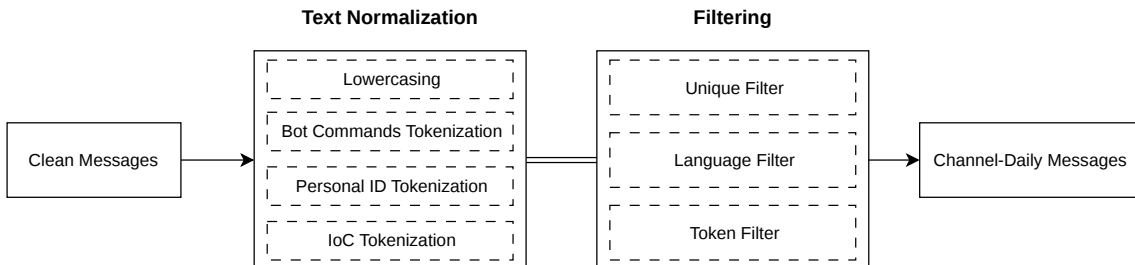


Figure 4: Bertopic Preprocessing Pipeline

1. *Text Normalization.* Raw message content is normalized to reduce vocabulary sparsity and preserve the semantic content of each message by reducing the impact of technical artifacts and platform-specific tokens

that would otherwise inflate the vocabulary.

- (a) *Lowercase*. All message content is converted to lowercase. This step reduces vocabulary sparsity in the BERTopic embedding model and makes sure that future string comparisons are not case-sensitive.
 - (b) *Bot Commands Tokenization*. Telegram allows users to interact with bots via slash commands, which are prefixed with a / character (e.g., /start, /help, /settings). These commands are typically used as standalone tokens at the beginning of a message, but can also appear elsewhere in the text. These strings do not contain topical content and would otherwise inflate the vocabulary with channel-specific command names. Therefore, each bot command is replaced with a placeholder token, according to the definition of Telegram bot commands ⁸.
 - (c) *Personal Identifier Tokenization*. User mentions (@username), email addresses, phone numbers, and numeric sequences are high-cardinality identifiers that do not carry topic signal relevant to topic modeling. Preserving their raw values would fragment the vocabulary and degrade the quality of the resulting topic representations. Therefore, each identifier type is replaced with a typed placeholder token.
 - (d) *IoC and technical artifact tokenization*. Cybersecurity-related artifacts are detected and replaced with typed placeholders. Furthermore, this includes also known IoC patterns.
2. *Filtering*. After text normalization, a sequence of filters is applied to remove documents that do not contain enough semantic content for topic modeling. It is important to note that filtering is based on *non-placeholder tokens*. This prevents channel-day documents dominated by URLs, identifiers, hashes, or other technical artifacts from being retained solely because they contain many placeholder tokens.
- (a) *Message-level token filter*. Individual messages with fewer than a threshold of 3 non-placeholder tokens are discarded. Messages that consist solely of placeholders are discarded because they do not provide sufficient information for topic modeling.
 - (b) *Placeholder collapse*. Consecutive placeholder tokens are collapsed to reduce repetition introduced by long lists of technical artifacts.
 - (c) *Channel-day deduplication*. Telegram channels frequently contain repeated messages generated by bots, spam accounts, or coordinated reposting activity. To prevent this message from monopolizing the channel's topic on that day, we remove duplicate messages within each channel-day unit. This ensures that repeated copies of the same message do not dominate the channel-day document, allowing each unique message to have the same influence on the final topic assignment.

⁸<https://core.telegram.org/bots/api#botcommand>

- (d) *Channel-day token filter*. After messages are aggregated into channel-day documents, channel-days with fewer than 15 aggregated non-placeholder tokens are discarded. This value is selected through a sensitivity analysis over alternative thresholds, reported in the Appendix A.2. This ensures that each document carries enough textual mass.
- (e) *Language-mass filter*. Languages with fewer than a threshold of 2,500 channel-day messages in the cleaned corpus are excluded. This value is selected through a sensitivity analysis over alternative thresholds, reported in the Appendix A.3. Furthermore, the need for a threshold was demonstrated in the study conducted by Conneau et al. [53] showing that languages with low resources are generally harder to represent in shared multilingual spaces.

It should be noted that using placeholder tokens preserves information about the presence and type of cybersecurity-relevant entities while remaining independent of the specific indicator values.

The complete list of all applied placeholders is available in the Appendix A.1.

d) IoC Preprocessing

The cleaned corpus is used to build the candidate list of IoCs. The pipeline extracts matching indicators, filters noise and verifies the surviving candidates against an external reputation service. The process is summarized in Fig. 5.

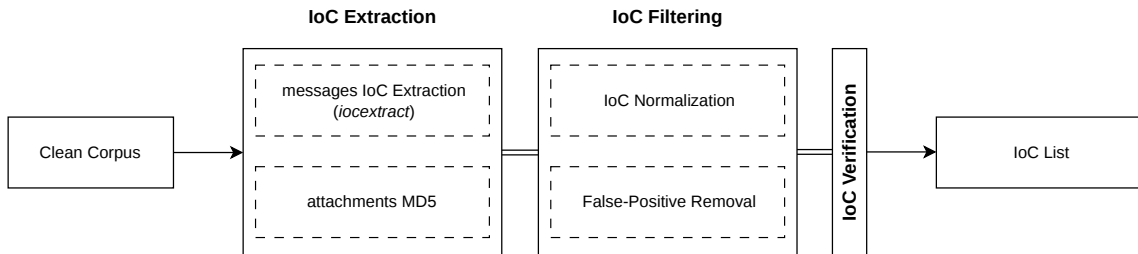


Figure 5: IoC Preprocessing Pipeline

1. *IoC Extraction*. IoC extraction combines two distinct layers. Firstly, we process each message with the `iocextract`⁹ library to detect URLs, IPv4 and IPv6 addresses, MD5, SHA1, SHA256, and SHA512 hashes from each message. Second, the output is combined with the MD5 hashes present in the attachment metadata.
2. *IoC Filtering*. In this stage, we normalize and filter the extracted IoC list. Normalization is applied to MD5 attachments, which are aggregated into macro categories based on the mime type, the full list is available in the Appendix A.4. Filtering is applied by discarding malformed IoCs and false positives erroneously captured

⁹<https://github.com/InQuest/iocextract>

by the `iocextract` library. In addition, we exclude valid indicators that do not carry a compromise signal, such as popular domains or hashes of widely distributed software. URLs are matched against the Tranco ¹⁰ ranking of the most popular domains. Hostnames within the top 100,000 are treated as benign popular infrastructure. Hash indicators are queried against the CIRCL hashlookup ¹¹ service, and those containing well-known provenance metadata are flagged as legitimate files. Hashes derived from media type attachments are excluded as they correspond to images, videos, and GIFs. IP addresses falling into private, loopback, and reserved ranges are also excluded [54, 55].

3. *IoC Verification*. VirusTotal API (v3) ¹² is used to verify the remaining candidate IoCs. An indicator is classified as malicious when flagged by at least one antivirus engines, consistent with the threshold adopted by Roy et al. and Arikkat et al. [3, 4].

A further aggregation is also computed for downstream analysis.

1. *extracted* represents all the IoCs extracted.
2. *VT-present* represents all the IoCs with a VirusTotal report. Include both malicious and not.
3. *confirmed-malicious* represent the set IoCs confirmed malicious from at least one engine by the VirusTotal report.

The verified indicators represent the final IoC corpus analyzed in Section 3.6.

3.4 Methods for RQ1: Community Structure and Topical Profiling

This section describes the methodology of the structural and thematic layers of TGIF, which together answer RQ1.

The rest of this section is structured as follow. Section 3.4.1 analyzes the structural layer of community detection by creating a user co-affiliation system. Section 3.4.2 describes the BERTopic pipeline and Section 3.4.3 enriches each community with additional metadata. Finally, Section 3.4.4 presents and compares alternative partitions with the core co-affiliation structure.

3.4.1 User Co-Affiliation Graph Construction

The community structure is created from a user co-affiliation representation. Our objective is to capture the structural composition through shared participation. We adopt a co-affiliation graph rather than a forwarding graph because of the limitations of the forwarding metadata, discussed in Section 3.3.

¹⁰<https://tranco-list.eu>

¹¹<https://hashlookup.circl.lu>

¹²<https://www.virustotal.com>

Therefore, the methodology can be divided into three core stages. First, user participation is represented as a bipartite graph linking users to venues. This bipartite structure is then projected onto a weighted, undirected graph $user \leftrightarrow user$, in which two users are connected if they participate in at least one shared venue. Second, the Leiden algorithm [24] is used for community detection and to produce partitions of structurally related users. Third, the resulting user communities are propagated back to Telegram venues, allowing each group or channel to be characterized according to the community composition of its active users.

The weight of each edge represents the strength of their shared participation between venues. The graph is formally defined in Equation 3.1.

$$G = (U, E) \quad (3.1)$$

where U is the set of active users and E is the set of weighted edges between pairs of users. An edge $e_{ij} \in E$ exists between u_i and u_j if they share at least one venue, and its weight w_{ij} quantifies the strength of their co-affiliation.

It should be noted that participation is encoded as a binary affiliation signal rather than as a message-volume weight. This choice follows the literature on two-mode and affiliation networks, where relations among actors are derived from shared membership in events, groups, or other affiliation contexts before being projected onto a one-mode graph [13, 56].

Posting activity in the Telegram ecosystem is highly uneven and may be dominated by administrators and automated accounts. Therefore, a message-volume projection would capture the intensity of posting rather than structural co-participation. For this reason, the construction of G is organized into sequential steps designed to maintain the analysis's focus on co-presence while reducing the influence of large venues and hyperactive posters.

a) Step 0: Venue definition.

As mentioned before, Telegram distinguishes between *groups*, which support many-to-many communication and allow all members to post, and *channels*, which are mainly broadcast spaces where only administrators can publish messages. Therefore, entities with a small set of posting users do not provide a meaningful co-affiliation signal and do not produce edges in the projected graph $user \leftrightarrow user$.

For this reason, we define a *venue* as any Telegram entity with at least k distinct non-bot posting users. This definition is based on observed posting activity rather than on subscriber or membership lists, which are not available in the dataset.

Let $\mathcal{C}$ denote the set of all Telegram entities in the dataset and let U denote the set of non-bot users identified during preprocessing. The venue set V is formally defined in Equation 3.2.

$$V = \{c \in \mathcal{C} : |\{u \in U : u \text{ has posted at least one message in } c\}| \geq k\}. \quad (3.2)$$

In this study, the threshold is set to $k = 3$.

b) Step 1: Bipartite affiliation matrix.

Once the set of venues has been defined, the next step is to represent user participation in a form that can later be projected into a graph. To do so, participation is encoded in the binary bipartite affiliation matrix illustrated in the Equation 3.3, with rows corresponding to users and columns to venues.

$$B \in \{0, 1\}^{|U| \times |V|}, \quad B_{u,v} = \begin{cases} 1 & \text{if user } u \text{ has posted at least one message in venue } v, \\ 0 & \text{otherwise,} \end{cases} \quad (3.3)$$

Here, U denotes the set of users not flagged as bots and observed in at least one venue, while V denotes the set of venues obtained from the previous step. In the implementation, the matrix is stored in sparse form as a $user \leftrightarrow venue$ edge list. Each row corresponds to one observed affiliation (u, v) for which $B_{u,v} = 1$.

It should be noted that participation is encoded as a binary signal rather than weighted by message count. This choice ensures that the analysis captures the shared presence between venues rather than differences in posting intensity, which is not the main object of interest in this study.

c) Step 2: User and venue filtering.

The bipartite matrix B is then filtered before projection. The purpose of this step is to reduce noise and ensure that the resulting graph $user \leftrightarrow user$ shows a meaningful structural affiliation rather than artifacts of the platform or highly local participation patterns.

Therefore, two filters are applied to B : a minimum number of venues per user and a minimum number of users per venue. Let

$$d_U(u) = \sum_{v \in V} B_{u,v} \quad \text{and} \quad d_V(v) = \sum_{u \in U} B_{u,v}$$

denote, respectively, the number of venues associated with user u and the number of users associated with venue v .

The two filters are described in Equation 3.4.

$$U' = \{ u \in U : d_U(u) \geq m_U \} \quad V' = \{ v \in V : d_V(v) \geq m_V \} \quad (3.4)$$

In this study, the thresholds are set to $m_U = 2$ and $m_V = 3$.

The first condition retains only users who are active in at least two venues. Users observed in a single venue provide only local evidence of co-presence and do not help identify cross-venue affiliation patterns. Including them directly in the projected graph would substantially increase graph density while including limited information for community discovery.

The second condition retains only venues with at least three active users after user filtering. This ensures that each venue included in the projection continues to provide sufficient participation evidence to generate meaningful co-affiliation. Although venues were already required to contain at least three non-bot posting users in Step 0, this condition is applied again because removing single-venue users can reduce the effective number of users remaining in a venue.

The two filters are applied iteratively until convergence, since removing users can cause some venues to fall below the minimum user threshold, and removing venues can, in turn, cause some users to fall below the minimum venue threshold.

Equation 3.5 defines the output of this step as a new filtered matrix B' .

$$B' = B[U', V']. \quad (3.5)$$

It is important to note that users excluded because they appear in only one venue are not discarded from the analysis. They are treated as peripheral users and re-attached after community detection based on the community composition of their venue.

d) Step 3: Weighted User–User Projection.

The objective of this step is to transform the co-affiliation matrix B' into a one-mode projection in which user communities can be detected directly. Therefore, the matrix is projected onto a weighted graph $user \leftrightarrow user$, in which edges represent shared participation in the same venues.

$P = B'B'^\top$ represents a simple initial projection, which counts how many venues are shared by each pair of users. However, this raw projection becomes problematic when venue sizes are highly heterogeneous. A venue with d active users induces $d(d-1)/2$ user pairs, so large venues mechanically contribute more than small ones. As a result, some highly populated venues can dominate the projection.

To reduce this effect, the main weighting scheme adopted in this work is the *Newman discounted projection* [15] defined in Equation 3.6, in which each shared venue contributes to the edge weight inversely to its size.

$$w^{\text{disc}}(u_i, u_j) = \sum_{v \in V(u_i) \cap V(u_j)} \frac{1}{d(v) - 1}, \quad (3.6)$$

where $V(u)$ denotes the set of venues in which user u is active, and $d(v)$ is the number of filtered users affiliated with venue v .

The reason for this choice is that a shared presence in a small, specialized venue is a stronger signal of social proximity than in a large public aggregation channel with thousands of members, where co-occurrence may be incidental. The Newman weighting encodes this intuition directly by penalizing edges produced by large venues.

As a robustness baseline, the raw overlap projection defined in Equation 3.7 is also retained.

$$w^{\text{raw}}(u_i, u_j) = |V(u_i) \cap V(u_j)|, \quad (3.7)$$

corresponding to the unweighted co-occurrence count. Comparing the partitions obtained from w^{disc} and w^{raw} allows us to check whether the detected communities are stable across alternative projection schemes or instead depend strongly on the down-weighting of large venues.

Therefore, this step defines two weighted $user \leftrightarrow user$ graphs over the same node and edge sets as presented in Equation 3.8.

$$G^{\text{raw}} = (U', E, w^{\text{raw}}) \quad \text{and} \quad G^{\text{disc}} = (U', E, w^{\text{disc}}), \quad (3.8)$$

where U' is the set of users from the previous step, and E is the set of unordered user pairs (u_i, u_j) that share at least one venue.

e) Step 4: Sparsification.

Even after venue-size discounting, the projected $user \leftrightarrow user$ graph can remain very dense, since any pair of users who share at least one venue induces an edge. Therefore, an edge sparsification process is used to improve interpretability while reducing noise before community detection.

Therefore, we applied two thresholds on the edge weight. First, an edge is maintained only if the two users share at least s venues in the raw projection. Second, among the edges satisfying the first condition, a threshold is applied to the Newman-discounted weight.

In our work, we set the minimum overlap condition to $s = 2$. This removes edges induced by a single shared venue, which are more likely to reflect incidental co-presence, especially in large public groups that are common in the Telegram ecosystem. The discounted-weight threshold τ is then empirically selected from the distribution of w^{disc} among edges satisfying $w^{\text{raw}} \geq s$. Specifically, τ is set to the median of this conditional distribution. Different s were tested.

As a result, this step reduces the density of the projected graph while preserving repeated and more selective co-affiliation ties. The resulting sparsified graph is formally represented by the Equation 3.9.

$$G^{\text{disc},*} = (U^*, E^*, w^{\text{disc}}). \quad (3.9)$$

f) Step 5: Community detection.

Once the sparsified graph has been obtained, the next step is to identify a hidden structural composition of the co-affiliation graph. Community detection is therefore performed on the weighted undirected graph $G^{\text{disc},*}$ using the Leiden algorithm [24]. A community is defined as a set of users whose patterns overlap more with each other than with the rest of the filtered user population.

Formally, the output of the algorithm is a partition as illustrated in Equation 3.10. Here, each $C_k \subseteq U^*$ is a

pairwise disjoint community of the sparsified user set U^* .

$$\hat{\mathcal{P}}_\gamma = \{C_1, C_2, \dots, C_K\}, \quad C_k \subseteq U^*, \quad \bigcup_{k=1}^K C_k = U^*. \quad (3.10)$$

We adopt a non-overlapping formulation because the structural questions addressed in this work require communities to be treated as distinct analytical units. Cross-community flow and bridging are most directly defined against a partition in which each user belongs to a single community. By contrast, under overlapping membership, a user assigned to several groups is partly internal to each of them by construction, making the boundary between within-community and between-community activity less clear.

According to [14], the choice of detection algorithm should align with the process that generates communities and with the use of the resulting partition. In this study, communities are created from co-affiliation structure rather than a flow, and the output partition is used as a fixed grouping variable. Therefore, we focus on modularity-optimization methods that partition the graph by maximizing the density of within-community ties relative to a null model, rather than flow-based alternatives, which are better suited to issues related to information transmission.

In the family of modularity-based algorithms, we prefer Leiden over the Louvain algorithm [23] as it provides a formal guaranty that the communities produced are internally connected, a property that Louvain does not guaranty, and is known to generate poorly connected or even disconnected communities [24]. This property is especially relevant in this work, where the graph is obtained by projecting the bipartite *user* $\leftrightarrow$ *venue* network to a one-mode *user* $\leftrightarrow$ *user* graph, which may still contain dense local structures, even after venue-size discounting and sparsification. Leiden additionally scales to large weighted graphs, which is necessary at the size of the projected user graph.

In our work, Leiden is applied with the Reichardt–Bornholdt modularity under a configuration null model [57] as quality function, evaluated on the discounted edge weights w^{disc} . This corresponds to the standard modularity [20], which is the standard objective in community detection in social networks [17], generalized by a resolution parameter γ , with which it coincides at $\gamma = 1$. This parameter controls the granularity of the partition with lower values tending to produce fewer and larger communities, and higher values producing smaller and more numerous ones. The resolution parameter mitigates, but does not remove, the resolution limit introduced by modularity [21], by which communities below a scale may be merged. The value of γ is selected through an exploratory parameter sweep, in which candidate values are compared according to quality, number of communities, and size of the largest community. In our study, we select $\gamma = 0.6$. The complete analysis is reported in the Appendix B.1.

g) Step 6: Robustness validation.

As mentioned in the previous step, community detection partitions depend on the chosen algorithm and its definition of community structure. According [14, 17], no single algorithm is optimal in all situations. The objective of this step is to assess whether the selected partition is consistent with the partitions produced by algorithms based

on different principles. The Leiden partition selected in the previous step is therefore validated against alternative detection methods before being adopted as structural reference for this work.

As stated previously, resolution sensitivity is addressed separately by the sweep analysis used to select γ .

Three alternative algorithms are applied to the same sparsified weighted graph $G^{\text{disc},*}$, using the Newman-discounted weight w^{disc} and the Leiden partition as reference.

1. *Louvain* provides a close baseline in the same family of modularity-based algorithms. An agreement indicates that the detected structure is not an artifact of the Leiden refinement procedure.
2. *Infomap* provides a comparison against a different paradigm, defining communities through the map equation over random-walk dynamics rather than through modularity.
3. *Label Propagation* provides a local diffusion-based baseline. Nodes iteratively adopt the most frequent label among their neighbors, producing communities from local consensus rather than from modularity optimization or random-walk flow compression.

