POLITECNICO DI TORINO

Master degree course in Cybersecurity

Master Degree Thesis

Designing a Topic-Targeted Telegram Crawler

Supervisor
Prof. Luca Vassio

Co-supervisor:
Ing. Giordano Paoletti

Candidate
Massimo Aresca

Academic year 2025 – 2026

To my beloved grandparents, Filippo and Vera

Abstract

Telegram is used daily by millions of people to communicate. It contains a multitude of discussion topics, some of which are violent, illegal, or extremist in nature.

To monitor public chats, various crawlers have been developed. Their main problem is their ineffectiveness in finding channels of interest in a timely manner based on a search topic, as well as their inability to retrieve all groups of interest. The following thesis aims to overcome these limitations by understanding whether it is possible to develop a crawler capable of not only obtaining almost all groups of interest, but also doing so using limited resources and in a reasonable amount of time. Beneficiaries of such a tool to detect criminal activities in a timely manner may include: OSINT Researchers, Government Data Scientists, and Threat Intelligence Analysts.

The crawler proposed in this thesis uses *Latent Dirichlet Allocation* (LDA) as its topic detection algorithm, a choice due to its processing speed compared to transformer-based tools like *BERTopic*. The efficiency of LDA is tested against the *US Dataset*, which contains more than 500,000 political messages regarding the US general election in 2024. To obtain an analysis of the crawler's performance starting from different topics and with a more chaotic dataset in terms of topics covered, we also apply the *LDA*-based approach to the *TGDataset*, which contains over 120,000 channels and nearly 500 million messages, some of which were labeled by the dataset's collectors. To best optimize *LDA*, the Tree-structured Parzen Estimator is used as a Bayesian optimization algorithm for efficient hyperparameter tuning. The parent-child relationship defines the order in which the crawler explores the *TGDataset* and *US Dataset*, going from parents to children. This order is obtained by forwarding messages from one group to another. In this system, a parent is a group that contains messages forwarded from its child groups.

The crawler starts from an initial group of channels, called seeds, and then visits their child groups, where the children become the new parents, and so on until there are no children left to visit. The innovative method

adopted is to switch from a parent group to a child group only if the parent group contains a certain percentage of messages on the topic of interest. This threshold is set a priori, drastically reducing the computing resources required for crawling.

Crawling the US Dataset with a blind crawler would have retrieved 20,443 groups, with almost 50% of channels irrelevant to the topic of interest. Conversely, the proposed crawler correctly discards 4,181 channels, maintaining a recall of approximately 99.8%, while incorrectly discarding only 22 political channels. The precision of the measurement on the TGDataset, defined as the percentage of channels that actually discuss the topic of interest of the total channels collected by the crawler, yields a value of 62.5%. Recall is also measured, defined as the percentage of channels collected on the topic of interest of the total channels in the datasets that address the research topic, reaching a value of 72.8%.

Future work may include implementing Dynamic Topic Models to update the topic distribution in real time when new messages arrive, without retraining the model. Finally, multimodal analysis would extend the scope to analyze not only messages but also shared multimedia.

Acknowledgments

I am deeply grateful to Prof. Luca Vassio and Ing. Giordano Paoletti for the explanations and their availability given in addition to their expertise.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Telegram Ecosystem

Telegram is a secure messaging and audio calling cloud-based app that allows you to send messages, photos, videos, and files of any type to your contacts for free. You can also create groups for up to 200,000 people or channels for broadcasting to unlimited contacts or audiences.

The social media ecosystem is becoming increasingly fragmented: alongside mainstream networks, alternative platforms are gaining traction by appealing to distinct user communities that are highly relevant for social-science research. Telegram, a messaging app that also functions as a social network platform, is the most prominent example, with more than 800 million active users and a strong adoption in contexts where online communication is tightly restricted. In recent years, Telegram has been used to chat with other users about numerous topics, including political topics. For this reason, it is used by fringe political communities that have migrated from more regulated platforms as explored by Jost and Dogruel [2023]. As a result, Telegram constitutes a valuable data source for studying online social and political dynamics.

1.2 Platform Monitoring Challenges and Crawling Motivation

A primary challenge for Telegram crawlers is not only the massive number of channels to crawl, but also the absence of a native ranking system to evaluate the importance of the various channels. Moreover, the search bar returns

only a limited sample of groups for a given search keyword. Consequently, new channels must be discovered using snowball sampling. In this context, this technique involves starting from a set of known initial channels (seeds) and iteratively uncovering new groups by tracing the origin of messages forwarded within them. However, without a targeted filtering mechanism, this approach inevitably collects a massive volume of irrelevant channels, introducing substantial noise into the dataset.

Since the number of channels to be analyzed is extremely high, it is difficult to find procedures that are not excessively resource-intensive, have a reasonable execution time, and are, at the same time, effective in collecting almost all channels relevant to the topic of interest.

A further obstacle lies in the limitations of official Telegram APIs, which do not expose direct connections between channels. Consequently, these connections must be identified indirectly, for example, by the messages forwarded from one group to another.

Another challenge is given by the different languages used on these channels, as topic detection algorithms struggle to form topics on channels with multiple languages. Furthermore, Telegram groups frequently shift topics, making it difficult for the crawler to keep up with the new messages.

To address these platform monitoring challenges and overcome the limitations of current scraping techniques, this thesis aims to answer the following research questions:

RQ1: Is it feasible to build a Telegram crawler that uses resources efficiently while still accurately classifying political channels?

RQ2: Can the crawler be configured with different accuracy levels to balance speed and quality of results?

RQ3: Can a crawler reliably understand a prompt about a specific topic and then accurately identify the related channels while remaining topic-focused?

RQ4: How can the crawler identify the central topic of a group at runtime?

1.3 Methodologies and contributions

The first part of this work centers on the US Dataset, introduced by Blas et al. [2024]. For this purpose, a crawler was implemented that, starting from a set of seed channels and a prompt about a specific topic, incorporates a

topic detection step to keep the process focused on the subjects of interest, identifying all other related channels.

Initially, an approach using BERTopic, UMAP as a dimensionality reduction algorithm, and HDBSCAN as a clustering algorithm was employed; this was done to achieve state-of-the-art classification. Later, Latent Dirichlet Allocation (LDA) was applied and compared to BERTopic to understand whether it manages to maintain high crawling quality or whether it significantly degrades.

Furthermore, the LDA-based crawler was tested on the highly heterogeneous TGDataset containing over 120,000 channels. By targeting the *Videogame Modding* niche starting from mixed seeds, this second experiment demonstrates the tool's robustness and precision in a noisy dataset.

Chapter 2

Related work

This chapter briefly summarizes the most notable work on the subject, the main tools used in the crawler, and the datasets used.

2.1 Large-Scale Telegram datasets

Regarding the application of the crawler in sensitive and specific domains, the work by Blas et al. [2024] offers a pragmatic example focused on political discourse related to the 2024 US presidential election. Unlike general collections, the authors adopted a targeted approach, starting from a curated list of keywords related to candidates and electoral issues to identify initial chats and subsequently expanding the network through internal links until collecting more than 30,000 chats.

Blas et al. [2024] dataset plays a crucial role in our research, serving as the ground truth to validate the political filter. The analysis conducted by Blas et al. [2024] at the central nodes of the network highlighted how political narratives tend to spread through specific hubs of influence. Using their dataset as a temporal and thematic reference, we were able to verify the precision of our LDA-based classification system.

An essential reference point for the quantitative study of the Telegram ecosystem is the work of La Morgia et al. [2025], who introduced the largest public data set of Telegram channels, comprising more than 120,000 channels and nearly half a billion messages. The relevance of this study for our research is twofold. From a data collection perspective, the authors validated the effectiveness of a snowball sampling strategy based on forwarded messages. By following the forwarding metadata, they were able to reconstruct the implicit graph of connections between channels starting from a set of

heterogeneous seeds provided by third-party aggregators such as TGStat.

This graph traversal methodology confirms the validity of the crawling approach adopted in our thesis, which uses similar mechanisms for the discovery of new nodes. Furthermore, La Morgia et al. [2025] apply the LDA algorithm to categorize channels into macro-topics such as Politics, Finance, and News, demonstrating how classic probabilistic models remain a robust and scalable solution for text analysis on large data volumes. Their dataset also provides an essential basis for comparison to evaluate the ability of our crawler to discriminate niche communities from the vast background noise present on the platform.

Moreover, the Telegram ecosystem also consists of illegal channels. Morgia et al. [2023] highlighted the presence of dangerous content by revealing thousands of fake channels and fraud operations, especially in the financial and cryptocurrency fields. The widespread prevalence of noise and dangerous content justifies the necessity of filtering mechanisms adopted in the proposed crawler for the purpose of isolating the relevant arguments from the spam.