Agreement between two partitions is measured using the ARI and NMI. Homogeneity and completeness are also evaluated, since ARI penalizes differences in granularity. Structural diagnostics, including the number of communities, the relative size of the largest community, and the number of singletons, are recorded to ensure each alternative is interpretable and not dominated by fragmentation or a single community.

h) Step 7: Community label propagation

Although the Leiden algorithm in Step 5 detects communities for users, many studies of this work have an interest in communities at the venue level. Therefore, the objective of this step is to propagate the user community labels back to Telegram venues using the filtered $user \leftrightarrow venue$ affiliation matrix from Step 2.

For each venue, we count how many users belong to each community, producing a venue-level distribution over user communities. The community that represents the largest share of labeled users is defined as the *dominant community* of the venue.

We implement a dominance threshold to distinguish venues that are strongly associated with a single structural community from venues that bridge multiple communities. If the dominant community is shared by at least 60% of the venue's labeled users, the venue is classified as *dominant-community venue*. Otherwise, it is classified as *multi-community venue*. This distinction separates venues mainly associated with a single structural community from those containing users of several communities.

The output of this step is therefore a per-venue community profile, comprising the community distribution, its dominant community, and the resulting venue label.

It should be mentioned that due to the sparsification process of Step 4, not every user present in the Step 2 matrix is necessarily assigned to a community.

i) Step 8: Community Label Propagation

The objective of this step is to extend the community assignments obtained from the previous sections to the full corpus without modifying the detected community structure. Firstly, peripheral users that were excluded from the community detection are attached to communities using the affiliation evidence derived from the venues in which they participate (a). Second, this new complete user set is used to project a community label to every channel in the corpus (b).

Step 8a: User Label Propagation A large number of users is filter out from the community detection during Steps 2 and Steps 4. Instead of completely discarding them, we decide to attach each peripheral user to a community using the affiliation evidence available from the venues in which it participates. This re-attachment does not alter the community structure itself, which is already fixed by the Leiden partition of Step 5. The objective of this step is to extend the community coverage to users whose participation is too sparse to enter in the community detection.

The procedure uses the venue-level community distributions estimated in Step 7. Formally, the assigned community $\hat{c}(u)$ is defined in Equation 3.11.

$$\hat{c}(u) = \arg \max_c \sum_{v \in V(u)} w(v) P(c | v), \quad (3.11)$$

where u represent a peripheral user and $V(u)$ is the set of venues in which u participates, while $w(v)$ represents the weights that count for each venue coverage and size. Following this equation, each peripheral user u is assigned to the community that maximizes the aggregated weighted evidence across all its venues. We define as *orphan* users who participate only in venues without any labeled core.

It should be noted that labels created from this projection procedure are an approximation of the community structure therefore a confidence score is also computed and used in the next step. A confidence of 1 is assigned to users labeled by Leiden algorithm and 0 to *orphan*, while for peripheral user the confidence score is the top score share [58].

Step 8b: Channel Label Propagation. Now that every user is assigned to a community or recorded as orphan, we project this labeling onto the complete set of venues in the corpus, including venues that were not part of the sparsified graph used for community detection. For each channel, participating users are aggregated by community, with each user weighted by its assignment confidence defined in Step 8a. Each channel is assigned the label of the community with the largest weighted share using the same threshold logic used previously in Step 7. Channels whose dominant community reaches the majority threshold are classified as *community channels*. Channels containing labeled users but not meeting the threshold are classified as *multi-community channels*. Channels whose users do not carry any community label are classified as *orphan channels*.

3.4.2 Topic Modeling

We used topic modeling to characterize the topics of interest of Telegram channels and of the communities to which they belong. The resulting topics are used in all the downstream analysis.

In particular, among all candidate approaches, this study uses BERTopic [35]. The model is selected for its performance on short, noisy social media text and for its modular framework.

Topic modeling is applied to the output of the BERTopic preprocess described in Section 3.3 to learn a global topic space from the broadest possible textual base. The topics are then restricted to the community-analysis universe channels for downstream interpretation and comparison.

This design ensures that the topic space is not shaped solely by the smaller subset and preserves the comparability between the entire ecosystem and the channels retained for community-level analysis. Thus, the model is fitted globally and interpreted on the subset of channels retained for community-level analysis.

Additionally, in Section (f) we hierarchically aggregate each topic into a smaller set of macro-topics to support interpretation at the community and information-flow level, where each unit must be summarized by the dominant themes.

Finally, in Section (g) we derived a more specific CTI-focused from the corpus-wide one by restricting the corpus to its CTI topic and refitting. The initial topics identified across the entire corpus are referred to as G_0 and are used to characterize communities at the ecosystem level and for the overall analysis of the corpus, while G_1 refers to specific CTI topics and provides the thematic representation used for further analysis of information flow and IoCs distribution.

Although this study follows the BERTopic pipeline, the corpus size requires a custom modular pipeline in which embeddings, dimensionality reduction, and clustering are computed as separate stages.

Figure 6 illustrates the complete custom BERTopic pipeline adopted in this study. The complete configuration is presented in the Appendix C.1.

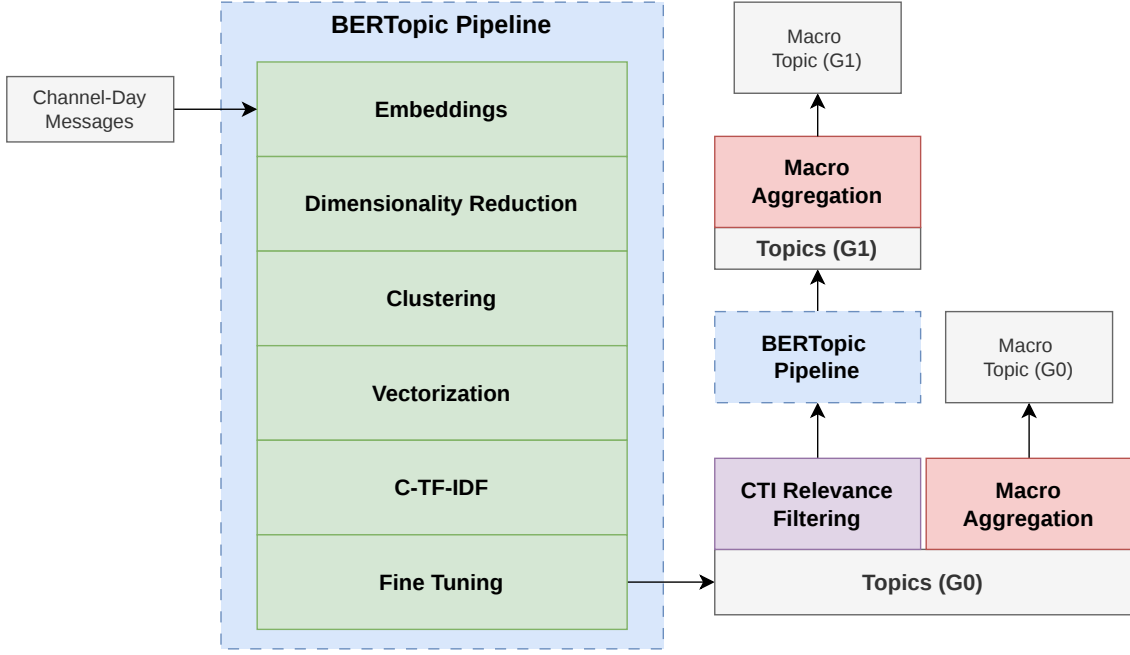


Figure 6: BERTopic Pipeline for G_1 and G_0

a) Embeddings

The first step in our topic-modeling pipeline is to transform each preprocessed channel-day document into a dense vector representation. This step uses the output of the BERTopic preprocessing pipeline described in Section c).

Each channel-day document is embedded using the multilingual sentence-transformer model `paraphrase-multilingual-MiniLM-L12-v2`¹³.

The model encoder supports a maximum input length of 128 tokens. Therefore, messages shorter than 128 tokens are encoded directly, while longer messages are divided into overlapping blocks of up to 128 tokens. Thus, guaranteeing that long messages are represented without exceeding the encoder input limit.

Each chunk is then independently encoded into a 384-dimensional vector. For messages split into multiple chunks, mean pooling is used to obtain a single message-level embedding. The final channel-day embedding is then obtained by averaging the embeddings of all messages in the same channel-day.

In this way, each message contributes uniformly to the channel-day representation, independently of its original length or number of generated chunks.

Finally, we normalized the resulting channel-day embedding to unit length, producing a single fixed-size vector for each channel-day document.

¹³<https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2>

b) Dimensionality Reduction

The embedding process represents each channel-day document as a dense vector within a 384-dimensional semantic space. While these embeddings capture extensive multilingual semantic information, clustering them directly in such a high-dimensional space is suboptimal. High-dimensional representations can be sparse and the computing requirements may cause instability in density-based clustering. To address this, we perform a dimensionality-reduction step prior to clustering.

In this study, we use *Uniform Manifold Approximation and Projection* (UMAP) [59], a dimensionality reduction technique suitable for high-dimensional sentence embeddings, and the default reduction technique in the BERTopic pipeline.

In our study, we reduce the embeddings dimension from 384 to $d=5$ before the clustering step.

We also considered the UMAP neighborhood parameter `n_neighbors` as a sensitivity parameter, together with the number of dimensions and minimum distance. We tested $n_neighbors \in \{15, 30, 50, 100\}$ and chose the final value based on the number of clusters, outlier rate and cluster-size. In our study, we selected `n_neighbors=30`.

UMAP also serves as a denoising step, by projecting the embeddings into a lower-dimensional space. Thus, documents that are close to each other in the original embedding space are expected to remain close in the reduced representation, while some minor local variation is smoothed out. This creates a representation that is more suitable for finding compact and separable topic clusters.

The detailed comparison of the tested configurations is reported in Appendix C.2.

c) Clustering

After dimensionality reduction, the reduced channel-day embeddings are clustered to identify groups of documents discussing similar topics. In the BERTopic pipeline, this step assigns a topic label to each document before computing topic representations with c-TF-IDF.

Clustering is performed using *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN) [60]. HDBSCAN is selected because it does not require the number of clusters to be specified in advance and can identify clusters of varying density. This is particularly important in the Telegram corpus, where some themes are discussed frequently across many channels, while others appear only in smaller and more specialized subsets of the data.

Unlike centroid-based approaches, HDBSCAN is able to assign to low-density documents an outlier class. This behavior is useful for topic modeling on noisy social media data, since not every channel-day necessarily corresponds to a coherent or recurring topic.

Equation 3.12 formally describes the topic assignment vector output of the HDBSCAN.

$$z_i \in \{-1, 0, 1, \dots, k-1\}, \quad (3.12)$$

where z_i denotes the cluster assigned to channel-day document i , k is the number of discovered topics, and $z_i = -1$ indicates that the document was classified as an outlier.

The main HDBSCAN parameters were treated as sensitivity parameters. In particular, different configurations of the minimum cluster size and neighborhood sensitivity were evaluated in order to control the trade-off between topic granularity and cluster stability. In our study, we select `min_cluster_size=300` e `min_samples=15`.

The final clustering configuration was selected through an exploratory parameter sweep. Candidate partitions were compared based on the number of partitions, outlier rate, and cluster-size distribution. The detailed comparison of the tested configurations is reported in the Appendix C.2.

d) Vectorizers and c-TF-IDF

After the HDBSCAN clustering step, each channel-day document is associated with a cluster label. These labels identify groups of semantically similar channel-day observations, but they do not provide an interpretable description of each topic. A separate topic-representation step is therefore required to extract the lexical features that best characterize each cluster.

Before vectorization, we create a representation-specific version of the documents by filtering them from generic placeholders that do not provide useful representative information for topic labeling, such as user mentions, bot commands, URLs, domains, email addresses, phone numbers, file extensions, UUIDs, and generic numeric tokens. Thus, improving the interpretability of the extracted terms. In contrast, security-relevant technical placeholders are retained because they may carry a meaningful cyber-threat context.

The cleaned representation documents are then converted into a sparse bag-of-words matrix using a `CountVectorizer` [35]. In our configuration, we used:

- `ngram_range=(1,2)` to extract both unigrams and bigrams;
- `min_df=50` to reduce noise and avoid unstable lexical features by selecting only terms that appear in at least 50 documents;
- `max_df=0.40` to remove terms that occur in more than 40% of documents. Such terms are unlikely to be topic-specific because they occur across a large fraction of the corpus;
- `max_features=50000` to cap the vocabulary sparse matrix size;
- `stopwordsiso` as multilingual stop-word library. Custom domain stop-words are also introduced.

The topic representations are then computed using C-TF-IDF. This makes it possible to identify words that are frequent within a given topic but distinctive relative to the other topics. C-TF-IDF is mathematically defined by the Equation 3.13.

$$w_{x,c} = \|tf_{x,c}\| \cdot \log \left(1 + \frac{A}{f_x} \right) \quad (3.13)$$

where $tf_{x,c}$ is the frequency of term x in class c , L1-normalized to account for differences in topic size, f_x is the frequency of word x across all classes, and A is the average number of words per class [35]. The resulting score assigns high weight to terms that are common within a topic but uncommon across the rest of the topic space.

In our implementation, the `ClassTfidfTransformer` is configured with the following parameter:

- `bm25_weighting=True` to apply BERTopic’s class-based BM25 variant instead of the default c-TF-IDF weighting scheme, which is intended to make topic representations more robust to frequent terms. The weight scheme is formally described in Equation 3.14 [61].

$$w_{x,c}^{\text{BM25}} = \|tf_{x,c}\| \cdot \log \left(1 + \frac{A - f_x + 0.5}{f_x + 0.5} \right) \quad (3.14)$$

- `reduce_frequent_words=True` is used to further down-weight highly frequent words by reducing their term-frequency contribution after normalization. We applied the square root as Equation 3.15 illustrates.

$$w_{x,c}^{\text{BM25}} = \sqrt{\|tf_{x,c}\|} \cdot \log \left(1 + \frac{A - f_x + 0.5}{f_x + 0.5} \right) \quad (3.15)$$

e) Fine-tuning

As an optional final step, we applied the Maximal Marginal Relevance (MMR) to reduce redundancy among similar keywords and produce more diverse lexical topic descriptions. In our configuration, we set the diversity parameter to 0.3, which provides a trade-off between relevance and diversity.

f) Macro Themes

The output of the BERTopic pipeline is a fine-grained set of topics which well suited for detailed topic interpretation but too granular to described the thematic profile of a channel or a community. Therefore, topics are aggregated into a set of macro-themes by agglomerative clustering of the topics themselves.

Each topic is represented by the centroid of the embeddings associated with the channel-day documents assigned to that topic. The outlier class is ignored from this step. Topic centroids are L2-normalized and clustered using Ward’s agglomerative hierarchical clustering [62, 63] with Euclidean distance.

In our study, macro-themes are derived directly from topic embedding centroids rather than c-TF-IDF topic representations. This allows the aggregation to be based on semantic similarity in the original embedding space.

The resulting dendrogram is cut at a predefined number of clusters k to assign to each fine topic exactly one macro-theme. The value of k is selected through an exploratory scan of candidate solutions, balancing interpretability and cluster size distribution. In particular, values producing excessively large clusters or a high number of singleton themes are discarded. The selected configuration is then used throughout the subsequent analyzes to describe the thematic profiles of channels and communities. In our study, we select $k = 60$.

g) CTI-Focused Topic Models

The need for a second model comes from the nature of the corpus. As noted in the dataset limitation presented in Section 3.3, the corpus includes not only discussions about hacktivism and cyber threats but also large amounts of irrelevant content, such as general news, political commentary, commercial advertisements, and spam. Although this diversity is useful for describing the ecosystem as a whole, it reduces and obscures the thematic resolution of threat-related analysis.

Therefore, a second model is created from the initial list of topics G_0 . Each topic of G_0 is summarized through its c-TF-IDF keywords and a sample representative documents. These summaries are classified by the large language model GPT-4o-mini as CTI-relevant or non-CTI-relevant [64, 65]. Ambiguous topics are manually reviewed before the final inclusion decision. Channel-day embeddings assigned to topics identified as unrelated to CTI are excluded from the new model. Documents assigned to the BERTopic outlier class are excluded from this procedure, as they do not belong to any interpretable topic and therefore their CTI relevance cannot be assessed.

Therefore, BERTopic is applied again using the same configuration, producing a new G_1 topic space specialized for CTI content. Following the methodology described in the previous Section (f) the output of this new model is aggregated into macro-themes by using a $k = 30$. The detailed comparison of the tested configurations is reported in Appendix C.3.

The resulting G_1 topics and their macro-themes are then assigned natural-language labels by the same large language model. Each fine topic is labeled from its top c-TF-IDF keywords together with a small sample of representative channel-day documents. The same process is applied to each G_1 macro-theme from the largest to the smallest. Labels already assigned are provided to the model so that each receives a distinctive name instead of a generic name.

The semantic fidelity of G_1 is verified through an external-validity check against known real-world events. The outcome of this check is reported in Section 4.2.2.

It should be noted that the filtering is performed at the channel-day level rather than at the channel level. Therefore, the same channel may contribute both CTI-related and non-CTI-related documents depending on the content discussed on a given day.

Finally, the resulting G_1 corpus is substantially smaller than the original G_0 corpus because it retains only channel-days associated with CTI-relevant topics. Therefore, not every message, channel-day, or IoC observed in

the full dataset is represented in the G_1 topic space. Analyses that rely on topic assignments therefore operate on a narrower but semantically more focused subset of the corpus, while analyses requiring complete coverage continue to use the broader G_0 .

3.4.3 Community Profiling

After the structural communities have been identified, each community is profiled using three descriptive dimensions: topical composition, language composition, and temporal behavior.

Together, these three dimensions provide an interpretable profile for each structural community: what it talks about, which languages it uses, and how its activity evolves over time.

Community profiling and successive analysis are performed on a restricted common universe formally defined as U in Equation 3.16.

$$U = L \cap B_{G_0}, \quad (3.16)$$

where L is the set of channels carrying a Leiden community label (Section 3.4.1), and B_{G_0} is the set of channels retaining at least one non-outlier channel-day assignment in the BERTopic corpus G_0 (Section 3.4.2). This restriction is necessary because the topic profile may be undefined for channels without non-outlier assignment. The cardinality of U and the coverage in respect to L are reported in Section 4.2.3.

Topic profile. We compute the distribution $P(t \mid c)$ over the channel-day topics of each community. Outliers are excluded from the distribution and reported separately as a per-community outlier rate. Each distribution is summarized by the k dominant topics and by the normalized Shannon entropy of the topic distribution, which differentiates thematically focused communities from thematically diverse ones. In addition to topic frequency, we compute the lift defined in Section 3.2.2 to identify topics that are distinctive rather than simply common in the corpus. Here *Observed* corresponds to $P(t \mid c)$ and the *Expected* corresponds to $P(t \mid \neg c)$ which correspond to the probability that the same topic is outside that community. A lift greater than one means that topic t is more frequent in community c than in the rest of the corpus. Finally, we compute a pairwise similarity between community topic profiles by using Jensen–Shannon distance between their macro-topic distributions. This provides a distribution-level measure of whether communities have similar or distinct thematic profiles beyond individual high-lift topics.

Language profile. Each community is described by the empirical distribution over the detected languages labels of its channels. Communities are summarized by their dominant language and the normalized Shannon entropy of their language distribution, which distinguishes linguistically homogeneous communities from multilingual ones.

Temporal profile. Community activity is described through an hour-of-day fingerprint. Messages associated with each community are aggregated over 24 UTC hours and normalized within each community. This profile serves as a fingerprint for the dominant operational time zone.

3.4.4 Alternative attribute-based partitions

The channel partitions obtained in Section 3.4.1 are compared with a set of alternative partitions originating from individual channel attribute. With these alternative partitions, we test the hypothesis that Leiden communities can be reduced to a single, simpler descriptor. To verify it, we construct three attribute baselines on the universe U defined in Equation 3.16.

- **Language baseline.** We aggregate channels based on their dominant language label, computed in Section 3.3.
- **Topic baseline.** We compute the k -means partition of the per-channel macro-topic distributions $P(m \mid c)$ over the G_0 macro-topics. The number of clusters k is selected by maximizing the silhouette score over a candidate range. Therefore, channels with similar topic distributions are assigned to the same partition. The full report is available in Appendix B.2. In our study $k = 29$.
- **Temporal baseline.** We compute a k -means partition of the per-channel hour-of-day fingerprints. The number of clusters k is selected by maximizing the silhouette score over a candidate range. The full report is available in Appendix B.3. In our study $k = 10$. For each channel, we aggregate and $L1$ -normalize the message volume over the 24 UTC hours to have a probability vector of the activity of the channel. Using the k -means, we cluster channels with a similar vector of activity.

The three baselines are compared with the Leiden partition using the ARI, the NMI, and Cramer’s V defined in Section 3.2.2. ARI and NMI are used to quantify how far each baseline reproduces the structural partition, while Cramer’s V quantifies how strongly the attribute is associated with it.

This comparison assesses whether a simpler clustering on a single attribute would be sufficient to recover the structural communities, or whether the Leiden partition captures organization not reducible to any single descriptor.

3.5 Methods for RQ2: Information Flow and Broker Detection

This section describes the methodology developed of the information-flow layer of TGIF, which answer RQ2. The objective of this layer is to analyze how content spreads across the communities identified in RQ1 and which entities play a key role in that process.

This section is structured as follows. Section 3.5.1 constructs the forwarding graph which will be the input for all the downstream analysis. Section 3.5.2 presents the forwarding ecosystem. Section 3.5.3 analyzes a possible

correlation between topics and messages. Finally, Section 3.5.4 presents an independent content-hash propagation analysis.

3.5.1 Forward Graph and Source Attribution

This section defines the universe of forwarding events analyzed in RQ2. In our study, we used the `forwardFromId`¹⁴ field exposed by the Telegram API. By design, this field exposes only the original author of a message and not the intermediate hops in the case of forwarding. Therefore, it is not possible to reconstruct the original forwarding cascade. This limitation is described in Figure 7 that considers a scenario in which a message published on channel A is forwarded to channels B and C and then re-forwarded from B to channel D.

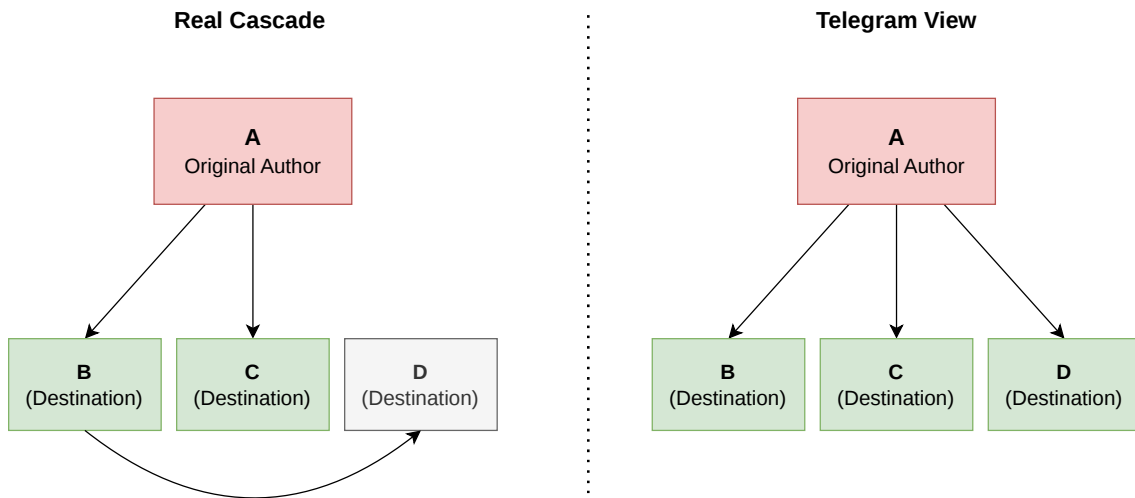


Figure 7: Forwarding visibility limitation

Our analysis focuses on flows of information between communities and therefore self-forwarding are discarded. In more detail, we retain a forwarding event only when the destination channel differs from the declared origin and also when the destination is a labeled venue, so that every event carries a destination community.

The `forwardFromId` field by definition can refer either to a channel or to a user, and for this reason, we assign to each forwarding event one of four source classes:

1. *labeled* when the origin matches a venue labeled (Section 3.4.1);
2. *orphan* when the origin matches an *orphan* channel (Section 3.4.1);
3. *corpus_user* when the origin matches a user;
4. *external* when the origin is not inside the dataset.

¹⁴https://t1.telethon.dev/constructors/message_fwd_header.html

The resulting attributed forward table F serves as the universe for all subsequent analyzes in RQ2 and is formally defined in Equation 3.17.

$$F = \{ e = (s, d, t) : d \in L, s \neq d \}, \quad (3.17)$$

here L is the set of labeled venues (Section 3.4.1), $e = (s, d, t)$ represent a forwarding event where s is the declared origin (`forwardFromId`), d the destination channel (`channelId`) and t its timestamp.

3.5.2 Forwarding Ecosystem Analysis

This section analyzes the forwarding ecosystem F through three different questions: Where do labeled communities get their content, whether the forwards stay within community boundaries, and whether a small set of channels or users concentrates the flow between communities.

This section examines the forwarding ecosystem F along three axes. First, we analyze the sources from which labeled communities draw content. Second, we assess the extent to which forwarding activity remains within community boundaries. Finally, we compute the degree to which inter-community flows are concentrated among a small subset of channels or users.

Community-level forwarding structure The first objective is to characterize how labeled communities receive information. To this end, inbound forwards are aggregated by source class, both globally and per destination community, providing a view of the relative contribution of labeled channels, orphan channels, corpus users, and external sources.

The same forwarding events are then used to analyze whether information exchange remains within the community boundaries identified in RQ1. For this purpose a forward is classified as *intra-community* when source and destination belong to the same community and as *cross-community* otherwise.

Finally, cross-community forwards are examined to determine whether they are concentrated in a small set of channels or users, confirming the presence of broker entities acting as bridges between otherwise distinct communities.

Orphan feeder typology. The objective of the orphan feeders analysis is to determine whether orphan channels act as general-purpose information sources or preferentially feed specific communities. To this end, we identify recurrent orphan feeders, defined as orphan channels that contribute at least k forwards to labeled communities during the observation period.

In our work $k = 10$ is used to separate recurrent feeders from incidental forwarding activity. The value was selected by inspecting the distribution of forwarding counts across orphan channels and confirmed with a robustness analysis of alternative threshold values.

We compute for each feeder f the distribution of its forwarding activity across destination communities using two metrics. The first is the HHI defined in Section 3.2.2, applied here to the shares $n_{f,c}/N_f$, where $n_{f,c}$ is the number of forwards from f to community c and $N_f = \sum_c n_{f,c}$ is its total forwarding volume. The second is $pct_top(f)$ which measures the activity directed to the feeder's most targeted community. Values of both measures close to one mean that a feeder concentrates almost all of its activity on one community, whereas lower values indicate a more diversified targeting pattern.

3.5.3 Topic-Driven Sourcing

In this section, we analyze whether forwarding activity follows a topical logic. If content is shared between channels discussing similar topics, forwarded messages should connect source and destination channels associated with the same macro-topic more often than expected by chance.

To this end, each forwarding event is enriched with the macro-topic assignments derived from G_1 (Section 3.4.2). For every forwarding event, a macro-topic is assigned based on the origin and destination channel-day. The date of the original message is exposed by the field `forward_date_dt`¹⁵. Only events in which both endpoints contain a valid G_1 label are analyzed. Since G_1 already restricts the corpus to content relevant to the CTI, this requirement results in a loss of coverage compared to the total number of forwardings.

It should be noted that dominant topics occur with higher frequencies. As a result, some level of topic concordance is expected even in random forwarding. Therefore, we evaluate the topic concordance as a lift over a chance baseline.

For the forwarding analysis, the generic lift defined in Section 3.2.2 is applied to source–destination topic alignment. In this context, the *Observed* quantity is the proportion of forwarding events whose source and destination share the same macro-topic, and the *Expected* is estimated through a permutation-based null model. For each forwarding regime (intra-community, orphan-orphan, orphan-labeled, labeled-orphan, cross-community), destination topics are randomly shuffled 1,000 times, preserving the overall topic frequencies while breaking the original source–destination associations.

A *lift* value greater than one indicates that forwarding events connect channels sharing the same topic more often than expected by chance, indicating that content flows preferentially between channels interested in the same topic.

3.5.4 Content-Hash Propagation

Following the methodology of Zannettou et al. [41], the same file hash appearing in multiple channels is treated as evidence that a content artifact has been reused or redistributed. The objective is to observe how CTI-related

¹⁵https://tl.telethon.dev/constructors/message_fwd_header.html

content moves across the communities identified in Section 3.4.1.

Since the forwarding metadata exposes only the original author and not the intermediate hops, the forward cascade cannot be reconstructed from individual edges (Figure 7). In this analysis, we track the appearance of the same hash in forwarding events across channels over time to recover its propagation footprint.

It should be mentioned that a file appearing in multiple channels is not necessarily evidence of propagation, since publicly available malware samples, leaked archives, or other artifacts may be posted independently by unrelated actors. For this reason, our analysis is restricted to *forward-verified* MD5 hash extracted in Section 3.3. Thus, an MD5 hash is retained only if it is carried by at least one observed forward and if it appears in at least two distinct channels.

In our study, we group forwarding events according to their forwarding origin. Each forwarding event carrying the same MD5 hash generates a directed propagation edge from the forwarding source channel to the destination channel. The same MD5 hash may therefore contribute to multiple propagation events when observed with different forwarding origins. Each propagation edge is labeled using the community assignment of its source and destination channels, allowing events to be classified according to their flow direction.

The propagation graph is used to characterize how CTI-related artifacts circulate throughout the ecosystem. In particular, we analyze the distribution of propagation events across the different flow categories and identify the artifacts that exhibit the largest diffusion footprint.

3.6 Methods for RQ3: IoC Extraction and Community Patterns

This section describes the methodology of the indicator layer TGIF, which answers to RQ3. The objective of this layer is to analyze how IoCs are distributed across the Telegram ecosystem identified in RQ1.

Instead of analyzing IoCs as a single independent unit, our study contextualizes each indicator within the structural communities, topical profiles, and broker channels previously identified. This enables the investigation of whether threat indicators are concentrated in specific communities of the ecosystem and whether some structural or semantic contexts are associated with earlier IoC appearance.

This section is structured as follows. Section 3.6.1 examines how IoCs are distributed between structural communities and topical groups. Finally, Section 3.6.2 analyzes temporal patterns of IoC appearance, including topic-IoC lead-lag relationships and the relative timing of indicator observations across Telegram and external intelligence sources.

3.6.1 IoC Distribution Analysis

The objective of this section is to determine whether IoCs are distributed uniformly across communities and topics or concentrated in a small structural or semantic minority.

The community analysis is restricted to the universe L defined in Section 3.4.3. Orphan channels are excluded from this view because they do not carry a community label. The topic analysis is restricted to the universe of G_1 topics which is defined in Section 3.4.2.

a) Community-Level Distribution

The objective of this analysis is to determine whether malicious indicators are distributed uniformly across communities or concentrated in a small subset of the ecosystem.

To this end, each loC mention inherits the community label of the channel in which it appears. This allows the same indicator to contribute to multiple communities when it is observed in different parts of the ecosystem.

Indicators are aggregated by community using the nested tiers defined in Section 3.3 (*extracted*, *VT-present*, *confirmed-malicious*). The concentration of malicious indicators across communities is quantified using the Gini coefficient defined in Section 3.2.2 and the cumulative share held by the top- k communities.

It should be noted that a community can produce many indicators simply because it produces more indicators overall. To account for differences in baseline activity, we applied a chi-square goodness-of-fit test described in Section 3.2.2, where O_c is the observed number of confirmed-malicious indicators in the community c and E_c is the expected count under the extracted-volume baseline. This test provides a global assessment of whether the production of malicious indicators is more uneven than would be expected from the overall activity alone. In addition, the main contributing communities are characterized using lift, where *Observed* represent their share of malicious indicators, while *Expected* represent their expected share under the same baseline.

Finally, to assess whether communities share the same malicious indicators or whether each contribute to a separate set, each malicious indicator is assigned to the set of communities in which it appears. An indicator observed in more than one community is classified as cross-community and one observed in a single community as community-exclusive. Malicious indicators are also grouped by loC type to characterize the indicator profile of each community.

b) Topic-Level Distribution

The objective of this section is to determine whether the appearance of malicious indicators is uniformly distributed across thematic areas or concentrated in a specific set of topics.

As mentioned in the introduction, this analysis is restricted to the G_1 topics set introduced in Section 3.4.2, which refers to topics related to CTI. Each loC mention inherits the macro-topic of the channel-day in which it appears. Malicious-indicator mentions are aggregated at the macro-topic level using the G_1 macro-topics partition described in Section 3.4.2.

As in the previous Section, the concentration of malicious indicators across macro-topics is quantified using the

Gini coefficient and the cumulative share held by the top- k macro-topics, both defined in Section 3.2.2. To assess whether this concentration exceeds what topic exposure alone would predict, the observed distribution is tested with the chi-square goodness-of-fit test defined in Section 3.2.2, taking as expected the count for each macro-topic its share of extracted indicators among those carrying a G_1 assignment. Finally, lift is used to characterize macro-topic that over-produce malicious indicators relative to their overall indicator activity.

By construction, the G_1 corpus contains only channel-days associated with CTI-relevant topics and therefore does not provide complete coverage of the original G_0 corpus. As a result, not all malicious indicators can be linked to a G_1 topic assignment. Instead of discarding these indicators, we retain them and characterize their context using the larger topic space G_0 . Indicators outside G_1 are classified either as BERTopic outliers or as members of non-CTI thematic domains. This complementary view allows the analysis to distinguish between indicators associated with CTI-focused discussions and those appearing in other parts of the Telegram ecosystem.

3.6.2 Temporal Analysis

This section analyzes the temporal behavior of malicious indicators. This section is structured as follows. Section (a) examines whether bursts of thematic activity precede or follow the production of indicators within the same macro-topic. Section (b) compares, for each indicator, its first appearance in the corpus with its first submission to VirusTotal.

a) Topic–IoC Lead–Lag Analysis

To examine whether spikes in thematic activity precede, follow or coincide with the production of IoC, we built two daily series for each macro-theme.

1. The *topic-activity* series, which refer to the number of G_1 channel-days assigned to the macro-topic on day t .
2. The *indicator-production* series, which refer to the number of malicious-indicator mentions carrying that macro-topic on day t .

Since low-volume topics do not provide enough active days for a stable estimation the analysis is limited to the macro-topics with the highest indicator volume and channel-days.

As mentioned the limitation in presented in Section 3.3, the corpus grows over time as the channels are progressively monitored. Thus, raw daily counts may increase simply because more channels are observed and not for any real signal. To account for this, we *exposure-normalized* each series by dividing daily counts by the number of channels onboarded by day t .

Furthermore, the resulting series are smoothed using a centered seven-day moving average. This reduces day-level noise while preserving broader weekly patterns. Finally, both series are standardized using z-score normalization which ensures that similarities between the two series are captured.

We used a cross-correlation function to measure the relationship between the two series. For each temporal delay δ in the range $[-14, +14]$, we computed the Pearson correlation between topic activity on day t and indicator production on day $t + \delta$. By progressively shifting one series relative to the other and recomputing the correlation at each offset, the procedure reveals whether activity in one series consistently anticipates the other.

Therefore, a positive peak lag indicates that topic activity tends to precede indicator production, a negative peak lag indicates the reverse relationship, and a peak at $\delta = 0$ indicates temporal synchrony.

It should be noted that the peak lag reflects a consistent average trend rather than the behavior of any single burst. Single events timing are examined separately in the next Section.

b) First-Seen Analysis

This analysis examines single indicator events that first appear in the corpus against when it was first submitted to VirusTotal. For each indicator, the corpus first-seen date is the earliest channel-day in which it is observed, and the VirusTotal first-seen date is taken from the API report first-submission timestamp. Only indicators with both dates are retained.

For each indicator, the difference $\Delta = \text{first-seen}_{\text{corpus}} - \text{first-seen}_{\text{VT}}$ is computed in days. A negative Δ means that the indicator was present in the corpus before being submitted to VirusTotal, a positive Δ the opposite, while a zero the same day. Indicators whose VirusTotal date precedes the corpus start are excluded from this analysis.

We examine the subset of indicators with $\Delta \leq 0$ to provide a contextual view of early providers. Therefore, each indicator is enriched with the type, the channel, and community in which it first appears, and where available, its associated G_1 topic label.

4 Results & Discussion

This chapter presents the empirical results of the Telegram Community Intelligence Framework. The chapter is organized according to the analytical pipeline described in Chapter 3. Section 4.1 reports the composition of the final analytical corpus after preprocessing and filtering. Section 4.2 presents the results for RQ1, combining user co-affiliation community detection with BERTopic-based topical profiling. Sections 4.3 and 4.4 then analyze the information flow, the broker channels, and the loC patterns.

4.1 Dataset Overview and Preprocessing Outcomes

This section reports the composition of the corpus after the initial dataset restriction and preprocessing stages. The goal is to describe the analytical universe used in the subsequent graph, topic modeling, information flow, and loC analyses. The section first describes the temporal coverage and corpus size, then analyzes the language composition of the retained data, and finally describes the distribution of activity across Telegram channels.

4.1.1 Corpus Size and Temporal Coverage

The composition of the corpus before and after the data cleaning pipeline presented in Section 3.3 is described in Table 4.1.

Table 4.1 Corpus composition before and after the general cleaning pipeline

Metric	Raw corpus	After cleaning	Δ
Messages	52,973,413	45,658,188	−13.8%
Channels	6,240	6,223	−0.27%
Unique senders	612,286	596,994	−2.5%

The filtered records mainly consist of messages with empty content after removing emojis or null-content messages without any attachment. Among the retained messages, 1,227,621 (2.69%) come from 446 bot-flagged accounts. Although these accounts represent only 0.07% of unique senders they present a high posting volume, characteristic of automated accounts and spam on Telegram.

Our inspection of the cleaned corpus shows that the activity is highly concentrated at both the user and channel level. As shown in Figure 8a, the distribution of messages per sender is strongly right-skewed, with 65.4% of accounts sent fewer than five messages and the median sender contributed only two. A small group of hyperactive users accounts for a disproportionate share of total message volume. Similarly Figure 8b describes concentration at the channel level in which a small number of high-volume channels dominate the message volume, and the median channel has a single active sender, and only 10.3% of channels have at least three distinct non-bot senders

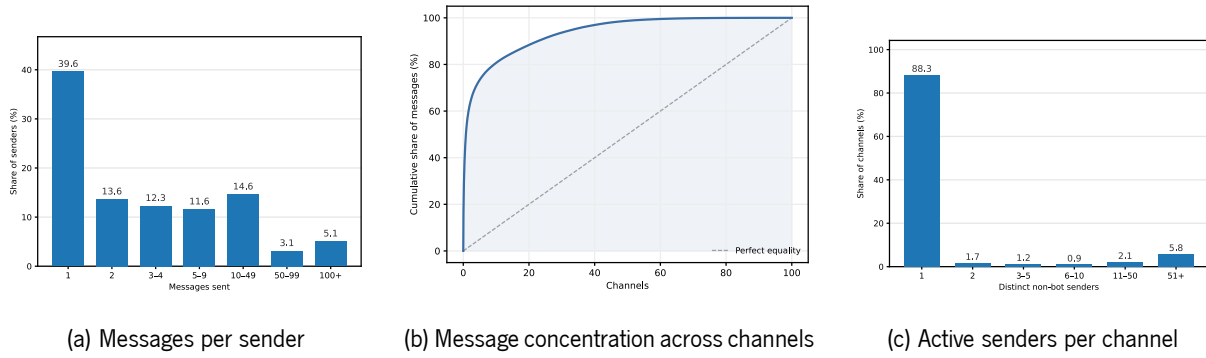


Figure 8: Corpus insights graphs

(Figure 8c).