2.2 Topic Modeling on Short Text and Social Media

From a sociological perspective, the importance of monitoring Telegram is underlined by Jost and Dogruel [2023], who analyzed how radical actors make use of Telegram for mobilization during a crisis. Their work highlights how Telegram takes the role of a hub for fringe communities, making it relevant for OSINT investigations.

Furthermore, the natural evolution of propaganda strategies has been studied by Steffen [2025], the authors propose a multimodal methodology that combines textual and visual analysis in order to track conspiracy narratives, highlighting how images play a crucial role alongside text in marginal communities, although in the following thesis this multimodal approach has been avoided in order to focus more on the text to avoid computational overhead.

Turning to the crawling methodology, Silva [2024] introduced *Telegram-Scrap*, a complete tool to collect data. However, while Silva [2024] concentrates on generic scraping applications, the work proposed in this Thesis aligns more closely with that of Perlo et al. [2025], which crawls groups based on a specific topic of interest.

The application of topic modeling to the Telegram messages presents specific challenges that arise from the limited length of the documents. Cheng et al. [2014] described the problem formally as sparsity of word co-occurrence patterns, where traditional models fail because short texts lack sufficient context.

Although the state of the art in the field of natural language processing is increasingly oriented towards models based on neural networks, as widely illustrated in the context of social networking by Vassio et al., Devika et al. [2021] and Benedetto et al. [2024] in which they managed to demonstrate how models based on BERT and Sentence Transformers can significantly outperform statistical methods in capturing semantic nuances. As a means to address this problem, they proposed the Biterm Topic Model (BTM), which aggregates co-occurrence patterns through the entire corpus. Similarly, Rashid et al. [2019] explored fuzzy topic modeling approaches to mitigate the sparsity and coherence of topics in short text environments. These studies highlight the need to aggregate messages, for example, by channel or utilizing optimized models when data from chats is considered.

To shed light on the structural properties of the network formed by Telegram groups, it is important to base the analysis on consolidated metrics. For this purpose, Himelboim et al. [2017] provides a fundamental framework to classify social media networks based on the flow of information. Their work connects graph metrics such as density, modularity, and centralization to specific conversational archetypes. Applying these concepts to the Telegram graph allows us to determine whether political communities act as cohesive echo chambers or as dispersed networks.

Based on Akama and Husnaqilati [2023], the study evaluates rigorous statistical systems in the PCA algorithm for dimensionality reduction, specifically the Guttman-Kaiser criterion. However, the PCA-based approach was not used because UMAP proved to be more suitable for preserving the local structure of sparse data, such as Telegram messages, something that linear approaches like PCA lose.

Moving beyond the one-to-many structure of channels, Perlo et al. [2025] shift their attention to public groups where many-to-many interactions occur. Their study presents a horizontal analysis that compares user behavior across different topics, ranging from education to cryptocurrencies to dark web markets. To achieve this, they developed an open-source crawler based on the library. It is relevant to understand the complexity and variety of themes present in Telegram channels.

Moving beyond purely textual analysis, the work presented in Paoletti

et al. [2025] investigates the factors that determine user participation within groups, identifying social curiosity and peer influence as key factors in user participation in groups. This can be used to understand the target audience of users captured by the crawler and identify people actively interested in the topics discussed in the groups.

In the following article Barravecchia et al. [2020], the use of the Structural Topic Model to classify the quality of user-generated content was analyzed. STM wasn't adopted because the computational complexity of the model would have made the crawler excessively slow.

Barravecchia et al. serves as an illustrative reference for the practical implementation of Topic Modeling with reference to unstructured datasets. Notwithstanding its focus on the Digital Voice of Customers, the paper reveals important limitations of LDA for very short messages.

2.3 How LDA works

A comprehensive theoretical overview of LDA is given by Blei et al. [2003] Latent Dirichlet Allocation tries to find a solution to this problem, given a collection, also called a corpus of documents, in our case, telegram messages. Each message must be associated with a particular topic, such as science or politics. The problem is that the topic associated with each document is not known in any way without manually reading the message. LDA places the documents in the corners of a simplex where each vertex represents a topic in a way that documents are close to the corners of the topics that most represent them; for example, if a document collocates to an edge in the middle of two vertices, it means that it belongs to both topics. In simple terms, the aim is to map the documents in their correct position in the simplex. LDA is a generative algorithm; it means that it automatically generates documents. In most cases, these documents are random documents that do not mean anything. In other cases, with a small probability, it can get documents similar to the original documents. If, for example, there are many machines and they give some fake documents, comparing them with the original document, the machine that gives a document that is more similar to the original one means that it has better settings than the other machines. The goal of the algorithm is to set a machine that best gives as output the original documents.

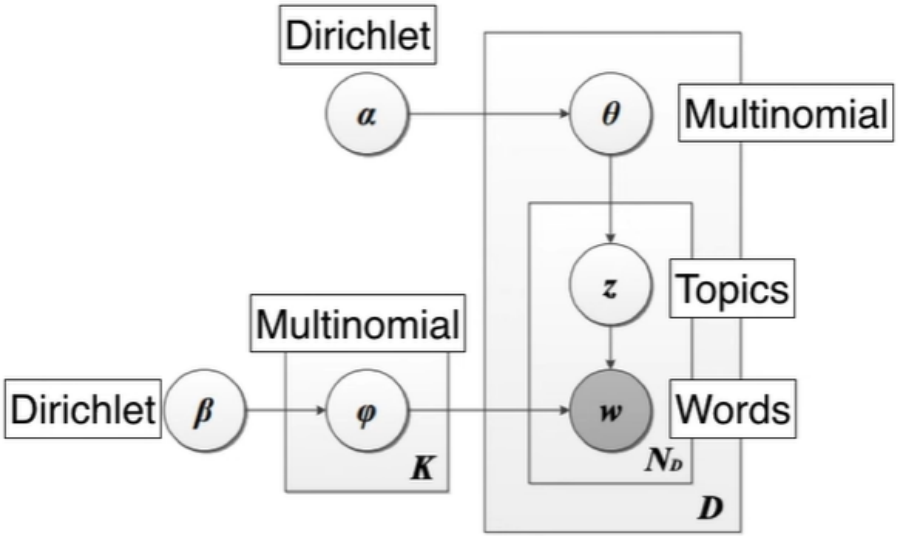

Figure 2.1: Overview of the LDA structure.

- α is the parameter that generates the Dirichlet distribution
- θ is the Dirichlet distribution that generates α
- β is the parameter that generates the Dirichlet distribution
- φ is the Dirichlet distribution that generates β
- z means what topic has generated this word
- w is a single word token
- D block represents the fact that θ, z and w are repeated for every document.
- K block means that exist an instance of φ for every topic.
- N_d means that both z and w must be repeated for every word of the document.

During training, the following formula is used to maximize.

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \times \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}}) \tag{2.1}$$

In this formula:

- $W_{j,t}$ is the word in the document j in position t
- W is all the observed text, more formally $W = \{W_{j,t}\}$.

- $Z_{j,t}$ is the topic assigned to the word in position t in document j, so it is a number between 1 and K
- Z is the collection of all the topics assigned to each in every document formally $Z = \{ Z_{j,t} \mid j = 1, \ldots, M;\ t = 1, \ldots, N_j \}$
- M is the number of documents
- K is the number of topics
- N_j is the number of words in document j
- α is the hyperparameter for the Dirichlet distribution of topics in a document
- θ_j is the Dirichlet topic distribution of the document j, formally

$$\theta_j \sim \text{Dirichlet}(\alpha).$$

- θ is the collection of Dirichlet topic distributions for every document formally $\theta = \{ \theta_j \mid j = 1, \ldots, M \}$
- β is the hyperparameter for the Dirichlet distribution of words in a topic
- φ_i is the Dirichlet word distribution of topic i; formally

$$\varphi_i \sim \text{Dirichlet}(\beta).$$

- φ is the collection of Dirichlet distribution of words for every topic formally $\varphi = \{ \varphi_i \mid i = 1, \ldots, K \}$

This formula represents the joint probability of W, Z, θ, φ, α and β. To maximize this probability during training, the procedure alternates between two steps:
φ and θ_j are called latent variables because they are not observable but must be estimated, while α, β, and $\varphi_{k,w}$ are the parameters of the models.

- The E-step takes the φ fixed and the observed word W and updates the latent variables θ_j and $Z_{j,t}$ to estimate them.
- The M-step takes the output of the E-step and, with that, updates the distribution of the words of topic k, namely $\varphi_{k,w}$, usually α and β, are fixed.

They are iterated until convergence.

In the `scikit-learn` implementation of LDA, an approximate version of this formula is used.

2.4 How BERTopic works

In order to extract meaningful topics, BERTopic is used as the state of the art for topic detection as cited in Mazzei et al. [2022]. BERTopic relies on sentence transformer models in order to convert messages into meaningful vectors that capture the meaning of sentences. These sentence transformers are based on transformers. This representation is useful because it is possible to calculate the similarity of sentences with respect to each other by means of cosine distance.

Another component of BERTopic is the dimensionality reduction algorithm:
One option consists in using PCA which stands for Principal component analysis, but the most popular and power one for capturing non linear patterns is given by UMAP as state in McInnes et al. [2020], it is a non linear algorithm that works by comparing the spatial proximity between the vectors representing the document emebeddings in the embedding space and reducing the dimensionality keeping the all the relevant original patterns. An important parameter is *n_neighbors* that balances the importance of the local structure with the global structure. The purpose is to reduce the dimensionality and reduce the sparsity of the embeddings in order to have a faster execution time.

In order to group the embeddings into clusters, a clustering algorithm is needed. By default, BERTopic uses HDBSCAN, which is great at finding clusters with different densities.
The two key HDBSCAN parameters are the following:

- `min_cluster_size`: it works by setting the number of data points needed to constitute a cluster.
  If the value is too small, the risk is to end up with an unstable and noisy cluster, while setting too high a risk to merge clusters that should remain separate.

- `min_samples`, by looking at how far apart the points are from their neighbors, it sets how strict the algorithm should be in cluster creation. Increasing the value, the algorithm only forms clusters in zones where the points are very close to each other, resulting in a more conservative clustering.

Then the vectorization phase is performed, after the clustering phase ends,

the vectorizer is responsible for converting the text into numerical features that are useful for topic analysis.

In the end, BERTopic uses c-TF-IDF to generate the top-N words for each topic. It works by concatenating the documents in each cluster, then creating a BOW for each cluster where the number of times the word w appears in cluster C_i is registered with the formula:

$$C = \{C_1, \ldots, C_K\}$$

$$\mathrm{tf}_{i,w} = \sum_{d \in C_i} f(w, d)$$

Then a global IDF is created on the totality of clusters with the following formula:

$$\mathrm{df}(w) = |\,\{i \in \{1, \ldots, K\} \;:\; \exists\, d \in C_i \text{ such that } w \in d\}\,|$$

$$\mathrm{idf}(w) = \log\left(\frac{K}{1 + \mathrm{df}(w)}\right)$$

Finally, the final formula for the importance of each word in a topic is given by the formula:

$$\mathrm{cTFIDF}_{i,w} = \mathrm{tf}_{i,w} \cdot \mathrm{idf}(w)$$

$$\mathrm{TopWords}(C_i) = \mathrm{argsort}_w(\mathrm{cTFIDF}_{i,w})_{1:n}$$

2.5 Advantages of LDA over Neural Topic Models

At the moment, the state of the art in the field of *Natural Language Processing* is moving toward models based on neural networks. Devika et al. [2021] and Benedetto et al. [2024] demonstrated how models built upon BERT and Sentence Transformers significantly outperform statistical methods in capturing fine-grained semantic nuances. Specifically, Benedetto et al validated the effectiveness of BERTopic to summarize social-media posts. Additionally, Viegas et al. [2025] explored the integration of contextual embeddings in the context of hierarchical topic modeling; they achieved this by critically analyzing the limits of traditional metrics such as coherence.

Latent Dirichlet Allocation was selected for the crawling module developed in this thesis. This strategic decision is not only based on the established existing literature, but is also supported by a critical assessment of the trade-offs between performance, stability, and resource consumption, as highlighted in recent comparative studies such as Abdelrazek et al. [2023].

This decision is further supported by Ogunleye et al. [2023], who observed that although BERTopic produces superior coherence scores, the computational overhead caused by transformer-based models is substantial. The determining factor for a crawling system, which must iteratively process millions of messages, is computational efficiency. As demonstrated by Abdelrazek et al. [2023], although neural models such as ETM or CTM offer advantages in terms of semantic diversity, they entail a significantly higher computational cost than classical probabilistic models. LDA has proven to be one of the most efficient algorithms in terms of runtime, a prerequisite for maintaining scalability without requiring prohibitive GPU-based hardware infrastructure for each crawler iteration.

A second critical aspect concerns the reproducibility and reliability of channel classification. Topic modeling surveys such as Abdelrazek et al. [2023] highlight the existence of an inverse relationship between stability and the diversity of generated topics: while neural models tend to excel at generating highly diverse topics, they suffer from marked stochastic instability between different runs. For an automated application that must constantly discriminate between political and non-political channels, the stability offered by LDA ensures greater consistency compared to the fluctuating nature of NTMs, reducing the risk of the same channel being classified differently in separate runs.

Finally, the operational robustness of LDA was preferred over the configuration complexity of models based on Variational Autoencoders. As discussed in depth in Wu et al. [2024], NTMs exhibit significant sensitivity to hyperparameters and are often subject to topic collapsing, where distinct topics collapse into identical semantic representations, or to the generation of trivial topics composed predominantly of stop words. In contrast, LDA offers more predictable and robust behavior out of the box, avoiding the optimization complexities that, in a dynamic context like Telegram's, could introduce noise and inaccuracies that are difficult to automatically diagnose.

Therefore, while tools like BERTopic have been reserved for the high-precision offline analyses described below, LDA represents the sweet spot between speed and reliability required for the crawler's real-time discovery engine.

2.6 Tree-Structured Parzen Estimator for LDA Hyperparameter Optimization

In the context of topic modeling using Latent Dirichlet Allocation, the choice of hyperparameters, such as the number of topics K, the Dirichlet priors α and β, plays a crucial role in the quality and consistency of results. Manual search and traditional grid search are often inefficient approaches due to the high dimensionality of the search space and the computational cost required to evaluate each configuration. To overcome these limitations, this work adopted the Tree-Structured Parzen Estimator, a Bayesian optimization algorithm extensively analyzed by Watanabe [2025]. As pointed out in Panichella [2021], LDA delivers remarkably variable performance based on the quality of the hyperparameter configuration, and it further emerges that suboptimal hyperparameter choice leads to suboptimal results, thus further reinforcing the importance of automatic hyperparameter tuning.
The importance of this study lies in the empirical demonstration of the superiority of TPE over random search, especially in contexts where the objective function is expensive to evaluate and the parameter space includes discrete and conditional variables. The adoption of TPE allowed us to automate the tuning of the LDA model within the crawler, focusing exploration on the most promising regions of the hyperparameter space. Consequently, we were able to maximize the coherence metric C_v in a time frame compatible with the requirements of a real-time data collection system, ensuring that the extracted topics maintained a distinct semantic separation, even in the presence of typical Telegram message noise.

2.6.1 How TPE works

In order to choose the hyperparameters for a particular model, it is possible to avoid trying all candidate values that maximize or minimize one or more metrics; this approach can become too slow for resource-intensive tasks. In order to make the hyperparameter choice more efficient, a possible solution is given by TPE, which stands for Tree Parzen Estimation.

It is implemented in `Optuna` as described in Watanabe [2025]. `Optuna` is an advanced automatic hyperparameter tuning framework, designed specifically for machine learning. It is open source and easy to use. It can be consulted with the following link: Optuna documentation. the key idea of TPE is instead of evaluating P(y|x) where:

- x = value of the single hyperparameter
- y = the loss

P(x | y) is evaluated, which is the probability of observing the hyperparameter x given the loss y. It is done by fitting the surrogate model P(y|x) to the previous observations and then using it to search and predict areas that are encouraging in the search space.

In particular, two surrogate models are used:

- P(x | y > y*) = P(x | bad) which corresponds to the trials where the loss y is worse than the threshold $y*$
  , as is the distribution of the *bad* values.
  It is usually a large and sparse distribution because the values of x that are worse than the threshold are more than the good ones concentrated in particular areas, and it is sparse because the bad values do not concentrate in particular areas like the *good* ones.
- P(x | y < y*) = P(x | good) which corresponds to the trials where the loss y is better than the threshold $y*$
  , and so is the distribution of *good* values.

The threshold y^* is determined by a hyperparameter γ, so, for example, if there are 100 observations and if γ is set to 0.5, only the best 50% observations are used to fit the good distribution and the worst 50% to fit the bad distribution.

When searching for a new value for x, the algorithm will choose a value for which: P(x|good) / P(x|bad) is high. In fact, a promising candidate is likely to:

- have low probability under the *bad* distribution P(x|bad)
- have high probability over the *good* distribution P(x|good)

In Bergstra et al. [2011] it is shown that the promisingness score given by P(x|good)/P(x|bad) is actually proportional to the expected improvement.

In TPE for real-valued or integer hyperparameter types, the Mixture of truncated Gaussians is used for each sample.
Then the mixture of probability density in a particular point is given by the sum of probabilities from each Gaussian distribution, and then normalized by the number of distributions in the mixture.