These results are the reason for the thresholds and the logic applied in constructing the co-affiliation graph in Section 3.4.1.

4.1.2 Language Distribution

The language-detection procedure described in Section 3.3 assigns one dominant language label at the channel level. In total, 6,115 channels received a language label, and 33 distinct languages were detected. Only 108 channels for which the classifier did not have sufficient textual evidence were instead assigned to the `unknown` category.

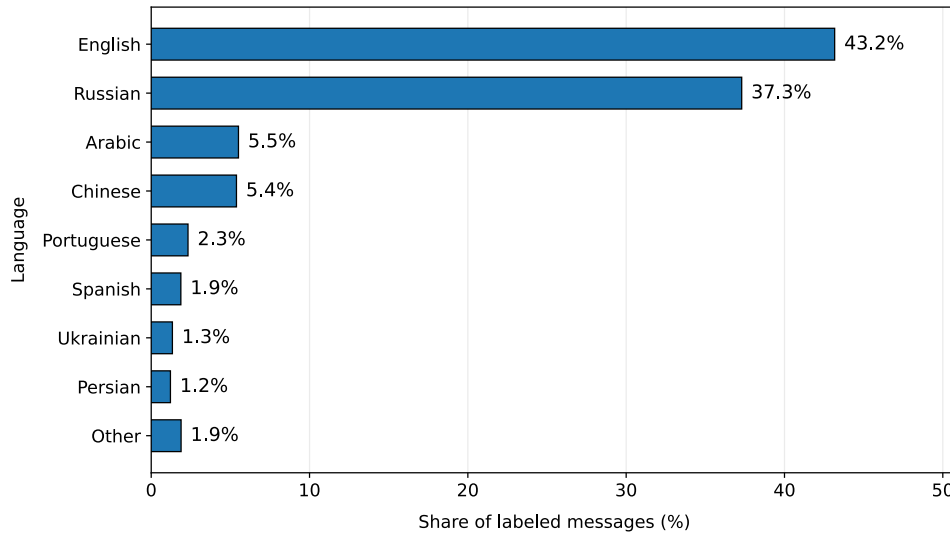


Figure 9: Language Distribution

Figure 9 describes the resulting distribution. The corpus is dominated by English (43.2% of message volume) and Russian (37.3%), which together account for approximately 80.5% of labeled messages. Arabic and Chinese form a second layer at 5.5% and 5.5%, respectively, while the remaining languages constitute a long tail.

Therefore, the corpus is multilingual but highly skewed. For this reason, it is necessary to ensure multilingual support in the topic modeling pipeline, as limiting the corpus to English would mean excluding more than half of the monitored ecosystem.

4.1.3 BERTopic Preprocessing Outcomes

Table 4.2 presents the outcome of each BERTopic-specific preprocessing stage. As described in Section 3.3, three filters are applied sequentially. The aggregation of the surviving messages produces 1,338,879 channel-day documents from 45.7M messages, which represents a compression of approximately 34 times. Furthermore, 108 channels do not contain any valid channel-day and are excluded at this stage.

Table 4.2 Corpus size at each stage of BERTopic-specific preprocessing

Stage	Unit	Records	Channels	Languages
After general cleaning	Messages	45,658,188	6,223	33
Channel-day aggregation	Channel-days	1,338,879	6,115	33
Token threshold (≥ 15 real tokens)	Channel-days	1,298,863	—	—
Language-mass threshold ($\geq 2,500$ cd)	Channel-days	1,273,289	5,775	10

The channel-day token threshold maintains 97.0% of documents, with a median document contains 382 real tokens (mean: 854), confirming that the filter affects only the sparsest tail of the distribution.

The language-mass threshold maintains ten languages: Russian, English, Arabic, Spanish, Ukrainian, Chinese, Persian, Vietnamese, Malay, and Hebrew. All ten together account for 98.7% of all channel-day records. Channels in the `unknown` language category (Section 4.1.2) are excluded at this stage, as they cannot satisfy a language-level threshold. Combined, the two channel-day-level filters retain 95.1% of documents.

4.2 RQ1: Community Structure and Topical Profiling

This section reports the results of the first layer of the framework. Section 4.2.1 presents the community-detection output and its robustness. Section 4.2.2 presents the topic space, while Section 4.2.3 characterizes the detected communities along their topical, linguistic, and temporal dimensions. Section 4.2.4 compares the structural partition reduces to any single channel attribute.

It should be highlighted that all profiling and cross-partition analysis operate on the universe $U = L \cap B_{G_0}$ defined in Equation 3.16, which is defined by the 620 channels that carry both a community label and at least one non-outlier topic assignment.

4.2.1 Structural Communities

This section presents the output of each stage of the community detection process described in Section 3.4.1. Table 4.3 reports the size and output of the corpus for each construction step.

Table 4.3 Size of the RQ1 structural corpus across the main graph-construction stages

Stage	Unit	Users	Venues	Edges / links
Cleaned corpus	Telegram entities	—	6,223	—
Venue selection ($k \geq 3$)	venues	—	618	—
Bipartite affiliation matrix	user–venue	591,126	618	630,670
Filtered bipartite matrix	user–venue	30,534	468	69,951
Full one-mode projection	user–user	30,534	—	50,062,869
Sparsified projection	user–user	24,335	—	2,556,476
Leiden partition ($\gamma = 0.6$)	communities	24,335	—	315

The first filtering step reduces the corpus from 6,223 Telegram entities to 618 venues satisfying the $k \geq 3$ non-bot-poster threshold. Although this corresponds to only 9.9% of entities, these venues account for 55.4% of the cleaned message volume. The venue filter therefore removes a large number of low-signal entities while preserving more than half of all observed activity.

The user–venue matrix obtained from these 618 venues contains 591,126 non-bot users and 630,670 affiliations. At this stage, the median user posts in exactly one venue, confirming that most users provide only local participation evidence. Filtering the bipartite matrix reduces it to 30,534 users, 468 venues, and 69,951 affiliations, retaining 5.2% of users. After filtering, the median user posts in two venues, while venues contain a median of 32 active users.

We then projected this filtered matrix onto a user–user graph, resulting in 30,534 nodes and 50,062,869 edges. This raw projection is too dense to be directly interpretable, with a median degree of 1,338. Therefore, through sparsification the graph retains 24,335 users, corresponding to 79.7% of the projected nodes, but only 2,556,476 edges, corresponding to 5.11% of the projected edges. The median degree drops from 1,338 to 105. Despite edge reduction, the graph remains structurally connected, where the largest connected component contains 21,949 users, covering 90.2% of the retained graph. This indicates that sparsification removes a large volume of low-strength links while preserving the core multi-venue user backbone.

The Leiden community detection algorithm is then applied to the sparsified graph with $\gamma = 0.6$. The resulting partition contains 315 user-level communities, no singletons, and the largest community contains 2,862 users, corresponding to 11.8% of the graph. It should be noted that this partition does not contain extreme fragmentation and collapse into a single dominant cluster, which are two common risks in dense co-affiliation projections.

Since the rest of the study operates at the venue level, the user communities are propagated back to channels through the user-community composition of each venue as described in Section i). After peripheral user re-attachment, 98.9% of users carry a community label. Therefore, by projecting these labels onto channels returns 720 labeled channels divided in 692 single-community channels and 28 multi-community channels. The remaining 5,503 channels are left as orphan.

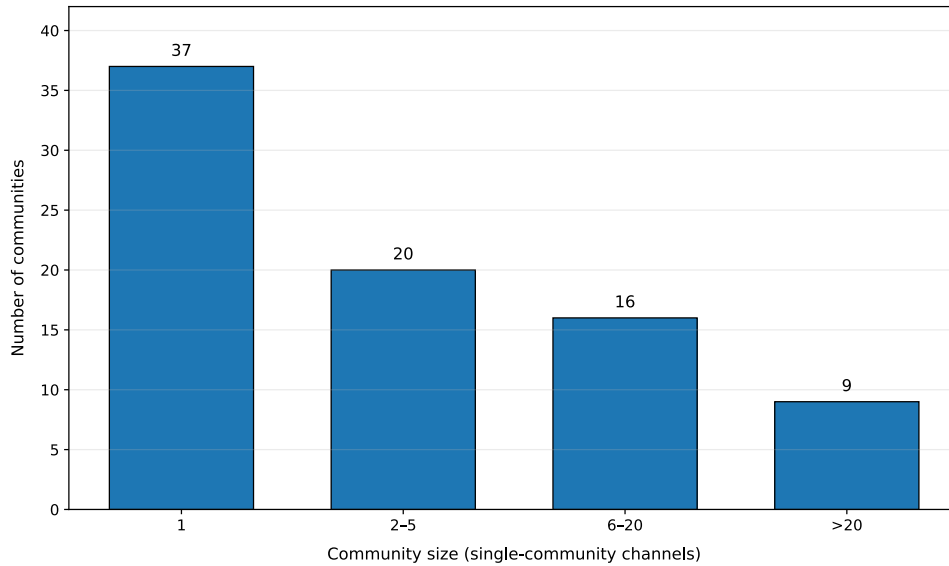


Figure 10: Community size distribution

Figure 10 reports the resulting community size distribution. Each of the 720 labeled channels is counted under its dominant community, resulting in 82 communities with a median size of two channels and a mean of 8.65. The gap between mean and median is a direct consequence of the skewed distribution. It should be noted that the presence of single-channel communities is a direct consequence of the propagation process of Section 3.4.1. When these user-level communities are mapped onto channels, a community whose users concentrate their activity in a single labeled channel appears as a single-channel community, even though the underlying user group may be large.

These results show that the hacktivist Telegram ecosystem contains a stable structural layer below the level of individual channels. The filtering and sparsification steps remove local or weak co-participation evidence, but they do not erase the multi-venue backbone. The Leiden algorithm identifies many communities of significant size, with no single group dominating the partition. The next paragraph assesses the robustness of this structure.

Robustness. To assess whether the detected communities reflect the affiliation structure of the corpus rather than an artifact of the selected algorithm, the Leiden partition is compared with three alternative community detection methods applied to the same sparsified graph. The results are reported in Table 4.4.

Louvain provides the closest comparison because it belongs to the same algorithmic family as Leiden. Louvain returns 311 communities, nearly the same largest-community share as Leiden, and no singletons. The agreement

Table 4.4 Robustness comparison between the selected Leiden partition and alternative community detection algorithms

Algorithm	Role	Communities	Largest	ARI	NMI	H	C
Leiden	Reference partition	315	11.8%	—	—	—	—
Louvain	Modularity baseline	311	11.8%	0.864	0.937	0.938	0.936
Infomap	Flow compression	674	9.9%	0.553	0.801	0.985	0.676
Label Propagation	Local diffusion	708	9.4%	0.525	0.796	0.977	0.672

with the Leiden partition is high, with $ARI = 0.864$ and $NMI = 0.937$. The corresponding homogeneity and completeness scores are also balanced, at 0.938 and 0.936, indicating that the two partitions have similar granularity. This suggests that the selected structure does not depend only on the refinement step introduced by Leiden over Louvain.

Infomap and Label Propagation produce smaller partitions. Infomap identifies 674 communities, while Label Propagation identifies 708 communities. Both alternatives presents no singletons and largest-community shares below 10%. Their ARI scores against Leiden are moderate, respectively 0.553 and 0.525, while their NMI scores remain high, at 0.801 and 0.796. This combination is consistent with a difference in resolution rather than a reorganization of the boundaries of the community.

Furthermore, the homogeneity and completeness values confirms this interpretation. Infomap reaches high homogeneity (0.985) but lower completeness (0.676), while Label Propagation shows the same asymmetry, with an homogeneity of 0.977 and a completeness of 0.672. High homogeneity indicates that the alternative clusters are mostly contained within Leiden communities while lower completeness indicates that Leiden communities are often split across several alternative clusters. This means that these algorithm mainly subdivide the Leiden partition rather than merging unrelated Leiden communities.

In conclusion, Louvain confirms the same modularity-based organization, while Infomap and Label Propagation recover finer-grained versions of that structure. The main disagreement is related to granularity rather than a contradiction of the community boundaries. Therefore, the robustness analysis supports the use of the Leiden partition as a structural reference for the remainder of the thesis.

4.2.2 Topic Space

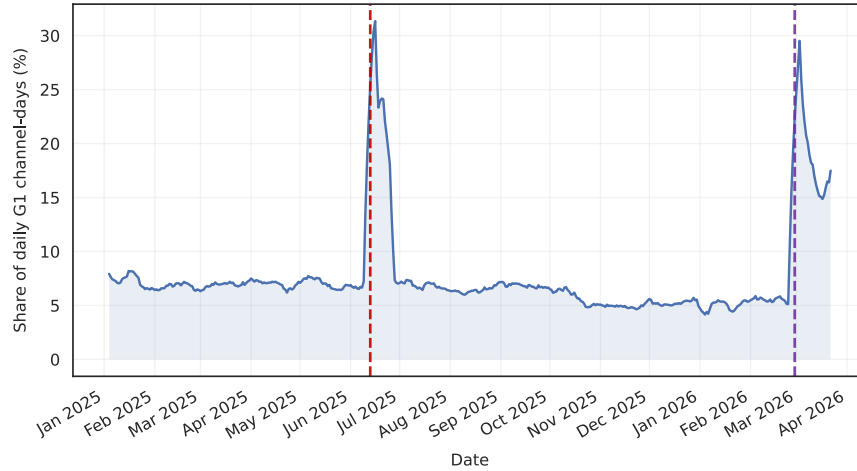
This section presents the output of the Topic Modeling Process described in Section 3.4.2. Two topic spaces are built from the channel-day corpus described in Section 3.3. The first, G_0 , captures the entire observed ecosystem and is used for community profiling in RQ1. The second, G_1 , represent the CTI-relevant subset of G_0 and is used in threat-specific analyzes of RQ2 and RQ3. Their characteristics are summarized in Table 4.5.

Table 4.5 The two BERTopic topic spaces G_0 and G_1

Topic space	Channel-days	Fine topics	Macro-topics	Outlier rate
G_0 full corpus	1,273,289	809	60	47.9%
G_1 CTI-filtered	192,190	223	30	5.8%

G_0 spans 1,273,289 channel-days and produce 809 fine-grained topics, which are then aggregated into 60 macro-topics. This first model assigned 47.9% of channel-days to the outlier class. As already mentioned in the dataset limitation presented in Section 3.3, the corpus contains not only threat-relevant discussion, but also local news, geopolitical commentary, spam and entertainment, which do not form recurring topical clusters. Therefore, this high outlier share derives from the corpus-composition itself.

G_1 retains 192,190 channel-days (15.1% of G_0) and produces 223 fine-grained topics, which are then aggregated into 30 macro-topics. G_1 present an outlier rate of 5.8%. The difference in the outlier share originates from the domain filter, which means that after cleaning almost all remaining documents are assigned to stable recurring topics.

Figure 11: Daily share of the *Middle Eastern Conflict and Activism* macro-topic within G_1

To assess whether the topics are capturing real content rather than lexical artifact, we conducted validity checks against known real-world events. Figure 11 reports the daily share of the *Middle East Conflict and Activism* macro-topic of G_1 , normalized by the total daily volume and smoothed using a seven-day mean. The series remains close to a 5 – 7% baseline during most of the observation window, but rises sharply around both conflict onsets marked in the figure. Around the start of the Twelve-Day War on 13 June 2025¹⁶, the mean share increases from 6.8% to

¹⁶Reuters, 13 June 2025

19.2%, which represents a 2.8 times increase. Around the February 2026 escalation¹⁷, it increases from 5.4% to 20.9%, which represents a 3.9 times increase.

This check does not aim to establish causality between the external events and Telegram activity, instead, it verifies that the topic space is sensitive to interpretable real-world shifts in the corpus. Furthermore, it indicates that G_1 captures real-world event-driven content, supporting its use as a thematic vocabulary for downstream analyzes of RQ2 and RQ3.

4.2.3 Community Profiles

This section presents the output of the Community Profiling Process described in Section 3.4.2. We profile structural communities identified in Section 4.2.1 on the three descriptive dimensions: topical composition, language composition, and temporal behavior.

The set of channels that carry both a Leiden community label and at least one non-outlier channel-day in the G_0 topic space is defined by Equation 3.16. Table 4.6 presents its coverage and composition. All subsequent profiling and cross-partition analyzes operate in the 72 communities of U .

Table 4.6 Channel and community coverage at each stage leading to the analysis universes U (G_0) and U_1 (G_1)

Set	Definition	Channels	Communities
Cleaned corpus	all retained channels	6,223	—
L	channels with a Leiden community label	720	82
B_{G_0}	channels with ≥ 1 non-outlier G_0 day	5,775	—
$U = L \cap B_{G_0}$	analysis universe (G_0)	620	72
B_{G_1}	channels with ≥ 1 non-outlier G_1 day	3,658	—
$U_1 = L \cap B_{G_1}$	CTI analysis universe (G_1)	540	67

Topical composition. This profile evaluates whether the structural communities detected from user co-affiliation also correspond to distinct thematic environments.

First, we identify which fine-grained topics are especially characteristic of each community using lift. This shows whether a topic is simply common in the corpus or concentrated in one community.

The lift results describe a strong topical concentration. In the G_0 topic space, the most distinctive community–topic associations are community-exclusive, meaning that the topic is observed only within one community after filtering, and the lift is therefore formally unbounded. Among non-exclusive cases, the median lift is 46.8, and the 90th percentile reaches 1,438.8. The CTI-focused G_1 topic space presents the same pattern with the median lift

¹⁷Reuters, 28 February 2026

at 19.1, with a 90th percentile of 762.7. These values indicate that the most distinctive topics are not only frequent in absolute terms but are strongly concentrated within specific communities.

Second, we compare the complete macro-topic distributions of all communities using Jensen–Shannon distance. This tests whether communities differ only by a few distinctive topics or whether their overall thematic profiles are separated. In the $U(G_0)$ universe, the mean off-diagonal distance between the 72 community profiles is 0.886. The same pattern remains in the $U_1(G_1)$ universe, where the mean off-diagonal distance remains high at 0.861. This indicates that most communities have substantially different thematic compositions, and that this separation persists after off-domain activity is removed.

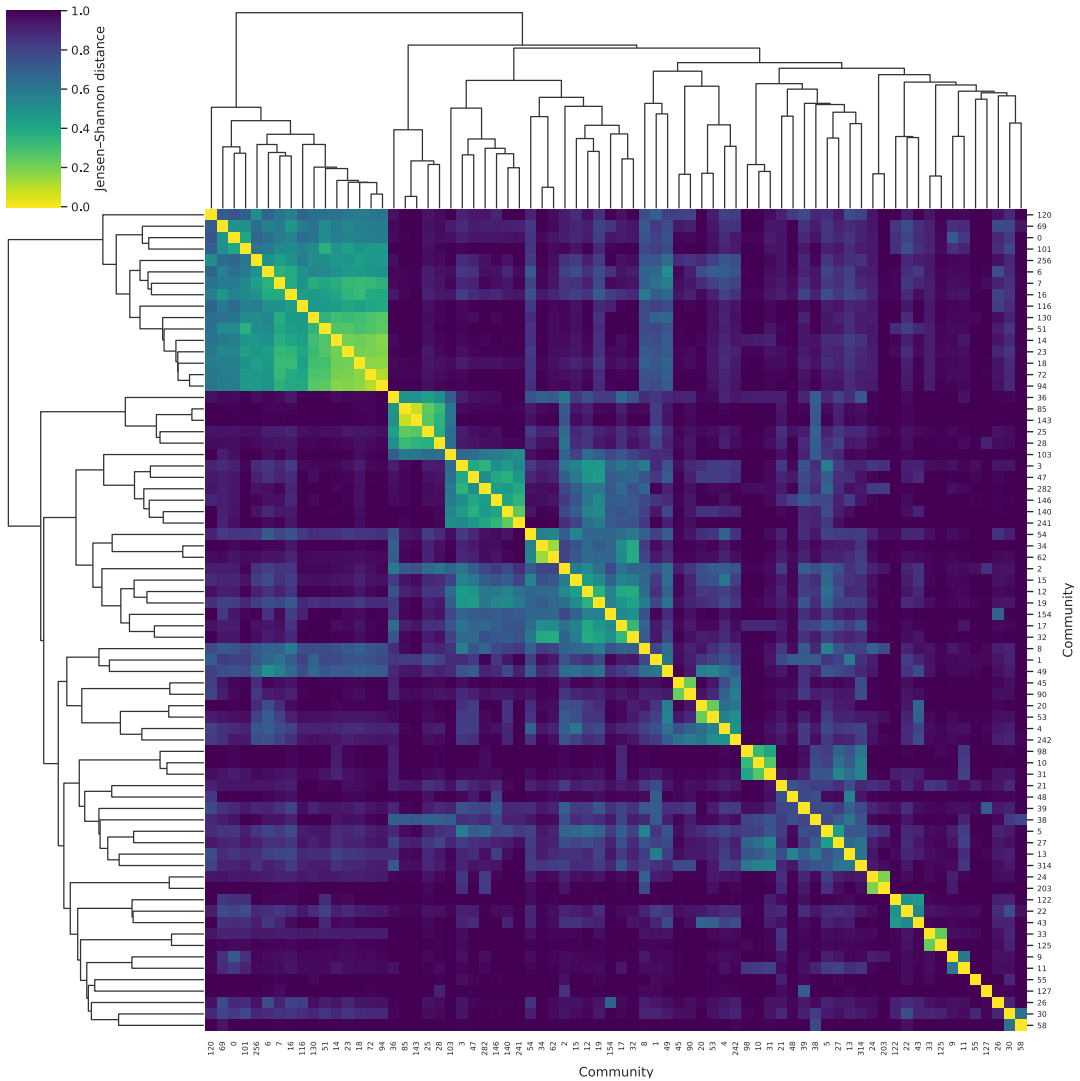


Figure 12: Pairwise Jensen–Shannon distance between the G_0 macro-topic profiles of the 72 venue-level communities

Figure 12 shows the full distance matrix. The matrix is dominated by high-distance values that confirm thematic separation. However, the figure also reveals several lower-distance blocks along the diagonal, which indicates a subset of communities that share similar macro-topic profiles.