Here are the main advantages of using TPE:

- Data efficient, which means that in order to find good hyperparameters, only a few trials are needed.
- Faster than a random search for what is said above

Chapter 3

Methodologies

This chapter outlines the methodological framework developed to construct and evaluate the topic-targeted Telegram crawler. First, we present the primary data sources: the US Dataset and the TGDataset. The US Dataset serves as the first case study, on which the crawler's logic is developed and refined, while the TGDataset is used as a second, more challenging case study to evaluate the tool's robustness in a highly chaotic and heterogeneous network. Before applying any analysis, all the messages analyzed during the crawling process must be cleaned. Because our automated crawler relies on two distinct tools working together: LDA, a statistical topic detector, and a Large Language Model (LLM) to classify topics as relevant to the researched topic, we implemented a dual-track text preprocessing pipeline. One track aggressively filters out structural noise such as URLs and stopwords, while the second track preserves the natural sentence structure so the LLM retains the proper context to understand and classify the conversations. With the data prepared, the chapter describes the core components of the crawler: the topic detection algorithm, the classification mechanism, and the threshold-based traversal strategy. Following the architectural description, a series of validation experiments justify the key design choices: the selection of LDA over BERTopic, the reliability of the LLM classifier, and the optimal threshold value. Finally, the specific experimental setups for both the US Dataset and the TGDataset are detailed.

3.1 Data Sources

This section describes the two primary datasets used for the experiments. The crawling process is first evaluated on the US Dataset, serving as a controlled environment, and subsequently applied to the more heterogeneous TGDataset.

3.1.1 2024 U.S. Presidential Election Dataset

In Blas et al. [2025], a database about the US election based on the Telegram API has been constructed to collect groups about the 2024 US presidential election.
It has been created in order to study topics such as radicalization dynamics, disinformation, and political trends on Telegram groups.
It is a huge database that contains over 30,000 chats (i.e, both channels and groups) and more than 500 million messages starting from the first of November 2023 to the end of 2024, with messages and metadata of the chats.
Among the most important fields, a message from a chat contains the following:

- `message_id`: the unique message identifier, used to uniquely identify messages during the crawling process
- `message`: the text of the message
- `message_date`: the timestamp of the message, useful to have temporal information about messages

3.1.2 TGDataset

The TGDataset stands as the largest public repository of Telegram data, providing an unprecedented scale for analyzing the platform's ecosystem as described in La Morgia et al. [2025]. The dataset gives a global overview of the entire Telegram ecosystem. The authors started from TGstat to get the first seed channels, then for each seed channel, they downloaded all the messages using the `Telethon` APIs, excluding videos, images, or PDFs in order to avoid illegal content or subject to copyright. Anyway, the metadata about the media content is still retrieved using the Telegram API. Then all the forwarded messages are analyzed, and if the original author of the forwarded message represents a channel not already present in the list, it is

added. Of the newly found channels, the messages are collected as described above; this procedure is continued iteratively.

It has a large collection of 120,979 public Telegram channels. It contains a total of 498,320,597 messages. The channels are divided into 121 JSON files and 4 directories. The folders are alphabetically sorted based on the username of the channel to which they refer:

- `TGDataset_1`: here are the channels from `A` to `freeJul`
- `TGDataset_2`: here are the channels from `freejur` to `NaturKind`
- `TGDataset_3`: here are the channels from `Naturmedi` to `theslog`
- `TGDataset_4`: here are all the remaining channels.

This ordering enables the analysis of channels in batches rather than considering them all together.

The following fields are stored for each channel:

- `channel_id`: numerical identifier of the channel
- `creation_date`: timestamp of creation of the channel
- `username`: public username of the channel
- `title`: title set by the owner;
- `description`: channel description
- `scam`: it is a flag that represents the fact that Telegram classified the channel as a scam or not
- `verified`: it is a flag that indicates if the channel is verified or not
- `n_subscribers`: the number of subscribers
- `text_messages`: list of textual messages published
- `generic_media`: multimedia elements shared in the channel

The fields related to individual channels are useful for obtaining additional information on the channels to evaluate different crawling strategies.
The following information is stored for each text message:

- `message`: the content of the message

- `date`: the timestamp of the message
- `author`: the ID of who posted the message
- `forwarding information`:
  - `is_forwarded`: a flag that indicates if the message was forwarded
  - `forwarded_from_id`: the ID of the person that forwarded the message
  - `forwarded_message_date`: the timestamp of the first post of the message

Knowing the information related to each message is useful during the crawling process to better classify them and place them on a timeline.
The following metadata are present for each media:

- `title`: the title
- `media_id`: the ID of the media
- `date`: the timestamp of the media
- `author`: the author of the media
- `extension`: the file extension of the media

The metadata enriches what we already know about groups, which is useful for possible future work.

3.2 Data Preprocessing

The preprocessing phase is fundamental to filter out noise and reduce the high dimensionality of raw Telegram messages. To serve the distinct needs of our pipeline, the text undergoes two parallel preprocessing tracks designed for the subsequent steps: one heavily filtered to optimize the Latent Dirichlet Allocation (LDA) algorithm detailed in Section 3.3.1, and another minimally processed to preserve the natural language context for the Large Language Model (LLM) used in Section 3.3.2.

For the LDA topic detection, the preprocessing consists of the following phases:

- Every text is converted to lower case
  It reduces the variance of the vocabulary; in fact, LDA works better if there are not many different forms of the same word, reducing the sparsity.

- strange characters or accented characters are converted to the base ASCII, for example: à becomes a, ñ -> n, and so on.
  This step is necessary to reduce the variance of the vocabulary and make the text more suitable for LDA.

- the mentions and hashtags like or # are removed.
  In fact, they are not so useful for understanding the general context of the message, and they are very specific to a particular user, so removing them reduces the noise given to LDA.

- All punctuation is removed, this means that:
  All punctuation marks are substituted with a space
  All line breaks and multiline is substituted with a space
  Remove all multiple spaces at the borders and collapse multiple spaces
  This removes redundant characters that are not semantically useful.

- remove stopwords
  Doing so removes the most frequent stopwords such as "the," "a," "is," "and," in," "on,"
  It is done because the stopwords do not bring much semantic meaning, but are very frequent.

- remove short words, all words between one and three characters are removed
  These words usually do not give much semantic meaning and just increase the noise for LDA.

- convert verbs to their base form
  Gerunds and conjugated verbs are reduced to their root or infinitive form (e.g., via lemmatization). This group words with the same semantic meaning together, reducing the vocabulary size and the dimensionality given to LDA.

- only English messages are kept, it is achieved thanks to the `langdetect` python library Danilák [2014]
  Same topics with the same meaning but different language (i.e., different words) appear in different clusters

For the LLM, we preprocessed text in order to keep the highest possible level of context. Only the following filters are done:

- only English messages are kept
  to maintain consistency with the LDA preprocessing track and ensure uniform data analysis across the entire pipeline.
- repeated characters are removed
  For example: the words veeery gooood and very good have the same semantic meaning, like also hello word! and hello word!
- text is truncated to 500 characters
  The original text is truncated to a maximum of 500 characters. This ensures the prompt remains within the token limits while preserving enough original context for the LLM to read a realistic example of the conversation.

All these steps are done to highlight the key terms, the most useful ones for LDA, and to give the best possible context for an LLM.

3.3 Crawler Architecture

With the data prepared, this section describes the core components of the automated crawler: the topic detection algorithm, the classification mechanism, and the threshold-based traversal strategy.

3.3.1 Topic Detection with LDA

LDA is used for the topic detection phase, as it is much faster than BERTopic. Two possible versions of LDA are implemented within the sklearn library. Here are the original papers for more details, Hoffman et al. [2010], Hoffman et al. [2013], and the `Gensim` version described in Griffiths and Steyvers [2004]. The `Gensim` implementation of LDA is preferred over the sklearn counterpart for the following reasons:

- the `Gensim` library uses collapsed Gibbs sampling, which is often better on short texts than the `scikit-learn` implementation that uses variational Bayes, as cited in Griffiths and Steyvers [2004].
- the `scikit-learn` library has difficulties in parallelization, and no particular performance changes are found when increasing the parallelization parameter n_jobs.

To determine the optimal number of topics for the LDA model, we employ Bayesian Optimization using TPE, detailed in Section 2.6. The optimization objective is to maximize the Coherence score (C_V). To strictly minimize the computational overhead during the hyperparameter tuning phase, the optimization trials are evaluated on a randomly sampled 10% subset of the documents. Once the optimal value for k is identified, the final LDA model is trained on the complete dataset to ensure maximum thematic accuracy.

Once topics are generated by LDA, they must be automatically classified as relevant or irrelevant to the research scope.

3.3.2 LLM-based Topic Classification

For the purpose of creating a fully automatic crawler, the topics generated by LDA must be classified as relevant or irrelevant by an LLM.

In order to classify the topics as political or non-political, we used a definition where a topic is considered political if it could be discussed during a US presidential debate between candidates.

For each topic generated by LDA, the LLM receives a structured prompt containing: the 20 most significant keywords of the topic, the 3 most representative messages (i.e., those with the highest probability of belonging to that topic), and 3 randomly selected messages assigned to the same topic. The representative messages provide the clearest examples of the topic's content, while the random messages help the LLM verify that the topic classification is consistent across a broader sample.

GPT-5-Nano was selected as it is cost-effective in terms of tokens while being effective in classification, as demonstrated in Section 3.4.3.

With topics classified, the crawler must decide which channels to explore.

3.3.3 Threshold-based Crawling Strategy

The original strategy proposed in this thesis is to define a political channel as a channel with a given percentage of political messages; a message has to be considered political if it belongs to a political topic with respect to the following definition: all those topics that could be discussed during a US presidential debate between candidates and presidents. To navigate the network, the crawler uses the parent-child structure of the datasets. In the US Dataset, children are identified through metadata like direct links, mentions, or forwarded posts found within the parent. In the TGDataset, children are identified exclusively through the `forwarded_from_id` field of each message.

A channel is defined as a political channel if at least a given percentage of its messages belong to a topic classified by the LLM as political. In order to reduce the resources required and make the crawling more efficient, only the children of the groups with at least a given percentage of political message compared to the total number of messages are inserted in the next level of groups to be analyzed; this approach resembles a breadth-first search approach, where one level is completed before analyzing the next one.

In summary, the crawler analyzes each level before moving on to the next, stopping as soon as it finds a level without groups. This means that the parent groups at the previous level have not led to new groups or that the new groups do not contain a sufficient threshold of political messages. Consequently, the crawler avoids analyzing their children and does not collect them. Additionally, this strategy is computationally efficient and significantly reduces the resources required for crawling. The quantitative validation of this threshold strategy is detailed in Section 4.2.1.

These components are assembled into a fully automated pipeline that processes each level of the network before advancing to the next.

3.3.4 Automated Pipeline

All previous components are concatenated to obtain an automated LDA-based crawler that automatically generates topics, classifies them using the chosen LLM model, and tags messages as political or non-political based on whether they belong to a political topic.

The following list summarizes the steps performed by the crawler for each level:

1. Preprocessing: language detection and dual-track text cleaning. As previously detailed, messages are heavily filtered for LDA topic detection, while the natural language structure is preserved for LLM classification.

2. LDA Training: hyperparameter optimization via TPE in order to identify the best possible parameters in a reasonable amount of time, according to the previously discussed metrics.

3. LDA inference: LDA is applied to all the messages of the current level for the topic detection, then each message is assigned to a specific topic.

4. Topic Classification: the LLM is used to classify topics as political or non-political, as a human would.

5. Politics Ratio: In this step, the number of messages assigned to a political topic is calculated and divided by the totality of messages in the group.

6. Group classification: In this phase, groups are classified as political groups if their ratio of political messages to the total number of messages exceeds a given threshold decided at the beginning of the crawling process.

7. Next-level group creation: in this phase, only the children of parents classified as political are included in the groups to be analyzed at the next level.

8. Iteration or termination: if there are no child groups of political parents, the process stops, and the parent groups are collected; otherwise, the process is repeated for all the previous steps.

The same pipeline is applied to the TGDataset, with the only difference being the prompt provided to the LLM; this change allows for testing a new topic, leveraging the higher heterogeneity of the TGDataset compared to the US Dataset.

3.4 Design Validation

Having described the crawler's architecture, this section presents The experiments conducted to validate the key design choices: the selection of LDA over BERTopic, and the reliability of the LLM classifier, and the optimal threshold value.

3.4.1 BERTopic Baseline

In this initial phase, the primary goal is to extract the most refined and coherent topics from the US Dataset; these results will serve as a high-quality benchmark to assess whether LDA is capable of achieving comparable performance.

To achieve the best possible topic creation, the BERTopic algorithm, a state-of-the-art topic detection algorithm as stated in Devika et al. [2021], is used. Starting from the initial seed channels, we utilized BERTopic to achieve state-of-the-art topic detection as stated in Mazzei et al. [2022]. Specifically,

we configured its underlying UMAP and HDBSCAN components for dimensionality reduction and clustering, respectively. First, of all three different sentence transformers, namely all-distilroberta-v1, paraphrase-MiniLM-L6-v2, and all-MiniLM-L6-v2. They are used to create a semantic embedding of the preprocessed messages. Dimensionality reduction was performed using UMAP. Feasible ranges for the UMAP parameters were selected according to recommendations in the official documentation. All possible combinations of the following hyperparameter values were evaluated:

$$n_{\text{components}} \in \{3, 5, 10\}, \qquad n_{\text{neighbors}} \in \{5, 25\}, \qquad \text{min\_dist} \in \{0.0, 0.1\}.$$

The same is done for the HDBSCAN *min_cluster_size* hyperparameter. All possible values of the hyperparameter have been tested and are compared with all possible previous UMAP combinations:

$$\texttt{min\_cluster\_size} \in \{3, 5, 10\}.$$

Hence, for each of the three sentence transformers cited above, all of these UMAP and HDSCAN combinations are used. To accelerate computation, CPU parallelization with the Parallel executor of the `joblib` Python library is used to leverage the cluster's full parallel computing power to evaluate different combinations of UMAP and HDBSCAN hyperparameters.

To choose the best sentence transformer and the optimal hyperparameter values, we calculate the following three key metrics:

- Coherence;
- Diversity;
- Silhouette.

The hyperparameter configurations achieving the highest scores for each individual metric, Coherence, Diversity, and Silhouette, as well as the one maximizing the normalized mean, are isolated. This subset of top-performing configurations is then manually reviewed by two researchers to determine the most suitable final setup based on qualitative thematic consistency.
The resulting optimal configuration is shown in Section 4.2.2.

The diversity score measures how distinct topics use different words, quantifying their semantic separation based on the vocabulary used in each.

The `OCTIS` Python library, as stated in Terragni et al. [2021], implements the diversity score, which is selected for its robustness to noise and superior

stability. Here is the formula:

$$\mathrm{TD} = \frac{1}{K \cdot T} \sum_{i=1}^{K \cdot T} \mathbf{1}\big(\#(x_i) = 1\big)$$

The silhouette score measures how well the clusters are separated from one another by quantifying how much a topic's points are near each other while also being farther from the points of the other topics. For this, the standard sklearn library version is used. Here is the formula:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i),\, b(i)\}}$$

The coherence score measures how often the top words of a topic co-occur in the same documents. If these words are distributed randomly throughout the corpus rather than appearing together, this indicates that the topic lacks semantic consistency, leading to a low value.

The `OCTIS` library was utilized for coherence analysis, employing the C_v metric. This measure provides a more comprehensive representation that aligns more closely with human judgment, as demonstrated in Röder et al. [2015] and Wu et al. [2024].

Here, this C_v formula is used:

$$\mathrm{NPMI}(x_i, x_j) = \frac{\log\left(\frac{p(x_i,x_j)+\varepsilon}{p(x_i)\,p(x_j)}\right)}{-\log\left(p(x_i,x_j)+\varepsilon\right)}.$$

$$\mathbf{v}_{\mathrm{NPMI}}(x_i) = \left[\,\mathrm{NPMI}(x_i, x_j)\,\right]_{j=1}^{T}.$$

$$C_V = \frac{1}{T}\sum_{i=1}^{T} \cos\Big(\mathbf{v}_{\mathrm{NPMI}}(x_i),\ \mathbf{v}_{\mathrm{NPMI}}(\{x_i\}_{i=1}^{T})\Big)\,.$$

With this high-quality baseline established, we now verify that LDA produces results comparable to BERTopic despite its lower computational cost.

3.4.2 LDA vs BERTopic Comparison

Although BERTopic has the best possible topic creation, it is computationally expensive and requires too much time for practical use. To reduce computational time, LDA was used as a lighter alternative to BERTopic as discussed in Abdelrazek et al. [2023].

To ensure that LDA and BERTopic agree on how to classify messages, a contingency table was used to represent the topics created by LDA and BERTopic, where each point on the two axes corresponds to a topic created by LDA and BERTopic, respectively. The contingency table counts how many documents fall into each combination; therefore, each LDA topic and each BERTopic topic. This approach evaluates the degree of agreement between LDA and BERTopic regarding document classification. If both see the same topics, then it is expected that if LDA places certain messages in topic X, then BERTopic will also place them in a single topic Y. If, on the other hand, the documents from a particular LDA topic are scattered across many BERTopic topics, it means the two methods capture different aspects of the same text. Since each topic in each method has a different number of associated documents, to better interpret the correspondence between the topics, the rows and columns are normalized. In addition, we evaluate the Adjusted Mutual Information (AMI) score and the Adjusted Rand Index (ARI) to quantify the agreement between the two methods. The final scores demonstrating LDA's validity are reported in Section 4.2.3. AMI measures how much knowledge of the topic assignment from one method reduces uncertainty about the assignment from the other. If the documents of a given LDA topic consistently fall into a single BERTopic topic, the uncertainty is low, and AMI is high; if instead they are scattered across many BERTopic topics, the uncertainty is high, and AMI is low. ARI takes a different perspective based on pairwise comparisons: for every possible pair of documents, it checks whether the two methods agree on placing them in the same topic or in different topics. A high ARI means that most pairs are treated consistently by both methods. Both metrics are adjusted for chance, meaning that a value of 0 corresponds to random agreement and 1 to perfect agreement. They are also independent of the number of topics and of the cluster sizes, making them suitable for comparing LDA and BERTopic, which produce different numbers of topics.