In conclusion, this topic composition shows that the ecosystem is fragmented into different thematic environments, and this fragmentation is not completely random, as some communities form smaller families of related topical profiles.

Language composition. The language profile provides a second layer of interpretation for the structural communities. All 620 channels in U have a detected language label, and the universe spans ten languages. At the community level, monolingualism is common with 69% of communities present a single language only. However, this pattern is strongly dependent on the size of the community. As Figure 13 shows, all single-channel communities are monolingual by construction, since a single-channel community cannot hold more than one language. The share of monolingual communities then decreases with the increase of the channel size. Monolingualism is therefore mostly a property of small communities rather than of the ecosystem as a whole.

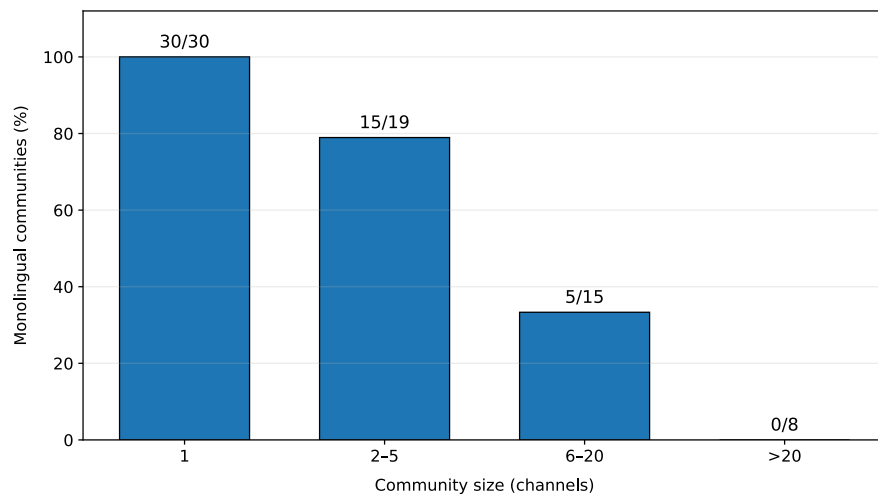


Figure 13: Share of monolingual communities by community size in U

The larger communities instead span several languages, but they are not balanced multilingual mixtures. Their language profiles usually retain a single clear dominant language, accompanied by smaller minority-language shares. Figure 14 reports these patterns for the 20 largest communities. The figure shows that in most of the cases one language accounts for the majority channels, while additional languages appear as smaller segments.

In conclusion, language helps to interpret the identity of the community, but does not fully explain structural partition. Linguistic homogeneity reinforces the boundaries of many small communities, whereas larger communities remain structurally cohesive despite spanning several languages. This suggests that co-affiliation patterns are also shaped by topical, operational, or audience-based factors beyond language alone.

Temporal behaviour. The objective of this layer is to capture when they are active. For each community, messages are aggregated over 24 UTC hours and normalized within the community, producing an hour-of-day

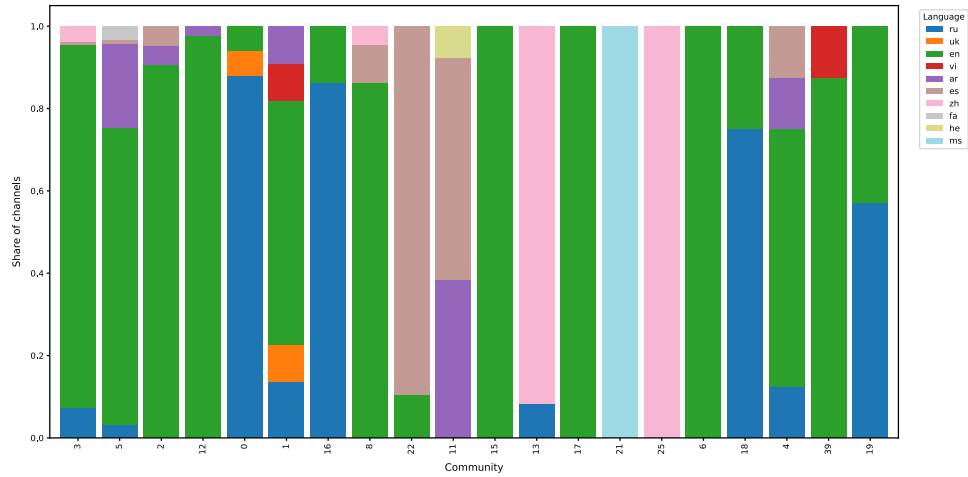


Figure 14: Language composition of top-20 communities by size

fingerprint that describes the share of activity occurring in each UTC hour.

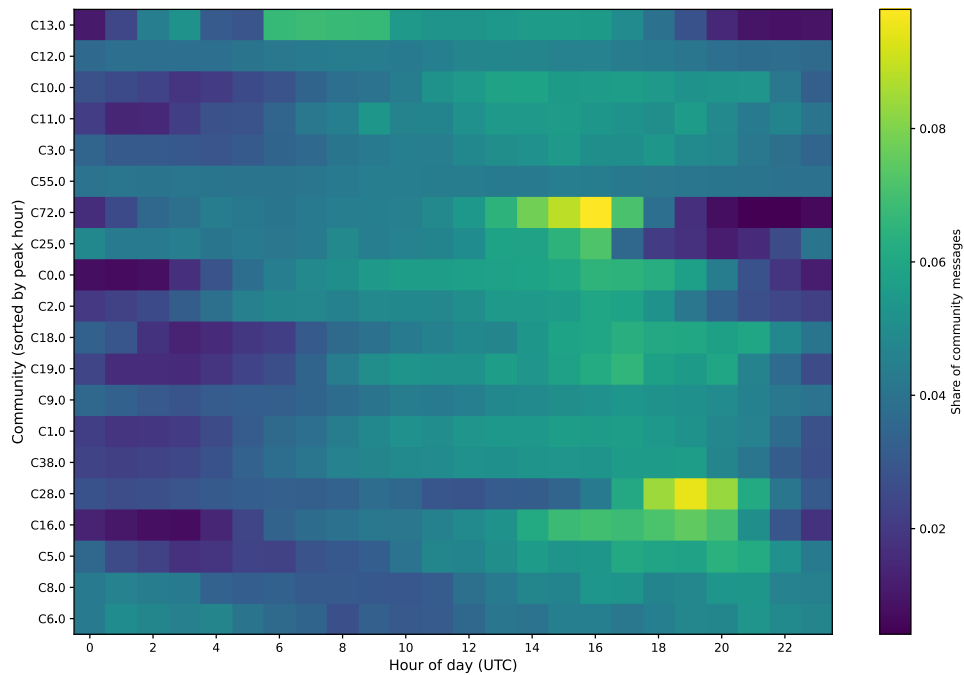


Figure 15: Hour-of-day fingerprints for the 20 highest-volume communities

Figure 15 reports the fingerprints of the 20 highest-volume communities, ordered by peak activity hour. The heatmap shows that communities differ in their daily rhythm. In some cases, activity is concentrated around clear and specific UTC windows, while in others the activity remain a flatter distribution throughout the day. This suggests that structural communities are differentiated not only by topic and language, but also by temporal activity patterns.

4.2.4 Alternative Attribute-Based Partitions

This section presents the results obtained by the methodology described in Section 3.4.4. The outcomes are summarized in Table 4.7.

Table 4.7 Comparison between the Leiden structural partition and attribute-based baseline partitions on U

Baseline	Groups	ARI	NMI	Cramer's V
Language	10	0.121	0.401	0.737
Topic	29	0.097	0.402	0.487
Temporal	10	0.016	0.161	0.400

The structural partition contains 72 communities, while the attribute baselines contain 10 language groups, 29 topic clusters, and 10 temporal clusters. ARI and NMI are sensitive to this mismatch in resolution and are therefore used to assess whether an attribute-based partition reconstructs the Leiden partition.

All three baselines have low ARI values (≤ 0.121), indicating that none of them reproduces the structural partition. The NMI values provide a less strict view of the same comparison, where all three baselines share a moderate amount of information with the Leiden partition. This shows that the attribute baselines are not independent of the structural partition, even though none recovers it. Therefore, Cramer's V is used to compare how strongly each attribute is associated with the Leiden communities.

The language partition shows the strongest association (Cramer's $V = 0.737$, $NMI = 0.401$), confirming that linguistic structure is closely related to community structure, consistent with the results of the language-profile in Section 4.2.3. However, the low ARI shows that language alone does not recover the partition.

The topic partition presents an intermediate association (Cramer's $V = 0.487$, $NMI = 0.402$) where the thematic composition is correlated with the co-affiliation structure, but does not define it.

The temporal partition shows the weakest association (Cramer's $V = 0.400$, $NMI = 0.161$, $ARI = 0.016$), contributing some information but less than the other two attributes.

Overall, the Leiden partition captures a multi-dimensional form of organization. No single attribute reconstructs it, as shown by the low values of ARI. This supports interpreting the partition as a genuine co-affiliation structure rather than as an artifact of a single channel attribute.

4.2.5 RQ1 Answer

This section answers the first research question of this thesis: *How are Telegram users organized into communities across hacktivist channels, and what topics characterize those user communities?*

Our analysis reveals that the user co-participation across the hacktivist Telegram ecosystem presents a non-trivial latent community structure. The output of our community detection is 82 different channel level communities,

which become 72 by restricting the analysis to the profiling universe U . Robustness checks show that alternative algorithms either recover a very similar modular structure or subdivide the selected Leiden partition, rather than contradicting it.

Moreover, these communities are not only a structural grouping, but also interpretable. Section 4.2.2 shows that communities are topically distinguishable in both the full G_0 topic space and the CTI-focused G_1 space. They also differ in language composition and daily activity rhythm as presented in the result of Section 3.4.3, although these attributes do not fully explain the partition.

Finally, the results of Section 4.2.4 confirm this interpretation. No single baseline reconstructs the Leiden communities. Therefore, the hacktivist Telegram ecosystem is organized along a co-affiliation structure that is strongly aligned with, but not reducible to, language, with topic and time providing secondary structure.

4.3 RQ2: Information Flow and Broker Detection

This section reports the results of the second analytical layer of the framework, which examines how content circulates between the communities identified in RQ1 and which entities mediate that circulation.

The structure of this section follows the methodology described in 3.5. Therefore, Section 4.3.1 reports the overall structure of the forwarding ecosystem, including intra-community concentration and the composition of the source classes feeding the labeled partition. Section 4.3.2 examines recurrent feeder sources outside the labeled universe and evaluates whether they support the presence of cross-community brokerage. Section 4.3.3 reports whether topic and forwarding events are correlated. Section 4.3.4 presents the content-hash propagation results. Finally, Section 4.3.5 summarizes these findings into the answer to RQ2.

It should be highlighted that, unless otherwise specified, all forwarding-based results operate on the forwarding universe F defined in Equation 3.17.

4.3.1 Forwarding Structure and Source Ecosystem

This section characterizes the forwarding universe used to analyze the information flow across the labeled Telegram ecosystem. The analysis considers the set F of forward events and examines according to the methodology described in Sections 3.5.1 and 3.5.2.

As shown in Figure 16, the label ecosystem is strongly dependent on sources outside the dataset. In fact, external sources account for approximately 64% of the observed volume. The second largest source is represented by the user with 24.04% of the total event, followed by labeled channels and orphan channels. This indicates that most of the content entering in the communities identified in RQ1 is imported from sources that are not part of the dataset.

However, if we restrict the analysis to only the sources contained in our corpus, we obtain that the information

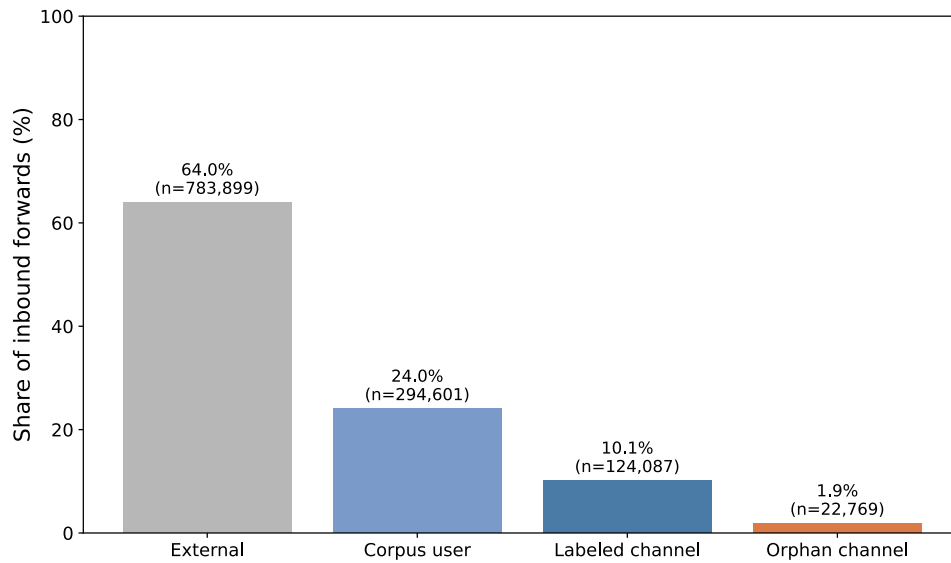


Figure 16: Aggregate and per-community inbound source composition

flow is highly concentrated within the community boundaries. As shown in Figure 17, among the channel-to-channel forwarding labels the 99.62% are intra-community while for the forwarding users the percentage is 97.3% (which decreases to 85% including also peripheral users). These results are important for the interpretation of RQ1, as the community partition was created from user co-affiliation and not from the flow of information. However, it reproduces the channel-level forwarding structure with 99.62% intra-community agreement. Therefore, forwardings provide an independent validation of the structural communities.

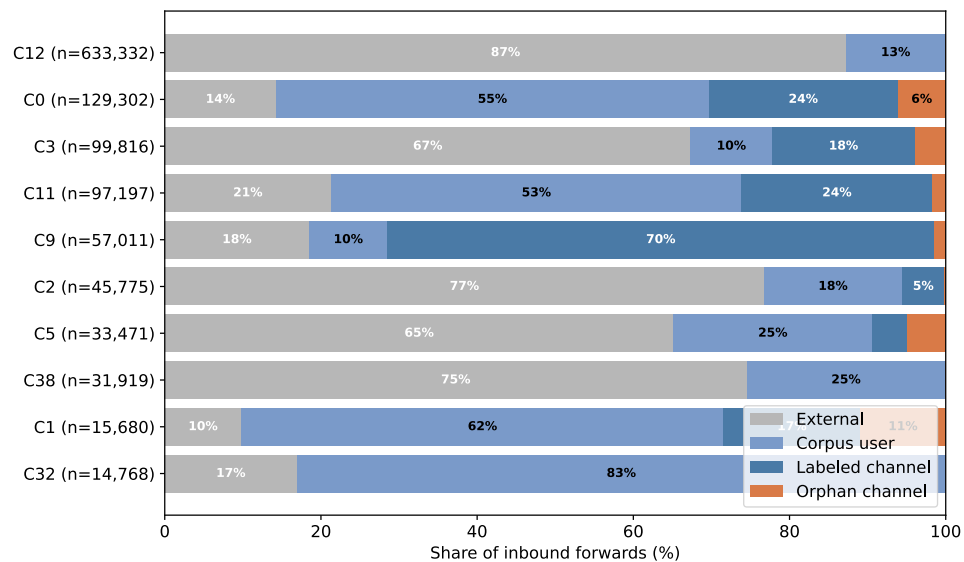


Figure 18: Community forwarding distribution

Figure 18 shows that this structure is not uniform across the ecosystem. Some communities behave primarily as broadcast aggregators, while others are more strongly driven by user-mediated or labeled-channel forwarding.

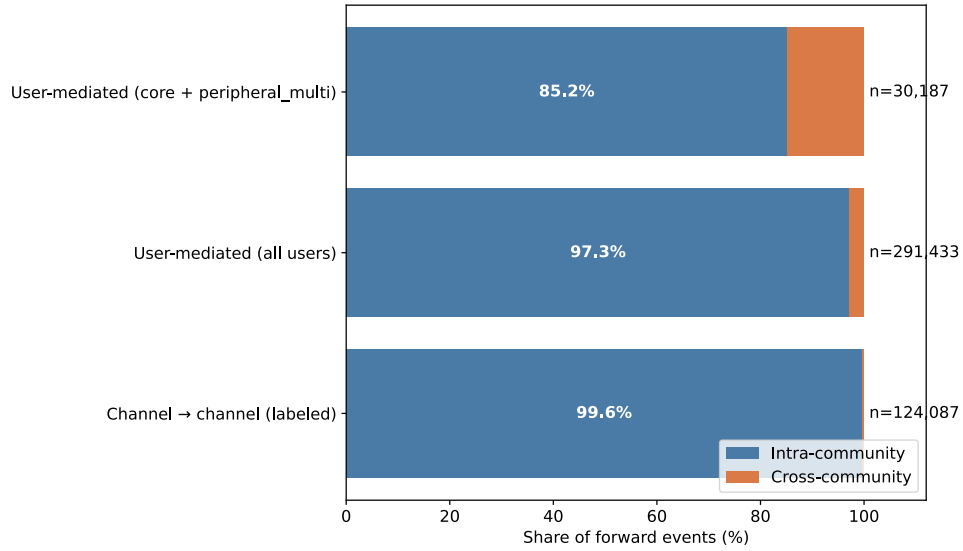


Figure 17: Intra-community versus cross-community forwarding share across source levels

Therefore, the composition of forwarding sources is itself a community-level property, not merely a global characteristic of the corpus.

4.3.2 Recurrent Feeders

This section focuses on *orphan channels* to assess whether they act as ecosystem feeders or whether they are associated with a specific target community. Orphan channels are defined in Section 3.4.1.

As Figure 19 shows, the forwarding behavior of *orphan channels* is highly concentrated. Among the retained recurrent feeders, the 75% send at least 80% of their forwarding to a single destination community. This is confirmed by both mean top share and HHI, which respectively are 88.8% and 0.842. These results indicate that the messages of the recurrent orphan feeders are usually forwarded to a single specific community, thus acting as an external supply channel attached to specific communities of the ecosystem.