Having confirmed that LDA is a valid replacement for BERTopic, we verify that the LLM can reliably classify the topics it generates.

3.4.3 LLM Classification Validation

All topics are initially classified by two researchers who have to agree on every topic whether it is political or not. The following analysis is performed to verify that the LLM's classification closely aligns with human judgment.

We tried all the following LLM models, from the least token expensive to

the most token expensive:

- GPT-5-Mini
- GPT-5-Mini-2025-08-07
- GPT-5-Nano-2025-08-07
- GPT-5-Nano

To assess the quality of the LLM classification, we compare it with the ground truth provided by human classification. A comprehensive breakdown of these performance metrics is provided in Section 4.2.4. The following metrics were calculated using the topics generated by LDA, based on the seed nodes of the US Dataset.

- Recall (TPR) = $\frac{\text{TP}}{\text{TP+FN}}$
- Precision = $\frac{\text{TP}}{\text{TP+FP}}$
- F1-score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}}$
- Accuracy = $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
- Specificity (TNR) = $\frac{\text{TN}}{\text{TN+FP}}$

Starting from the recall, it is used to measure how much the LLM is able to classify as political all topics classified as political by the ground truth; therefore, a high value close to 100% is the proof of how the LLM is able to miss only a very small number of political topics. Instead, precision measures how reliable the model is in classifying a political topic; a high value indicates that the model avoids incorrectly classifying a non-political topic as political. Regarding the F1-score, which is the weighted average of Recall and Precision, it provides a single value that represents the overall quality of the classification. In the experiment, accuracy is defined as the percentage of correct classifications relative to the total, including both positive and negative classifications. In short, it indicates how many times the LLM correctly classified compared to the total number of classifications. Finally, specificity measures how well the LLM model correctly classifies non-political topics.

3.5 Experiments

With these design choices validated, the following sections detail the experiments conducted on both datasets.

3.5.1 US Dataset Experiments Details

This section describes the methodology used for the US Dataset. We evaluated whether the crawler can correctly filter political groups from non-political groups across different thresholds in a reasonable time, and how it identifies specific discussion topics at particular time intervals. From the US Election 2024 Dataset, only the messages from the first of July to the 31 of August were taken for simplicity. To determine how precision and recall change, the following thresholds were tested: 20%, 40%, 60%, and 80%. The messages were collected from July 1st to August 31st 2024, when most were concentrated, as shown in Figure 4.2. We conduct these experiments to understand how different thresholds affect the capture rate of political versus non-political messages and channels. The comparative results are summarized in Section 4.3.1. We conduct an additional experiment to determine how many of the groups excluded from the crawling process are actually political groups. The results of this coverage analysis are discussed in Section 4.3.3.

These experiments are key to determining which threshold is most appropriate for a given purpose, whether to gain a general overview of the topics discussed on Telegram or to conduct a more in-depth analysis of individual political channels. Once a threshold that offers a fair balance between time, computational resources, and accuracy is found, it is applied to the entire time period of the dataset, from November 2023 to October 2024, as discussed in Section 4.3.3. We conducted a further experiment to compare the time period from July 15th to August 15th with the period from the beginning to the end of March. This experiment aims to verify whether the crawler can recognize specific discussion topics from a selected time window, as demonstrated in Section 4.3.2.

Having defined the US Dataset experiments, we now describe the setup for the more challenging TGDataset evaluation.

3.5.2 TGDataset Experiments

The TGDataset provides a wide range of Telegram channels and groups. The authors of the TGDataset used LDA to label a subset of the groups; one of

these labels is *Videogame Modding*. The crawler targets *Gaming* in general to verify its ability to identify and retrieve specific sub-communities, such as *Videogame Modding*, even when starting from a broad thematic category. In the TGDataset, identifying channels with a very high percentage of messages about a specific topic is more difficult, so lower thresholds are used.

TGDataset Graph Analysis

To obtain more information about the connections among channels, we utilized the `NetworkX` library described in Hagberg et al. [2008] on the unweighted directed graph constructed from the dataset's channels. The resulting topological metrics are presented in Section 4.4.1. Each node represents a single channel, and each edge from A to B represents that channel A has forwarded messages belonging to channel B to its chat.

All channels are subjected to the dual-track preprocessing detailed in Section 3.2; thus, only channels with at least 10 English messages are selected to avoid running LDA on fewer than 10 documents. This filtering ensures that the analyzed graph perfectly matches the actual network the crawler navigates during the experiments. Only edges between valid channels are used, where a channel is valid if it contains at least 10 English messages. To this end, the following metrics are calculated:

- Density: Ratio between existing edges and possible edges. A low value indicates a sparse graph, while a value close to 1 represents a nearly complete graph where each node is connected to all other nodes by an edge.
- In-degree: The in-degree of vertex A indicates how many channels have forwarded messages from A into their group.
- Out-degree: The out-degree of a vertex A is calculated as the number of channels, other than A, from which A has forwarded messages into its own chat.
- WCC (weakly connected components): A group of nodes that are mutually reachable regardless of the direction of the edges.
- SCC (strongly connected components): A group of nodes that are mutually reachable from each other by following the directions. This can be significant, as the crawler may miss entire sets of groups if it does not visit any groups in a given SCC.

- PageRank: measures the importance of a node based on the quality of those who cite it via message forwarding. It can be summarized as the idea that being cited by important channels is more valuable than being cited by unknown channels, where being cited means that other channels have forwarded messages from the cited group to their channels.

- HITS: measures authorities; their score increases as their in-degree increases; a high score indicates that messages from those groups are forwarded to many groups. On the other hand, the score attributed to hubs increases with their out-degree; thus, a high score indicates that they forward messages from many groups. Therefore, authorities produce original and valuable content, while hubs select and collect quality content.

- Reciprocity: measures the percentage of bidirectional edges; therefore, in the case of directed graphs like this one, it means that a bidirectional edge is a cycle of length two between two edges that forward messages to each other.

- DAG: Checks whether the graph is a directed acyclic graph or whether there are cycles in the graph.

TGDataset Seed Creation and Crawling Setup

For the TGDataset, to generate the seed groups, Stratified Random Sampling was used. For the *Videogame Modding* label, each group counts how many other groups of the same topic it points to via forwarded messages, then a stratified random sample is taken based on this metric to create a realistic case.

More specifically, for each *Videogame Modding* group, the number of other *Videogame Modding* groups it points to is computed, and the quantiles of this distribution of groups based on their links are calculated. A fixed fraction is then sampled from each quantile. Specifically, there are 10 quantiles; we randomly sample 10% of the groups from each quantile, yielding 10% of the initial set of groups labeled as *Videogame Modding*. In the mixed case, these seeds are also mixed with groups labeled with other topics, which account for 50% of the total. In the end, 50% of the seeds are tagged with *Videogame Modding*, and another 50% with other labels. In the pure case, groups tagged with other topics are not added. The pure case is useful for understanding how well the crawler remains on the specified topic when starting from groups

labeled with that topic, to assess whether the choice of good seeds is relevant. The parent-child relationship is always established by forwarded messages from one channel to another.

Two different experiments are conducted. In the first type, which we will call pure experiments, all seed groups are assigned to the *Videogame modding* tag. In the second type, which we call Mixed, the groups tagged *Videogame modding* are mixed with groups assigned to other labels to resemble the real case in which we start from seeds obtained via a query-based search for specific keywords, as in the US Dataset. The mixed experiments provide a realistic view of a potential commercial use of the crawler, in which not all initial groups necessarily address the topic of interest.

LDA is used for topic detection, and GPT-5-Nano is used for classification. For each topic, GPT-5-Nano acts on the 20 most significant keywords, the 3 most significant messages, and 3 random messages, as it is cost-effective in terms of tokens, but effective in classification.

We applied lower thresholds because, compared to the US dataset, group topics are much more varied. The impact of these lower thresholds is evaluated in Section 4.4.2.

We deliberately chose not to search specifically for *Videogame Modding* but rather for *Gaming* groups in general, to assess whether the crawler can find relevant groups even without providing the exact label definition, which may be unknown before data collection.