Our analysis also provides evidence against the presence of broker entities in the observed forwarding universe. As Figure 17 illustrates, the results of the previous section have already confirmed that approximately 99% of the forwarding respect the community boundaries. Only 17 community exceed 20 forwards into two or more destination communities, and their activity is spread across multiple destinations rather than forming a cohesive brokerage layer. Therefore, the observable forward graph does not support the hypothesis that the flow of information is mediated by a small minority of broker entities.

Our conclusion is that the absence of a broker minority is analytically important. Diffusion studies often assume that circulation of information across communities is mediated by a small number of actors. In this dataset, the observable forward graph does not support that pattern. Our analysis better describes the flow of information as segmented and specific.

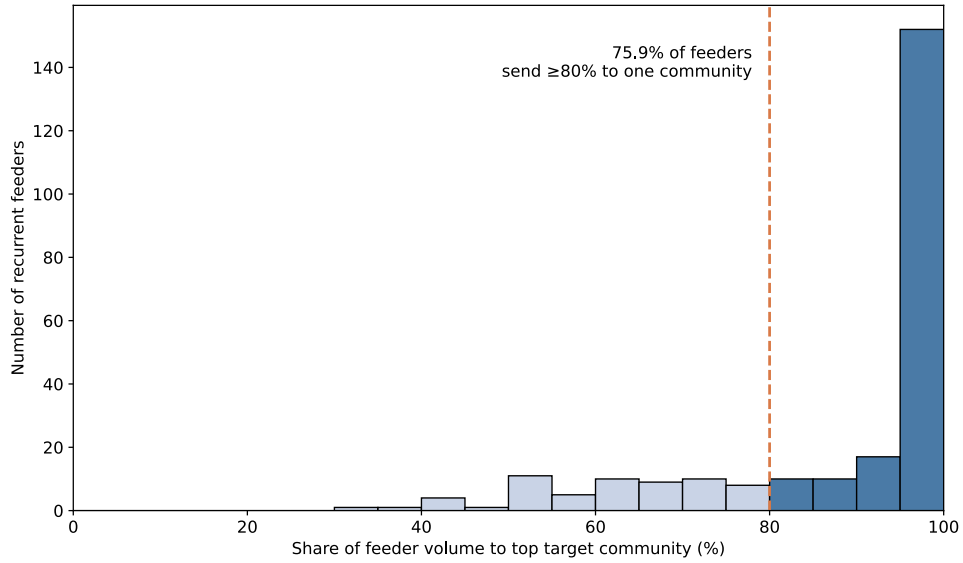


Figure 19: Distribution of recurrent feeders by the share of their forward volume sent to their single most-targeted community

As mentioned in the dataset limitation presented in Section 3.3, this conclusion refers to the observable forwarding universe, which may be limited due to the temporal metadata discrepancy.

4.3.3 Topic-Driven Sourcing Across Community

This section assesses whether forwarded content preserves topical coherence between origin and destination. The methodology of this section is described in Section 3.5.3. The analysis uses the G_1 macro-topic model described in Section 3.4.2. Therefore, the results should be interpreted as evidence of CTI-relevant forwarding rather than of the full forwarding universe. Only 100,446 forwards, corresponding to 5.6% of all directed forwards, carry a G_1 macro-topic on both endpoints.

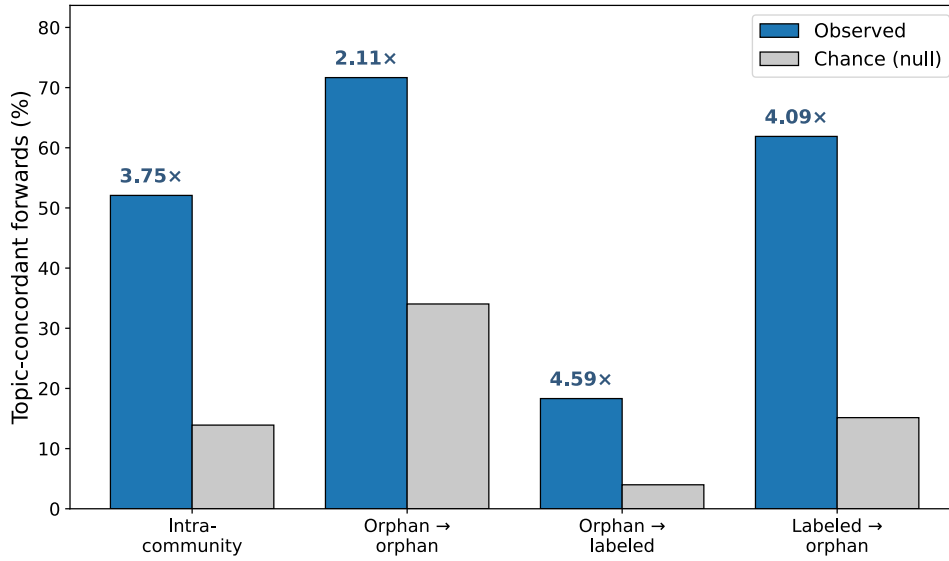


Figure 20: Observed topic-concordant forwarding compared with the permutation null by flow regime

As shown in Figure 20, all flow regimes exhibit topic concordance above chance. Intra-community forwarding matches the topic at 3.75 times the null rate. Although the highest match rate is observed among orphan-orphan submissions, the lift is 2.11 because orphan sources represent a smaller set of dominant topics and tend to match more often by chance. In contrast, orphan-labeled forwarding has the lowest observed match rate, but the highest lift, indicating that topic alignment is much stronger than expected once the marginal topic distribution is controlled. Labeled-orphan forwarding also exceeds chance.

Our analysis provides evidence that forwarding is not only structurally segmented but also topically selective. Content moves between origin and destination channel-days that share the same CTI macro-topic more often than expected from the topic distribution alone. This result also enriches the previous section outcome by showing that *orphan channels* are not merely external suppliers of volume, but when topic information is available, their forwardings align with the topic of the communities they feed.

4.3.4 Content-Hash Propagation

This section analyzes the propagation footprint of CTI-related content artifacts across the Telegram ecosystem, following the procedure described in Section 3.5.4.

Firstly, the set of 586,144 non-benign MD5 events extracted as IoCs generate 10,902 directed propagation edges where each edge represents a forwarding event carrying an MD5 hash from a source channel to a destination channel. Each edge is then classified according to the community status of its endpoints.

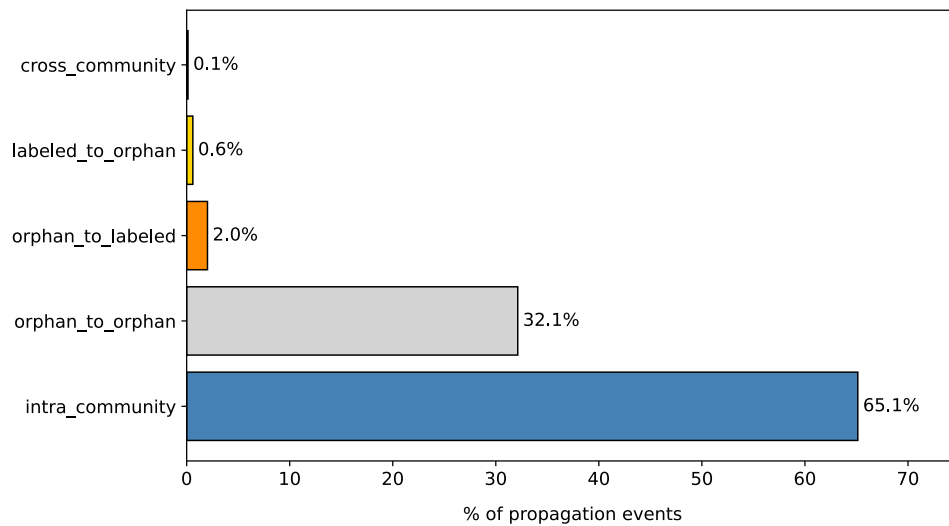


Figure 21: Flow-type composition of the 10,902 content-hash propagation edges

As shown in Figure 21, content-hash propagation strongly reflects the same pattern described in Section 4.3.1, where our results describe the internal forwarding ecosystem as highly concentrated within the community boundaries. This result is expected, since the content-hash graph is a subset of the forwarding graph. However, its value is more specific as it shows that the same community-bounded pattern also holds when the unit of analysis is a repeated technical artifact.

The footprint at the artifact-level is also limited. Most recurrent hashes reach only one destination channel, with a mean hash forwarding of 1.6. The main exception concerns a small number of orphan-origin artifacts that enter labeled communities. Figure 22 shows one representative example. The artifact originates from an orphan channel and appears to be forwarded four times over a span of 42.5 hours, reaching four different channels in two different communities.

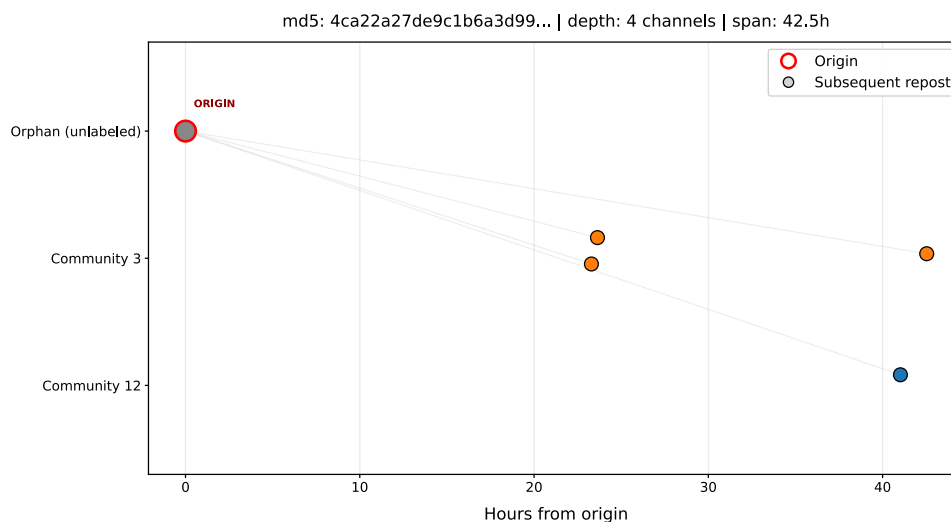


Figure 22: Content-hash propagation example

In conclusion, this analysis narrows the findings of the previous sections by showing that the community-bounded structure observed in the forwarding graph also holds when the unit of analysis is a repeated technical artifact. In addition, this section provides a conservative way to identify possible entry point of MD5 artifacts by combining forward metadata with first-observed timestamps and unique values.

4.3.5 RQ2 Answer

This section answers the second research question of this dissertation: *How does threat-relevant content spread across Telegram communities and which channels play key roles in that process?*

Our analysis reveals that the information flow in the observed Telegram ecosystem is strongly community-bounded. The forwarding graph presented in Section 4.3.1 shows that the most observable forwarding activity occurs within the same community rather than across community boundaries. In addition, these forwardings events are topically selective. Section 4.3.3 reveals a correlation between content and topics.

At the same time, the ecosystem is not completely closed. Sections 4.3.1 and 4.3.2 show that many labeled communities receive content from sources outside the labeled partition. These external and orphan channels act as content feeders of the ecosystem.

The content-hash analysis in Section 4.3.4 narrows this finding to the subset of forwarding events that carry recurrent MD5 artifacts. In addition, by combining forwarding metadata, first-observed timestamps, and unique MD5 values, this analysis provides a conservative way to identify candidate entry points of technical artifacts into the observed ecosystem.

Finally, the result of Section 4.3.2 does not support the presence of broker entities that mediate the diffusion between communities.

4.4 RQ3: IoC Patterns

This section reports the results of the third analytical layer of the framework, which examines how IoCs are distributed across the communities identified in RQ1.

The structure of this section follows the methodology described in 3.6. Section 4.4.1 summarizes the results of the extraction and verification of IoC. Section 4.4.2 reports the concentration of malicious indicators across structural communities. Section 4.4.3 reports the IoC distribution across the G_1 macro-topics. Section 4.4.4 reports the temporal behavior of indicators, including topic-IoC lead-lag relationships and the relative timing of indicator observations across the corpus and VirusTotal. Finally, Section 4.4.5 summarizes these findings into the answer to RQ3.

It should be highlighted that, unless otherwise specified, the community-level results operate on the labeled universe L with orphan channels excluded, while the topic-level results are restricted to the G_1 topic space defined

in Section 3.4.2.

4.4.1 IoC Extraction and Verification Summary

This section reports in detail the results of the extraction and verification methodology defined in Section 3.3.

The results of the extraction pipeline are presented in Table 4.8. The extraction identifies 6,509,685 raw indicator mentions in the corpus, which after deduplication became 2,163,113 unique indicators. The benign procedure filter 498,776 entries. Finally, 462,261 of the remaining 1,664,337 are queried against VirusTotal, corresponding to 27.77% of the extracted non-benign tier. As the table shows, among the resolved indicators only 12.17% have a VirusTotal report available. Finally, 8,992 indicators are classified as confirmed malicious under the thesis threshold of at least one VirusTotal engine, corresponding to 1.95% of the resolved set and 15.99% of the VT-present subset.

Table 4.8 IoC extraction and verification results over the whole corpus

Stage	Count	Share
Raw indicator mentions	6,509,685	—
Distinct indicators	2,163,113	33.2% of mentions
Extracted non-benign indicators	1,664,337	76.9% of distinct indicators
Resolved against VirusTotal	462,261	27.77% of extracted indicators
VT-present	56,242	12.17% of resolved indicators
Confirmed malicious (≥ 1 engines)	8,992	$\approx 1.95\%$ of resolved indicators

It should be mentioned that the verification process prioritizes IoC that appears in the labeled universe L . Therefore, the verification rate should not be interpreted as a uniform coverage of the full corpus. Of the total indicators queried against VirusTotal, approximately 71% belong to the lab-labeled universe L , while the remaining 29% belong to orphan channels. By restricting to the universe L , the confirmed-malicious set contains 3,672 distinct indicators. This subset forms the working set for the community-level analysis.

As shown in Table 4.9, the resolved sample is dominated by MD5 hashes, which account for 288,198 of 462,261 resolved indicators. However, this high verification volume does not translate into a high malicious rate. By contrast, IPv4 addresses represent a smaller queried set, but have the highest within-type malicious rate, with 16.25% of confirmed malicious indicators. The number of detecting engines provides an additional perspective for this result. For IPv4 addresses and URLs, the median number of engines is one. This means that most of the malicious network indicators are not supported by broad multi-engine agreement. By contrast, hash-based indicators present a stronger engine consensus once they are flagged as malicious, with a median number of detecting engines of 21 for MD5, 29 for SHA-256, and 35.5 for SHA-1.

Table 4.9 Resolved and confirmed-malicious indicators by type

Type	Resolved	Malicious	Malicious / resolved	Share of malicious	Median engines
URL	140,416	4,452	3.17%	49.51%	1.0
IPv4	16,895	2,746	16.25%	30.54%	1.0
MD5	288,198	1,710	0.59%	19.02%	21.0
SHA-256	13,728	67	0.49%	0.75%	29.0
SHA-1	2,730	16	0.59%	0.18%	35.5
IPv6	294	1	0.34%	0.01%	1.0
Total	462,261	8,992	1.95%	100.00%	2.0

Table 4.10 further describes the origin of the malicious MD5 indicator. Of the 1,710 confirmed-malicious MD5 hashes, 1,698 originate from file attachments and only 12 from message text. Within the attachment-derived subset, the malicious signal is concentrated in archive files and octet-stream payloads, while text-file attachments contribute a large volume of resolved hashes, but almost no confirmed-malicious indicators. This pattern is consistent with reported malware-distribution practices on Telegram, where actors attach compressed archives (RAR or ZIP) to channel posts [3].

4.4.2 Distribution Across Communities

This section assesses whether malicious indicators are distributed uniformly across communities or concentrated in a small subset of the ecosystem. The methodology of this section is described in Section 3.6.1.

Our analysis reveals that the confirmed malicious indicators are highly concentrated in a small structural minority of communities. Of the 82 communities in L , 80 contain at least one extracted indicator, but only 55 contain at least one confirmed malicious indicator. The three largest communities for the number of indicators account for 50% of the total, while the top five for 72%, and the top ten account for 92.3%. The Gini coefficient of 0.893 further supports this concentration.

We verify that this concentration is not an artifact of the partial verification. Therefore, we compute the same analysis in the broader *VT-present* set. The results remain almost identical with a Gini coefficient of 0.886, and the top three, five, and ten communities with almost the same percentage of coverage. This similarity between the *VT-present* and *confirmed-malicious* tiers indicates that concentration reflects a broader structural distribution of threat-relevant indicators.

However, a community that posts more indicators has more opportunities to post malicious ones. The results of the chi-square goodness-of-fit test of $\chi^2 = 9,639.4$ reject this extracted-volume null model, confirming that malicious indicators are not distributed proportionally to the overall activity of the extracted-indicator. In addition,

Table 4.10 Origin and MIME-family composition of resolved and malicious MD5 indicators

Origin / MIME family	Resolved	Malicious
Attachment	277,895	1,698
text file	208,584	26
archive	51,737	1,215
document	9,867	28
octet-stream	5,695	404
other	1,470	15
text only	479	10
media	50	0
executable/script	13	0
Message text	10,301	12
Both	2	0
Total	288,198	1,710

the lift comparison further shows that several leading communities produce substantially more malicious indicators than expected from their extracted-indicator volume.

In our analysis, we also examine whether distinct communities share the same malicious indicators or whether each draws on a separate set. Our results indicate that communities are highly independent in terms of their threat indicators with only 6% of the total confirmed malicious indicators appearing in more than one community, while 94.0% are community-exclusive. This result is consistent with the forwarding structure described in Section 4.3.1.

In conclusion, this analysis indicates that malicious IoC production is not evenly distributed across communities, but rather is concentrated in a small minority. Furthermore, this concentration is stable across different tiers and is independent of the activity volume. Finally, our analysis also indicates that the top producers also differ in type, which means that each community has a specific type IoC behavior.

4.4.3 Distribution Across Topics

This section assesses whether malicious indicators are distributed uniformly across thematic areas or concentrated in a small subset of the topics. The methodology of this section is described in Section 3.6.1. The analysis uses the G_1 macro-topic model described in Section 3.4.2. Therefore, the results should be interpreted as evidence of CTI-relevant IoC production.

As Table 4.11 shows, of the total confirmed malicious indicators, only 37.3% appear on a channel-day carrying a G_1 topic. The remaining 62.7% fall outside the CTI-focused topic space. Malicious indicators are not predominantly

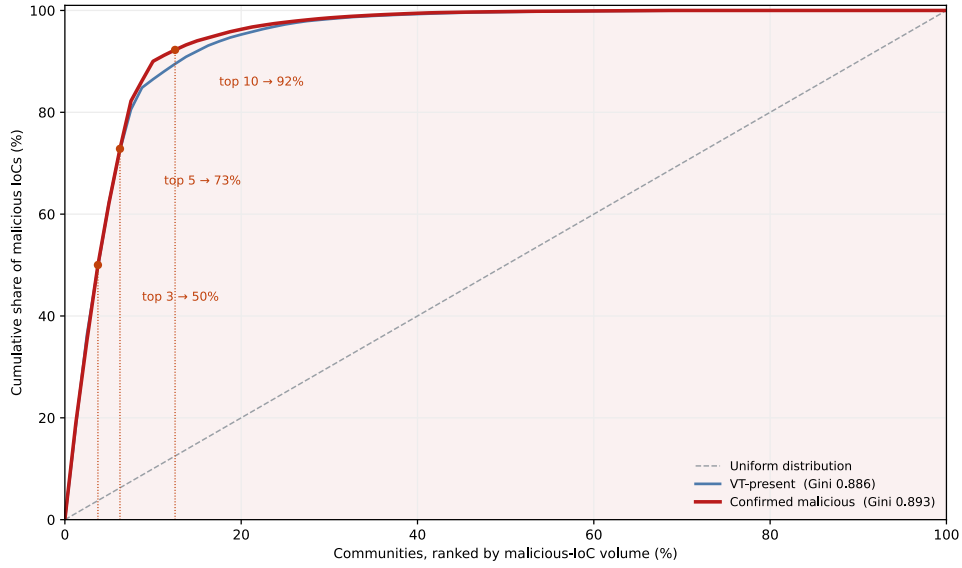


Figure 23: Concentration of malicious loCs across communities

posted within thematically CTI-focused discussions, but a substantial majority circulate in noisier or off-domain contexts. Our analysis describes the CTI-relevant fraction of the malicious set rather than its entirety.

Table 4.11 Topic-space coverage of distinct confirmed-malicious indicators

Context	Indicators	Share
G_1 CTI topic	3,355	37.3%
G_0 off-domain	1,062	11.8%
G_0 outlier	4,204	46.8%
Not modelled	371	4.1%
Total	8,992	100.0%

Our analysis indicates that malicious indicators are concentrated in a small set of macro-topics. The top three macro-topics account for 63.9% of topic-bearing malicious observations, the top five for 77.7%, and the top ten for 93.6% with a Gini coefficient of 0.776. Therefore, malicious-indicator production is clearly thematically concentrated, although the concentration is less extreme than the community-level pattern reported in Section 4.4.2. This difference is expected because the G_1 macro-topic space has less categories than the community space.

We verify that this concentration is not an artifact of the posting activity using the chi-square goodness-of-fit. The result ($\chi^2 = 4,772.7$, $p < 0.001$) indicates that the malicious indicators are not distributed proportionally to the overall activity of the indicator within G_1 . Additionally, Figure 24 compares the observed share of malicious indicators with the share expected under the extracted-volume baseline for the selected high-volume macro-topics.

The two highest-volume themes, *Cybercrime Services and Illicit Trade* (M6) and *Cybersecurity Vulnerabilities and Threats* (M13), account for the largest shares, but both have moderate observed-to-expected ratios of 2.18. By contrast, narrower themes show stronger over-representation. *Iranian Proxy Services and Cyber Donations* (M2) reach a boost of 6.94, while *Telegram Channel Growth and Services* (M4) and *Digital Scams and Cryptocurrency Exploits* (M7) reach respectively 3.92 and 3.10. This shows that the macro-topics with the highest number of malicious loC are not necessarily those that are most over-represented relative to their extracted-indicator baseline.

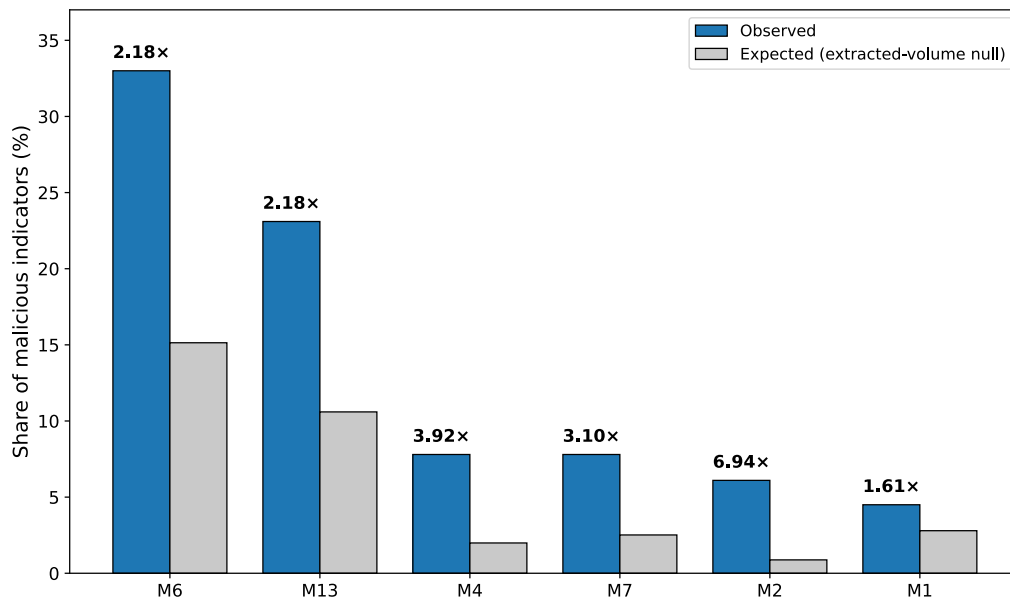


Figure 24: Observed share of malicious indicators per macro-topic against the share expected under the extracted-volume null

Our analysis also indicates a weak association between topic and loCtype, where market themes such as *Cybercrime Services* (M6) and *Digital Scams* (M7) lean towards URLs, instead proxy and infrastructure-related themes carry a larger share of IP addresses. However, this association is weaker than that observed in Section 4.4.2).

In conclusion, our analysis indicates that malicious-indicators posted within the G_1 subset are strongly concentrated in a small set of macro-topics and are independent of the posting loC activity volume. At the same time, the coverage result shows that most confirmed-malicious indicators appear outside CTI-focused topics.

4.4.4 Temporal Patterns

This section examines the temporal behavior of malicious. First, it asks whether bursts of thematic activity and indicator production are aligned in time. Second, it compares the first appearance of malicious loC in the corpus with its first submission to VirusTotal. The methodology of this section is described in Section 3.6.2.

a) Topic–IoC lead–lag Results

Our analysis of the cross-correlation between the two daily time series indicates synchrony as the dominant temporal pattern. This means that indicators of malicious activity tend to appear simultaneously with the occurrence of the corresponding topic.

Figure 25 presents the strongest association observed in our analysis. The *Cybersecurity Vulnerabilities and Threats* (M13) topic peaks at lag zero with $r = 0.73$. The curve shows a broad correlation band centered on $\delta = 0$, indicating that thematic activity and malicious IoC production rise and fall together. Only two macro-topics show a significant positive lag, M4 (4 days) and M14 (11 days), and in both the correlation at lag zero remains substantial, so the lead is short and partial rather than a clear precursor relationship. No macro topics show significant negative lag. The complete association report is presented in the Appendix D.1

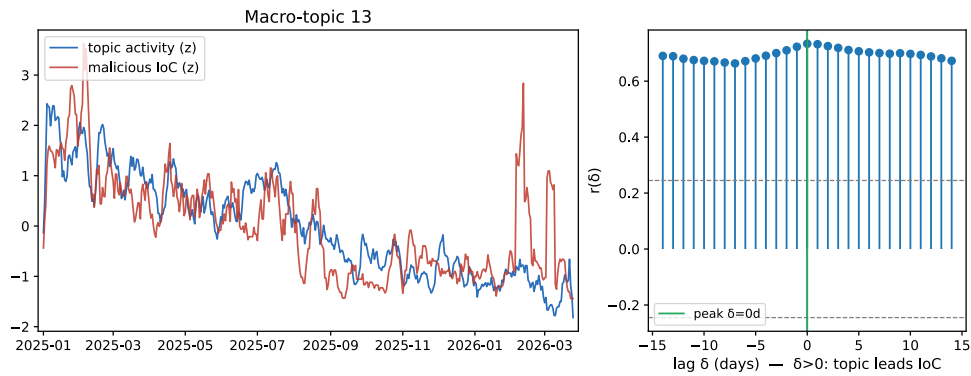


Figure 25: Lead–lag relationship for *Cybersecurity Vulnerabilities and Threats* (M13)

In conclusion, our analysis shows that the relationship between topic activity and malicious-indicator is dominated by synchrony. This result becomes more meaningful when read together with the first-seen analysis.

b) First-Seen Results

This analysis compares the first appearance of malicious IoC in the corpus with its first submission to VirusTotal. Of the confirmed-malicious set, 6,245 indicators (69.5%) carry a VirusTotal first-submission timestamp. After excluding indicators first submitted to VirusTotal before the corpus observation window opens, 3,316 retain both dates and enter the comparison. Among these indicators, only 1,308 appear in the corpus before VirusTotal, as reported in Table 4.12.

As shown in the table, the distribution is typically short with a median lead of two days. However, the distribution presents a small number of cases that extend to several months. For this reason, the median is more representative than the mean. Furthermore, the lead highly differs by indicator type. URLs account for the largest share of early-observed indicators and have a median lead of 14 days. By contrast, MD5 hashes are close to their VirusTotal submission date, with a median lead of one day. This suggests that malicious URLs often circulate in the monitored

Telegram ecosystem before being submitted to a reputation service, whereas file hashes tend to appear in the corpus and VirusTotal almost concurrently.

Table 4.12 Corpus-first malicious indicators by type, with MD5 hashes further decomposed by attachment MIME family

Indicator type	Subgroup	$n (\Delta \leq 0)$	$n (\Delta < 0)$	Median lead (d)	Max lead (d)
URL	–	991	846	14.0	450
MD5	all	852	460	1.0	368
	archive	649	351	1.0	368
	octet-stream	188	101	1.0	298
	document	3	2	7.0	175
	text file	6	2	0.0	49
	text only	6	4	1.5	4
SHA-256	–	17	2	0.0	2
All	–	1,860	1,308	2.0	450

Within the MD5 subset, early-observed hashes are concentrated in archives and octet-stream payloads. Together, they account for most corpus-first MD5 hashes. This aligns with the results of Section 4.4.1 and Roy et al. [3], where the archives were identified as the dominant malicious-attachment family.

By enriching the early observed IoCs we notice that the communities surfacing indicators early are therefore the same high-volume producers identified in Section 4.4.2, rather than a distinct group of early-warning channels.

In conclusion, this analysis indicates that the monitored Telegram ecosystem surfaces a substantial minority of malicious indicators before they reach an external reputation service. Furthermore, reading this result together with the lead-lag analysis gives a new perspective. Indicators are produced in sync with the topical activity surrounding them, and a share of them appear in the corpus ahead of their first external submission. This indicates that the ecosystem can expose threat artifacts while they are thematically active, before they are represented in external reputation data such as VirusTotal.

4.4.5 RQ3 Answer

This section answers the third research question of this dissertation: *Do certain communities consistently produce more cyber threat indicators, and can their structure or topic focus help predict that?*

Our analysis reveals that malicious-indicator production is highly uneven across the Telegram ecosystem. Section 4.4.2 shows that a small minority of communities accounts for the large majority of confirmed-malicious indicators. This concentration remains visible across verification tiers and cannot be explained only by of the

overall volume of extracted indicators. The same section also shows that the sharing of indicators across community boundaries is limited, which is consistent with the information flow results described in 4.3.

At the same time, Section 4.4.3 shows that malicious indicators are concentrated in a small set of macro-topics, indicating that this structural concentration is mirrored by the thematic layer. However, the topics with the largest absolute number of malicious indicators are not necessarily those most over-represented in comparison of their extracted-indicator baseline.

Finally, the temporal analysis in Section 4.4.4 indicates that the IoC production moves mainly in synchrony. Furthermore, a substantial minority of indicators appear in the corpus before their first external submission. This early observation is carried by the same high-volume communities identified in Section 4.4.2, rather than by a distinct group of early-warning channels. Moreover, this means that indicators are produced in synchrony with the topical activity surrounding them and a substantial minority appear in the corpus before their first external submission. This indicates that the monitored ecosystem can expose part of the IoC space while it is thematically active, before it is represented in external reputation data such as VirusTotal. However, this is better understood as early observation rather than reliable prediction.

5 Activity Plan

ID	Activity	Begin	End
1	Dissertation/Thesis	09/02/2026	20/06/2026
1.1	Research Foundation & State of the Art	09/02/2026	23/02/2026
1.1.1	Problem definition & RQ alignment	09/02/2026	13/02/2026
1.1.2	Telegram/CTI, community detection, topic modeling, IoC	09/02/2026	23/02/2026
1.1.3	Technology stack evaluation	16/02/2026	23/02/2026
1.2	Dataset & Preprocessing Pipeline	23/02/2026	25/03/2026
1.2.1	Infrastructure setup & data ingestion	23/02/2026	28/02/2026
1.2.2	Branch-specific cleaning, deduplication & language detection	25/02/2026	09/03/2026
1.2.3	Parquet store loading & validation	09/03/2026	25/03/2026
1.3	Methodology Consolidation & RQ Reformulation	18/03/2026	15/05/2026
1.3.1	Scope confirmation with supervisors	18/03/2026	25/03/2026
1.3.2	RQ2 reformulation & scope expansion	01/05/2026	15/05/2026
1.3.3	Methodology chapter revision	25/03/2026	15/05/2026
1.4	Structural Layer (RQ1)	25/03/2026	20/04/2026
1.4.1	User co-affiliation graph construction	25/03/2026	01/04/2026
1.4.2	Leiden community detection & resolution (γ) tuning	01/04/2026	08/04/2026
1.4.3	Community profiling & alternative-partition comparison	08/04/2026	20/04/2026
1.5	Thematic Layer (RQ1)	20/04/2026	12/05/2026
1.5.1	BERTopic dual-model (G0/G1) run & parameter tuning	20/04/2026	03/05/2026
1.5.2	Macro-topic mapping & community topical profiling	03/05/2026	09/05/2026
1.5.3	Lift / JSD analysis & topic-spike validity check	09/05/2026	12/05/2026
1.6	Information-Flow Layer (RQ2)	12/05/2026	28/05/2026
1.6.1	Forwarding graph & source attribution	12/05/2026	18/05/2026
1.6.2	Forwarding-ecosystem & orphan-feeder typology	18/05/2026	23/05/2026
1.6.3	Broker-absence, topic-driven sourcing & content-hash propagation	23/05/2026	28/05/2026
1.7	Indicator Layer (RQ3)	28/05/2026	10/06/2026
1.7.1	Regex-based IoC extraction	28/05/2026	01/06/2026
1.7.2	IoC verification (VirusTotal)	28/05/2026	05/06/2026
1.7.3	IoC distribution & community-exclusivity	05/06/2026	08/06/2026
1.7.4	Temporal redistribution analysis	08/06/2026	10/06/2026
1.8	Thesis Writing	09/02/2026	24/06/2026
1.8.1	Literature Review (Ch. 2)	09/02/2026	10/04/2026
1.8.2	Methodology (Ch. 3)	01/03/2026	15/05/2026
1.8.3	Results & Discussion (Ch. 4)	01/05/2026	15/06/2026
1.8.4	Introduction & Conclusion (Ch. 1, 6)	10/06/2026	22/06/2026
1.8.5	Consistency checks, figure regeneration & proofreading	15/06/2026	24/06/2026
1.9	Advisor Meetings (bi-weekly)	09/02/2026	24/06/2026

Table 5.13 Work Plan

6 Conclusion

With this study, we present TGIF, a multi-layer analytical framework for Telegram CTI that analyzes the Telegram threat ecosystem at the community level by combining four layers. Together, these layers allow for the study of the organization of threat communities, the flow of content within and across these communities, and the generation and propagation of malicious indicators. Applied to the corpus of 45.7 million Telegram messages provided by Constella Intelligence, the framework shows that the analyzed hacktivist ecosystem is neither a set of isolated channels neither a fully interconnected network. Instead, it appears as a segmented and community-bounded space. The structural layer identifies robust user communities that are not reducible to language, topic, or activity timing. The flow layer shows that forwarding and content-hash propagation almost do not cross community boundaries. The indicator layer further shows that malicious indicators are concentrated in a subset of communities and topics and the temporal analysis of IoC shows that part of the malicious-indicator space appears in the corpus before it is represented in external reputation data. Therefore, the main contribution of this dissertation is a multi-layer analytical pipeline that connects evidence from the structural, semantic, flow-based, and technical-indicator perspectives within the same corpus. Thus, this study demonstrates that community based perspective provides cross-layer findings that cannot be achieved by single-channel or single-method approaches.

6.1 Limitations

Beyond the dataset constraints already discussed in Section 3.3, the analysis carries several limitations relevant to the interpretation of the findings. All the results obtained are properties of the analyzed corpus. Regarding the structural communities, no ground truth community labels exist, and propagation is used to extent the partition to users and channels outside the core co-affiliation graph. The use of a non-overlapping algorithm forces channels and users that bridge several communities to single label. The forwarding universe analyzed in Section 4.3.1 is significantly influenced by the metadata limitation described in Section 3.3, which restricts the amount of data available. On the indicator side, feasibility constraints limit the verification to only a subset of the extracted IoC. We focus on the verification of the labeled universe and therefore the rate of confirmed-malicious cannot be read as uniform coverage over the corpus. Furthermore, a single engine is used as source of confirmation.

6.2 Future Work

This work suggests three directions for future research. The first is to complement the non overlapping Leiden partition with an overlapping community-detection method and examine whether the channels and users that bridge several communities form a distinct structural layer. The second is to expand this analysis to external platforms in order to trace content and threat actors beyond the Telegram boundary. The third concern how the ecosystem evolves over time. Threat-related Telegram communities are dynamic since channels can be removed or migrated to

successor. Reconstructing the community structure in temporal snapshots would then show whether communities persist or reorganize once a channel is removed.

7 Bibliography

- [1] Joel Francis. Mapping Threat Actor Abuse of Telegram: From C2 to Hacktivism, November 2025. URL <https://www.silobreaker.com/blog/cyber-threats/mapping-threat-actor-abuse-of-telegram-from-c2-to-hacktivism/>. Section: Cyber threats.
- [2] Roy Ricaldi, Tina Marjanov, Luca Allodi, and Alice Hutchings. Uncovering the Trust Signals Supporting Telegram's Cybercrime Economy. In *2025 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–17, November 2025. doi: 10.1109/eCrime66972.2025.11327769. URL <https://ieeexplore.ieee.org/document/11327769>. ISSN: 2159-1245.
- [3] Sayak Saha Roy, Elham Pourabbas Vafa, Kobra Khanmohammadi, and Shirin Nilizadeh. DarkGram: A Large-Scale Analysis of Cybercriminal Activity Channels on Telegram, October 2025. URL <http://arxiv.org/abs/2409.14596>. arXiv:2409.14596 [cs].
- [4] Dincy R. Arikkat, Sneha B. T, Serena Nicolazzo, Antonino Nocera, Vinod P, Rafidha Rehimani K. A, and Karthika R. CTI Dataset Construction from Telegram, September 2025. URL <http://arxiv.org/abs/2509.20943>. arXiv:2509.20943 [cs].
- [5] Vincenzo Imperati, Massimo La Morgia, Alessandro Mei, Alberto Maria Mongardini, and Francesco Sassi. The Conspiracy Money Machine: Uncovering Telegram's Conspiracy Channels and their Profit Model, September 2025. URL <http://arxiv.org/abs/2310.15977>. arXiv:2310.15977 [cs.CY].
- [6] Devang Shah, Hriday Ranka, Lynnette Hui Xian NG, and Swapneel Mehta. Bridging Nodes and Narrative Flows: Identifying Intervention Targets for Disinformation on Telegram, November 2024. URL <http://arxiv.org/abs/2411.05922>. arXiv:2411.05922 [cs].
- [7] Rohit Shewale. Telegram Users by Country 2026 (Global Data) | Resourcera, May 2025. URL <https://resourcera.com/data/social/telegram-users/>. Section: Social.
- [8] Massimo La Morgia, Alessandro Mei, and Alberto Maria Mongardini. TGDataset: Collecting and Exploring the Largest Telegram Channels Dataset. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, pages 2325–2334, New York, NY, USA, July 2025. Association for Computing Machinery. ISBN 979-8-4007-1245-6. doi: 10.1145/3690624.3709397. URL <https://dl.acm.org/doi/10.1145/3690624.3709397>.
- [9] Anh V. Vu, Daniel R. Thomas, Ben Collier, Alice Hutchings, Richard Clayton, and Ross Anderson. Getting Bored of Cyberwar: Exploring the Role of Low-level Cybercrime Actors in the Russia-Ukraine Conflict, April 2024. URL <http://arxiv.org/abs/2208.10629>. arXiv:2208.10629 [cs.CR].