The prompt used to classify topics is the following:

GAMING topics include:
- Videogame modding
- Video games (any platform: PC, console, mobile)
- Game mods, cheats, hacks, aimbots
- Esports, gaming tournaments
- Specific games (Minecraft, Fortnite, PUBG, COD, GTA, etc.)
- Gaming communities, clans, guilds
- Game streaming (Twitch, YouTube Gaming)

NOT GAMING:
- General entertainment (movies, TV, music): NO
- General software/tech: NO
- Crypto/trading/NFT: NO
- Adult content: NO
- News, politics: NO
- Social media, memes (unless specifically gaming memes): NO

- Real warfare, military news, geopolitical conflicts: NO
- Airsoft, paintball, real weapons: NO
- Gambling, betting, casinos (unless video game gambling): NO
- Sports betting, fantasy sports: NO
- Card games (poker, blackjack - unless digital/video game): NO
- Board games (unless digital versions): NO
- Scams, money doubling, carding, fraud: NO

KEYWORDS: (the top 20 most significant keywords associated with the topic)

TOP 3 REPRESENTATIVE MESSAGES: (the top 3 most significant messages associated with the topic)

3 RANDOM MESSAGES: (3 random messages associated with the topic)

The prompt is intended to be generic to assess whether it can classify channels of topics within the main topic, such as *Videogame Modding* within the general *Gaming* topic.

The following definitions of TP, FP, and FN were used to calculate precision and recall.

- TP: Channels predicted as *Gaming* by the crawler and simultaneously labeled as *Videogame Modding* by the TGDataset authors are included.
- FP: Channels predicted as *Gaming* by the crawler but labeled other than *Videogame Modding* in the dataset are included.
- FN: Channels labeled as *Videogame Modding* in the TGDataset, minus the groups classified as *Gaming* by the crawler, are included.

Chapter 4

Results

This chapter presents the empirical findings obtained by applying the proposed crawling methodology to both the US Dataset and the TGDataset. The discussion begins by characterizing the structural and temporal properties of the US Dataset, establishing the baseline network topology and linguistic distribution of the collected Telegram messages. Following this contextual overview, the focus shifts to validating the core components of the crawler. This includes a quantitative comparison between Latent Dirichlet Allocation (LDA) and BERTopic to justify the chosen topic modeling approach, and an evaluation of the threshold-based exploration strategy and the LLM's classification accuracy.

Building on these validated mechanisms, the remainder of the chapter details the performance of large-scale crawling across both datasets. For the US Dataset, the analysis assesses how varying thresholds impact overall precision and recall, ultimately demonstrating the system's capacity to detect specific political events over time. The evaluation is then extended to the highly heterogeneous TGDataset, which spans a wide range of topics. Here, a preliminary topological analysis of the Telegram network is used to contextualize the subsequent crawling results, which assess the crawler's robustness and accuracy in a noisy environment across different initial seed configurations.

4.1 US Dataset Characterization

Before evaluating the crawler's performance, we first characterize the US Dataset to understand the environment in which the crawler operates. This includes the forwarding network, the language distribution across channels,

and the temporal concentration of messages, all of which directly influence the crawler's behavior and provide context for the results that follow.

4.1.1 US Dataset Hierarchical Network Structure

To gain a clearer understanding of the topological structure of the graph connecting Telegram channels originating from the seed channels, we conducted a small experiment on the US Dataset to determine the maximum depth of the network and quantify the volume of data at each depth. We evaluated the computational load of a crawler that visits all children without filtering, to validate the necessity of an optimized threshold-based crawling strategy. The following results are summarized in Table 4.1.

Table 4.1: Distribution of unique channels and messages across hierarchical levels.

Level	Unique Channels	Total Messages
0 (Seeds)	156	570,427
1	4,449	62,689,935
2	21,852	350,084,913
3	3,348	72,592,311
4	1	746

As Table 4.1 illustrates, the network exhibits a massive branching factor that culminates in level 2. This structural pattern underscores the need for an intelligent crawler to filter out unpromising groups. Note that Table 4.1 applies only a language filter, retaining all channels with at least one English message regardless of message length or duplicates.

4.1.2 US Dataset Linguistic Distribution

Figure 4.1 illustrates the prevalent language distribution among the seed channels; it is calculated considering the language with the most messages for each channel.

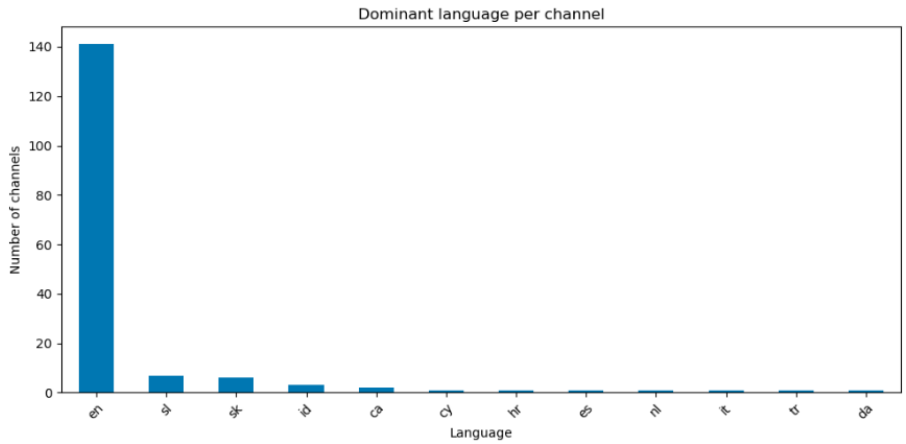

Figure 4.1: Language distribution among channels.

Figure 4.1 demonstrates that the vast majority of the groups are in English. Therefore, to effectively mitigate the challenge of cross-language topic detection, only English messages are retained.

4.1.3 Messages Temporal Distribution

Figure 4.2 presents the time distribution of messages in the seed channels.

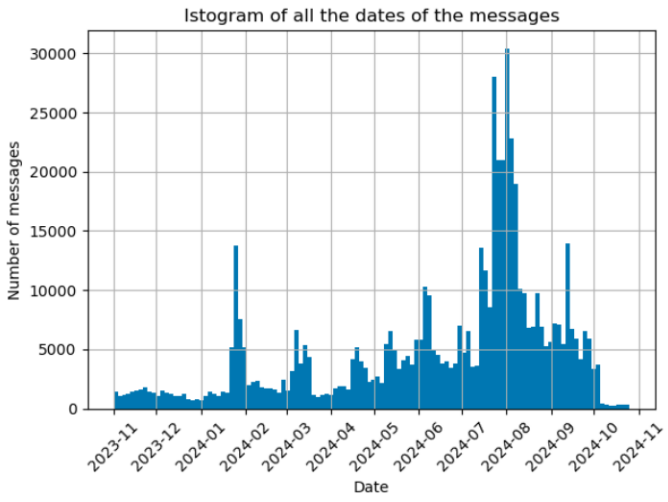

Figure 4.2: Histogram of message dates.

There is a significant surge in the second half of July and the first half of August. The specific topics driving this increase will be analyzed in subsequent sections.

To verify that the dataset captures actively posting channels rather than abandoned ones, Figure 4.3 depicts the date of the last message from each channel.

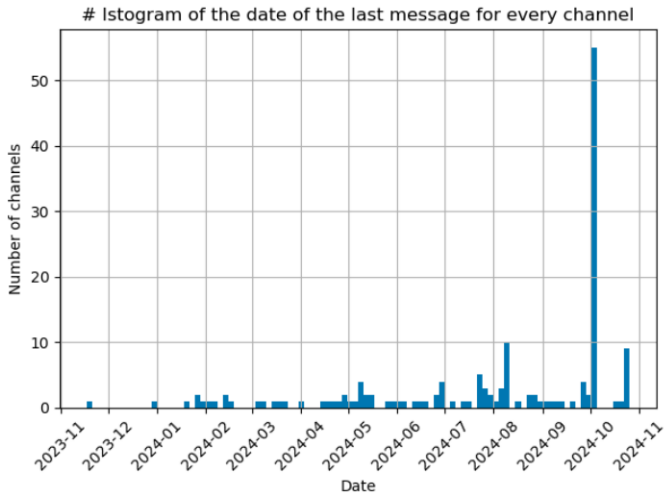

Figure 4.3: Histogram of last message dates.

As shown in Figure 4.3, a pronounced peak appears in October 2024, which corresponds to the date when the authors of the US Dataset stopped their data collectionas stated in Blas et al. [2025]. This confirms that the vast majority of channels were still actively posting at the time of collection.

4.2 Design Validation Results

With the dataset characterized, this section reports the results of the experiments that validate the core design choices of the crawler: the effectiveness of the threshold-based strategy using BERTopic as baseline, the validity of LDA as a lighter replacement, and the reliability of the LLM as an automatic classifier.

4.2.1 Optimal Strategy Validation US Dataset

This section presents the results related to the optimal threshold of the ratio of political messages in the parents of the children chosen for the continuation of the crawling process. The following results validate whether the adopted strategy offers a significant advantage over a crawler that visits all children without filtering, and identify the optimal threshold.