- [10] Pham Van Toi, Nguyen Trung Dung, Hoang Linh Ph  ng, and Nguyen Huu Long. A Scalable Telegram-Based Botnet Framework for Stealthy Remote Command and Control. *Journal of Science and Technology on Information security*, pages 43–51, September 2025. ISSN 2615-9570. doi: 10.54654/isj.v2i25.1102. URL https://isj.vn/index.php/journal_STIS/article/view/1102.
- [11] Mark Newman. Introduction. In *Networks*, pages 1–12. Oxford University PressOxford, 2 edition, July 2018. ISBN 978-0-19-880509-0 978-0-19-184323-5. doi: 10.1093/oso/9780198805090.003.0001. URL <https://academic.oup.com/book/27884/chapter/203813506>.
- [12] Ronald L. Breiger. The Duality of Persons and Groups. *Social Forces*, 53(2):181–190, 1974. ISSN 0037-7732. doi: 10.2307/2576011. URL <https://www.jstor.org/stable/2576011>.
- [13] Matthieu Latapy, Cl  mence Magnien, and Nathalie Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1):31–48, January 2008. ISSN 0378-8733. doi: 10.1016/j.socnet.2007.04.006. URL <https://www.sciencedirect.com/science/article/pii/S0378873307000494>.
- [14] Natalie R. Smith, Paul N. Zivich, Leah M. Frerichs, James Moody, and Allison E. Aiello. A Guide for Choosing Community Detection Algorithms in Social Network Studies: The Question Alignment Approach. *American Journal of Preventive Medicine*, 59(4):597–605, October 2020. ISSN 07493797. doi: 10.1016/j.amepre.2020.04.015. URL <https://linkinghub.elsevier.com/retrieve/pii/S0749379720302166>.
- [15] M. E. J. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1):016132, June 2001. ISSN 1063-651X, 1095-3787. doi: 10.1103/PhysRevE.64.016132. URL <https://link.aps.org/doi/10.1103/PhysRevE.64.016132>.
- [16] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002. doi: 10.1073/pnas.122653799. URL <https://www.pnas.org/doi/abs/10.1073/pnas.122653799>.
- [17] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659: 1–44, November 2016. ISSN 03701573. doi: 10.1016/j.physrep.2016.09.002. URL <http://arxiv.org/abs/1608.00163>. arXiv:1608.00163 [physics].
- [18] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks (long version), December 2005. URL <http://arxiv.org/abs/physics/0512106>. arXiv:physics/0512106.
- [19] Usha Nandini Raghavan, Reka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, September 2007. ISSN 1539-

- 3755, 1550-2376. doi: 10.1103/PhysRevE.76.036106. URL <http://arxiv.org/abs/0709.2938>. arXiv:0709.2938 [physics.soc-ph].
- [20] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0601602103. URL <http://arxiv.org/abs/physics/0602124>. arXiv:physics/0602124.
- [21] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, January 2007. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0605965104. URL <http://arxiv.org/abs/physics/0607100>. arXiv:physics/0607100.
- [22] V. A. Traag, P. Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, July 2011. ISSN 1539-3755, 1550-2376. doi: 10.1103/PhysRevE.84.016114. URL <http://arxiv.org/abs/1104.3083>. arXiv:1104.3083 [physics.soc-ph].
- [23] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, October 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008. URL <http://arxiv.org/abs/0803.0476>. arXiv:0803.0476 [physics.soc-ph].
- [24] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, March 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-41695-z. URL <https://www.nature.com/articles/s41598-019-41695-z>.
- [25] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, January 2008. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0706851105. URL <http://arxiv.org/abs/0707.0609>. arXiv:0707.0609 [physics.soc-ph].
- [26] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature03607. URL <http://arxiv.org/abs/physics/0506133>. arXiv:physics/0506133.
- [27] Andrea Lancichinetti, Filippo Radicchi, Jose' Javier Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, April 2011. ISSN 1932-6203. doi: 10.1371/journal.pone.0018961. URL <http://arxiv.org/abs/1012.2363>. arXiv:1012.2363 [physics.soc-ph].
- [28] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining, WSDM*

- '13, pages 587–596, New York, NY, USA, February 2013. Association for Computing Machinery. ISBN 978-1-4503-1869-3. doi: 10.1145/2433396.2433471. URL <https://dl.acm.org/doi/10.1145/2433396.2433471>.
- [29] Md Imran Hossen, Ashraful Islam, Farzana Anowar, Eshtiaq Ahmed, Mohammad Masudur Rahman, Xiali, and Hei. Generating Cyber Threat Intelligence to Discover Potential Security Threats Using Classification and Topic Modeling, November 2022. URL <http://arxiv.org/abs/2108.06862>. arXiv:2108.06862 [cs.LG].
- [30] Felipe Moreno-Vera. Inferring Discussion Topics about Exploitation of Vulnerabilities from Underground Hacking Forums. In *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 816–821, October 2023. doi: 10.1109/ICTC58733.2023.10393244. URL <http://arxiv.org/abs/2405.04561>. arXiv:2405.04561 [cs.CR].
- [31] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003. ISSN ISSN 1533-7928. URL <https://jmlr.csail.mit.edu/papers/v3/blei03a.html>.
- [32] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999. ISSN 1476-4687. doi: 10.1038/44565. URL <https://www.nature.com/articles/44565>.
- [33] Qiang Jipeng, Qian Zhenyu, Li Yun, Yuan Yunhao, and Wu Xindong. Short Text Topic Modeling Techniques, Applications, and Performance: A Survey, April 2019. URL <http://arxiv.org/abs/1904.07695>. arXiv:1904.07695 [cs.IR].
- [34] Dima Angelov. Top2Vec: Distributed Representations of Topics, August 2020. URL <http://arxiv.org/abs/2008.09470>. arXiv:2008.09470 [cs.CL].
- [35] Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure, March 2022. URL <http://arxiv.org/abs/2203.05794>. arXiv:2203.05794 [cs].
- [36] Hatma Suryotrisongko, Hari Ginardi, Henning Titi Ciptaningtyas, Saeed Dehqan, and Yasuo Musashi. Topic Modeling for Cyber Threat Intelligence (CTI). In *2022 Seventh International Conference on Informatics and Computing (ICIC)*, pages 1–7, December 2022. doi: 10.1109/ICIC56845.2022.10006988. URL <https://ieeexplore.ieee.org/document/10006988>.
- [37] Jaehwan Baek, Jeonghoon O, Seungwoo Jeong, and Wooju Kim. A Study on Improving the Automatic Classification Performance of Cybersecurity MITRE ATT&CK Tactics Using NLP-Based ModernBERT and BERTopic Models. *Electronics*, 14(22):4434, January 2025. ISSN 2079-9292. doi: 10.3390/electronics14224434. URL <https://www.mdpi.com/2079-9292/14/22/4434>.

- [38] Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyer. TopicGPT: A Prompt-based Topic Modeling Framework, April 2024. URL <http://arxiv.org/abs/2311.01449>. arXiv:2311.01449 [cs.CL].
- [39] Sharad Goel, Ashton Anderson, Jake Hofman, and Duncan J. Watts. The Structural Virality of Online Diffusion. *Management Science*, 62(1):180–196, 2016. URL <https://ideas.repec.org//a/inm/ormnsc/v62y2016i1p180-196.html>.
- [40] Roger V. Gould and Roberto M. Fernandez. Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks. *Sociological Methodology*, 19:89–126, 1989. ISSN 0081-1750. doi: 10.2307/270949. URL <https://www.jstor.org/stable/270949>.
- [41] Savvas Zannettou, Tristan Caulfield, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini, and Guillermo Suarez-Tangil. On the Origins of Memes by Means of Fringe Web Communities, September 2018. URL <http://arxiv.org/abs/1805.12512>. arXiv:1805.12512 [cs.SI].
- [42] Roy Ricaldi, Victor Asanache, and Luca Allodi. TeleHunt: A Framework and Tool for Efficient Cybercriminal Community Discovery on Telegram, June 2026. URL <http://arxiv.org/abs/2606.04657>. arXiv:2606.04657 [cs.CR].
- [43] Giordano Paoletti, Carlos H. G. Ferreira, Luca Vassio, Leonardo Rocha, and Jussara M. Almeida. Tracing the 2024 U.S. election debate on Telegram with LLMs and graph analysis. *Social Network Analysis and Mining*, 15(1):91, August 2025. ISSN 1869-5469. doi: 10.1007/s13278-025-01504-0. URL <https://doi.org/10.1007/s13278-025-01504-0>.
- [44] Robert Dorfman. A Formula for the Gini Coefficient. *The Review of Economics and Statistics*, 61(1):146–149, 1979. ISSN 0034-6535. doi: 10.2307/1924845. URL <https://www.jstor.org/stable/1924845>.
- [45] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, July 1948. ISSN 00058580. doi: 10.1002/j.1538-7305.1948.tb01338.x. URL <https://ieeexplore.ieee.org/document/6773024>.
- [46] Karl Pearson. On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to Have Arisen from Random Sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, July 1900. ISSN 1941-5982, 1941-5990. doi: 10.1080/14786440009463897. URL <https://www.tandfonline.com/doi/full/10.1080/14786440009463897>.
- [47] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985. ISSN 1432-1343. doi: 10.1007/BF01908075. URL <https://doi.org/10.1007/BF01908075>.

- [48] Andrew Rosenberg and Julia Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In Jason Eisner, editor, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL <https://aclanthology.org/D07-1043/>.
- [49] Harald Cramer. *Mathematical Methods of Statistics*, volume 9 of *Princeton Mathematical Series*. Princeton University Press, 1946. ISBN 978-0-691-08004-8.
- [50] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, January 1991. ISSN 1557-9654. doi: 10.1109/18.61115. URL <https://ieeexplore.ieee.org/document/61115>.
- [51] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, H erve J egou, and Tomas Mikolov. FastText.zip: Compressing text classification models, December 2016. URL <http://arxiv.org/abs/1612.03651>. arXiv:1612.03651 [cs].
- [52] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification, August 2016. URL <http://arxiv.org/abs/1607.01759>. arXiv:1607.01759 [cs].
- [53] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzm  n, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL <https://www.aclweb.org/anthology/2020.acl-main.747>.
- [54] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczy nski, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings 2019 Network and Distributed System Security Symposium*, 2019. doi: 10.14722/ndss.2019.23386. URL <http://arxiv.org/abs/1806.01156>. arXiv:1806.01156 [cs.CR].
- [55] Juan Caballero, Gibran Gomez, Srdjan Matic, Gustavo S  nchez, Silvia Sebasti  n, and Arturo Villaca  as. The Rise of GoodFATR: A Novel Accuracy Comparison Methodology for Indicator Extraction Tools. *Future Generation Computer Systems*, 144:74–89, July 2023. ISSN 0167739X. doi: 10.1016/j.future.2023.02.012. URL <http://arxiv.org/abs/2208.00042>. arXiv:2208.00042 [cs.CR].
- [56] Katharina Anna Zweig and Michael Kaufmann. A systematic approach to the one-mode projection of bipartite graphs. *Social Network Analysis and Mining*, 1(3):187–218, July 2011. ISSN 1869-5469. doi: 10.1007/s13278-011-0021-0. URL <https://doi.org/10.1007/s13278-011-0021-0>.

- [57] Joerg Reichardt and Stefan Bornholdt. Statistical Mechanics of Community Detection. *Physical Review E*, 74(1):016110, July 2006. ISSN 1539-3755, 1550-2376. doi: 10.1103/PhysRevE.74.016110. URL <http://arxiv.org/abs/cond-mat/0603718>. arXiv:cond-mat/0603718.
- [58] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):e1602548, May 2017. ISSN 2375-2548. doi: 10.1126/sciadv.1602548. URL <http://arxiv.org/abs/1608.05878>. arXiv:1608.05878 [cs.SI].
- [59] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, September 2020. URL <http://arxiv.org/abs/1802.03426>. arXiv:1802.03426 [stat].
- [60] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-Based Clustering Based on Hierarchical Density Estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-37456-2. doi: 10.1007/978-3-642-37456-2_14.
- [61] Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/15000000019. URL <https://doi.org/10.1561/15000000019>.
- [62] Joe H. Ward Jr. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963. ISSN 0162-1459. doi: 10.1080/01621459.1963.10500845. URL <https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845>. _eprint: <https://www.tandfonline.com/doi/pdf/10.1080/01621459.1963.10500845>.
- [63] Fionn Murtagh and Pierre Legendre. Ward’s Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward’s Criterion? *Journal of Classification*, 31(3):274–295, October 2014. ISSN 1432-1343. doi: 10.1007/s00357-014-9161-z. URL <https://doi.org/10.1007/s00357-014-9161-z>.
- [64] Evangelos Froudakis, Athanasios Avgetidis, Sean Tyler Frankum, Roberto Perdisci, Manos Antonakakis, and Angelos D. Keromytis. Revealing the True Indicators: Understanding and Improving IoC Extraction From Threat Reports, October 2025. URL <http://arxiv.org/abs/2506.11325>. arXiv:2506.11325 [cs.CR].
- [65] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. ChatGPT Outperforms Crowd-Workers for Text-Annotation Tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120, July 2023. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2305016120. URL <http://arxiv.org/abs/2303.15056>. arXiv:2303.15056 [cs.CL].

Appendix I - Preprocessing

Table A.1 Placeholder tokens used in BERTopic preprocessing

Category	Example pattern	Token
Bot command	/start	BOTCMD
User mention	@username	USERMENTION
Email address	user@example.com	EMAILADDR
Phone number	+1 555 123 4567	PHONENUM
Numeric token	2025, 1.2.3	NUMTOKEN
URL	https://x.ru/path	HTTPURL
Domain name	malicious.ru	DOMAINNAME
IPv4 address	192.168.1.1	IPADDR
IPv6 address	2001:db8::1	IPV6ADDR
Onion address	abc123.onion	ONIONADDR
CVE identifier	CVE-2024-1234	CVEID
MD5 hash	d41d8cd9...	MD5HASH
SHA1 hash	da39a3ee...	SHA1HASH
SHA256 hash	e3b0c442...	SHA256HASH
Ethereum address	0x...	ETHADDR
Bitcoin address	bc1...	BTCADDR
VPN/proxy config	vless://...	VPNCONFIG
Filename	payload.exe	FILEEXT
UUID	550e8400-...	UUIDTOKEN
URL-encoded string	%e4%b9%8c...	URLFRAGMENT
Hexadecimal blob	long hex sequence	HEXBLOB
Base64-like blob	long base64-like sequence	B64BLOB

Table A.2 Sensitivity of the real-token threshold

Threshold (real tokens)	Retained channel-days	Retention (%)
≥ 5	1,333,297	99.6
≥ 10	1,317,118	98.4
≥ 15	1,298,863	97.0
≥ 20	1,284,486	95.9
≥ 30	1,257,274	93.9
≥ 50	1,208,681	90.3
≥ 100	1,105,202	82.5

Table A.3 Sensitivity of the language-mass threshold

Threshold (channel-days)	Retained languages	Corpus coverage (%)
≥ 500	23	99.9
$\geq 1,000$	17	99.6
$\geq 2,000$	12	99.0
$\geq 2,500$	10	98.7
$\geq 5,000$	7	98.1
$\geq 10,000$	6	97.4

Table A.4 Mapping from attachment MIME type to file-type family (mime_family)

mime_family	MIME types matched
text_only	no attachment (text-derived MD5)
media	image/*, video/*, audio/*
text_file	text/*
document	PDF, Word (doc/docx), Excel (xls/xlsx), EPUB
archive	*zip*, *rar*, *tar*, *7z*
executable_script	PE/ELF/Mach-O binaries, MSI, shell, Python, JavaScript
octet_stream	application/octet-stream
other	any other MIME type

Appendix II - Structural Layer

Table B.1 Leiden resolution sweep summary

γ	Best quality	Mean quality	Std. quality	Communities	Largest community	Largest (%)	Singletons
0.5	18089.47	18084.41	4.31	313	3,481	14.3	0
0.6	17990.24	17985.27	3.18	315	2,862	11.8	0
0.7	17892.63	17887.69	4.90	318	2,847	11.7	0
0.8	17800.22	17793.86	5.47	319	2,847	11.7	0
0.9	17712.29	17704.66	6.14	321	2,835	11.6	0
1.0	17627.02	17622.90	4.32	324	2,835	11.6	0
1.1	17543.68	17535.45	4.17	327	2,835	11.6	0
1.2	17461.52	17457.49	4.41	327	2,835	11.6	0
1.3	17380.02	17375.52	4.48	329	2,835	11.6	0
1.4	17298.61	17294.93	3.33	329	2,827	11.6	0
1.5	17217.74	17215.07	3.09	331	2,827	11.6	0
1.6	17137.67	17135.04	1.43	334	2,827	11.6	0
1.7	17058.91	17055.70	2.69	335	2,827	11.6	0
1.8	16978.19	16976.38	1.63	342	2,810	11.5	0
1.9	16900.73	16897.87	3.14	342	2,786	11.4	0
2.0	16821.87	16819.04	3.12	343	2,786	11.4	0

Table B.2 Silhouette sweep for the K -means topic baseline over the per-channel macro-topic distributions $P(m \mid c)$

K	Silhouette	K	Silhouette
5	0.3000	25	0.5063
6	0.3257	26	0.5048
7	0.3508	27	0.5098
8	0.3701	28	0.5119
9	0.3900	29	0.5150
10	0.4175	30	0.5050
11	0.4316	31	0.5005
12	0.4557	32	0.4883
13	0.4735	33	0.4880
14	0.4849	34	0.4854
15	0.4888	35	0.4783
16	0.4590	36	0.4899
17	0.4747	37	0.4845
18	0.4644	38	0.4936
19	0.4777	39	0.4839
20	0.4843	40	0.4794
21	0.4884	41	0.5025
22	0.4962	42	0.5082
23	0.4915	43	0.4951
24	0.5009	44	0.4816

Table B.3 Silhouette sweep for the K -means temporal baseline over the per-channel hour-of-day activity fingerprints

K	Silhouette	K	Silhouette
5	0.1369	18	0.1189
6	0.1010	19	0.1165
7	0.1356	20	0.0863
8	0.1390	21	0.1187
9	0.1257	22	0.1114
10	0.1457	23	0.0959
11	0.1040	24	0.1104
12	0.0990	25	0.1120
13	0.1265	26	0.1132
14	0.0838	27	0.1167
15	0.0969	28	0.0898
16	0.1159	29	0.0977
17	0.1130	30	0.0935

Appendix III - Thematic Layer

Table C.1 Final BERTopic pipeline configuration

Stage	Parameter	Value
Embedding	model	paraphrase-multilingual-MiniLM-L12-v2
	dimension	384
	chunking	128 tokens, channel-day aggregation
UMAP	n_neighbors	30
	n_components	5
	min_dist	0.0
	metric	cosine
	random_state	42
HDBSCAN	min_cluster_size	300
	min_samples	15
	metric	euclidean
	cluster_selection_method	eom
Vectorizer	ngram_range	(1, 2)
	min_df	50
	max_df	0.40
	max_features	50,000
	stopwords	10-language ISO set + custom
c-TF-IDF	reduce_frequent_words	True
	bm25_weighting	True
Representation	model	Maximal Marginal Relevance
	diversity	0.3

Table C.2 HDBSCAN and UMAP selection

nn	nc	md	mcs	ms	Topics	Outlier
30	5	0.00	300	15	809	47.9%
30	5	0.00	250	100	846	49.7%
50	5	0.00	250	25	869	49.9%
30	5	0.00	250	25	892	50.0%
100	5	0.00	250	25	815	52.0%
15	5	0.00	250	25	712	54.4%
30	5	0.00	500	25	365	50.7%
50	10	0.00	500	25	356	54.3%
15	5	0.00	500	100	256	61.0%

Table C.3 Ward macro-topic resolution sweep on the G_1 refit (223 fine topics)

K	Max members	Singletons	Median size	Bootstrap ARI
15	50	1	10	0.730
20	50	1	8	0.702
25	35	1	7	0.690
30	24	1	6	0.733
35	24	2	5	0.745
40	24	5	4	0.757
50	24	10	3	0.768

Appendix V - Indicator Layer

Table D.1 Cross-correlation between macro-topic activity and loC appearance, per G_1 macro-topic

Macro	Peak lag (d)	r_{peak}	r_{lag0}	Pattern
13	0	0.734	0.734	synchronous
5	0	0.412	0.412	synchronous
4	4	0.353	0.285	topic leads loC
14	11	0.353	0.217	topic leads loC
2	14	0.213	0.084	topic leads loC
15	11	0.207	0.058	topic leads loC
24	1	0.185	0.183	topic leads loC
26	2	0.151	0.144	topic leads loC
7	0	0.137	0.137	synchronous
6	0	0.041	0.041	synchronous
23	0	0.001	0.001	synchronous
27	14	-0.039	-0.055	topic leads loC
1	0	-0.090	-0.090	synchronous
21	14	-0.138	-0.160	topic leads loC