For this validation, BERTopic was applied to Levels 0 and 1, as these two levels are sufficient to establish a parent-child relationship. While the analyzed time interval remains identical to that of Table 4.1, corresponding to the whole dataset, the number of evaluated channels is drastically reduced. This is because, in addition to the language filter, the full preprocessing pipeline described in Section 3.2 was applied to retain only the highest-quality channels required by BERTopic. Furthermore, because BERTopic assigns a significant fraction of messages as outliers, many channels have too few classified messages to compute a reliable political ratio, thereby excluding any parent-child connection in which either node falls below the minimum number of messages. The analysis yields 165 parents and 194 unique children.

The following two Figures, 4.4 and 4.5, compare the random selection of parents to a weighted selection in which parents with a higher ratio of political messages have a higher probability of being chosen in order to confirm the necessity of the subsequent analyses. In both figures, the x-axis represents the percentage of parents selected, with parents ordered by their political message ratio in descending order; for instance, at 10%, only the parents with the highest political message ratio are selected, whereas at 100%, all parents are included. In the random selection case, parents are randomly chosen with uniform probability, while in the weighted selection, each parent i is sampled with probability $p_i = r_i / \sum_{j=1}^{N} r_j$, where r_i is the political message ratio of parent i and the denominator is the sum of the political message ratios across all N candidate parents, acting as a normalization factor. This formulation ensures that parents with higher concentrations of political content are more likely to be selected, while still giving every parent a non-zero chance of being chosen, thereby reflecting a realistic scenario in which initial seed selection is informed but not deterministic. The probabilities p_i are computed once before the trials and remain fixed. For each of the 800 trials and for each percentage on the x-axis, a new independent sample of k parents is drawn without replacement using these fixed probabilities, where k corresponds to the given percentage of the total number of parents. This means that, at 10%, a sample of k parents is drawn, at 20%, a separate sample of $2k$ parents is drawn from scratch, and so on; each trial and percentage is independent. In both figures, the boxplots are generated from 800 trials. For each trial, the y-axis represents a ratio where the denominator is the total number of political messages from the children of all parents, and the numerator is the number of political messages from the children of the chosen parents.

As Figure 4.4 and Figure 4.5 show, the weighted approach consistently outperforms the random selection at every percentage. The median coverage

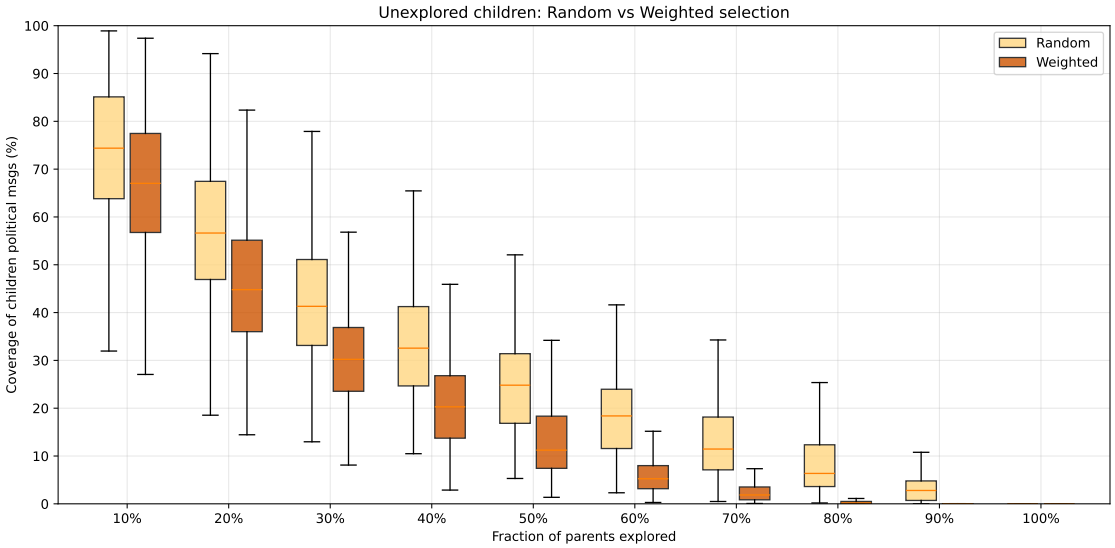

Figure 4.4: Comparison of unexplored children: random vs. weighted parent selection.

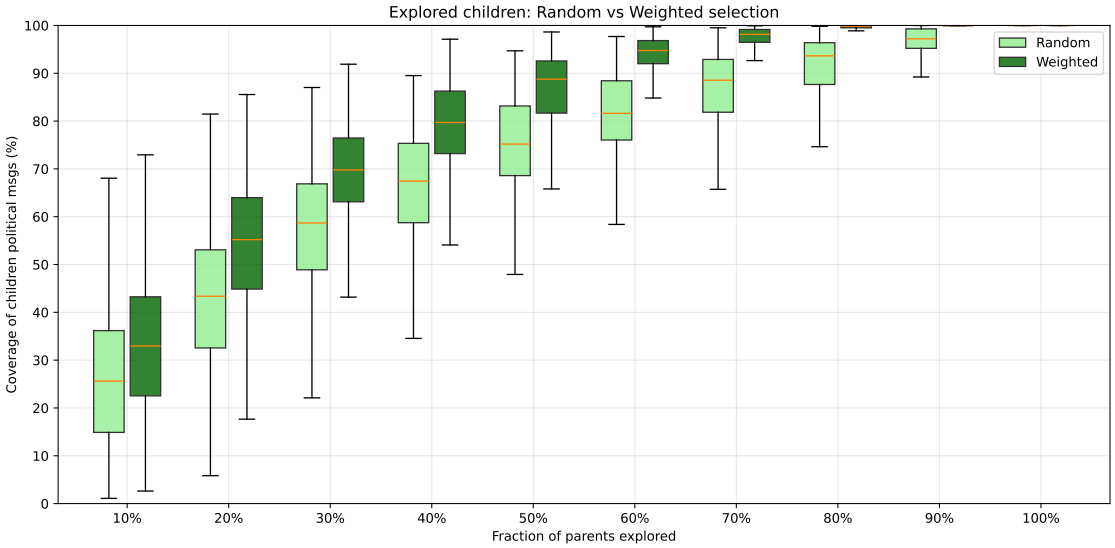

Figure 4.5: Comparison of explored children: random vs. weighted parent selection.

of explored children is higher, whereas that of unexplored children is lower, confirming that prioritizing parents with a higher political ratio yields better coverage.

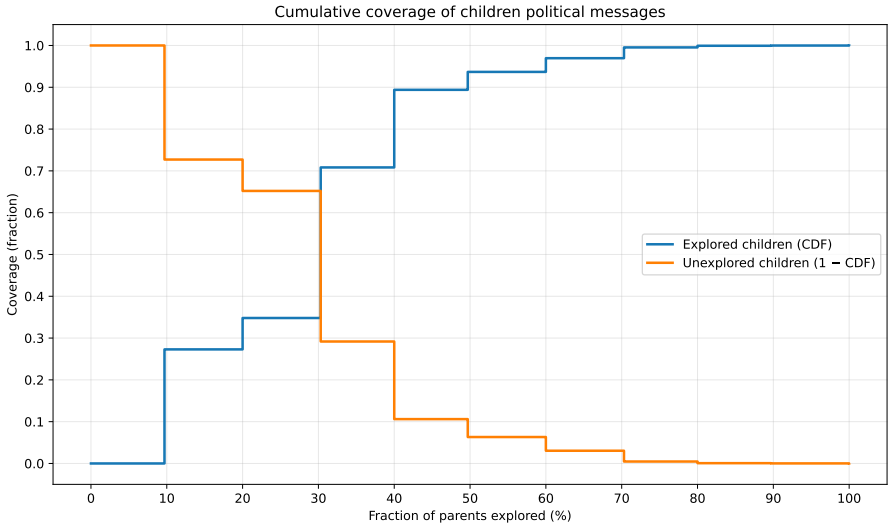

Figure 4.6: Cumulative children's political message coverage

Figure 4.6 is used to quantitatively understand the correct number of parents to obtain a sufficiently representative number of political messages in the children of the selected parents. The x-axis in Figure 4.6 represents the fraction of parents sorted in descending order by their percentage of political messages. The y-axis represents the cumulative percentage of political messages covered by the children of the parents selected along the x-axis, relative to the total number of political messages across all children. Thus, children of parents with the highest percentage of political messages are already included within the 0 to 10 x-axis range, and so on for lower concentrations of political messages.

As Figure 4.6 illustrates, after taking the best 50% of the best parents, the percentage of additional political messages covered is marginal.

Figure 4.7 is used to have a quantitative measurement of the number of children channels discarded in case only the best 50% of parent channels are used. The x-axis in Figure 4.7 represents, as before, the best percentage of parents based on the ratio of political messages in decreasing order based on the threshold. On the other hand, the y-axis shows the number of children explored starting from those parents, whereas the unexplored bars represent children that still have to be explored.

Figure 4.7 shows how keeping the best 50 percent of parent channels results in discarding 59 out of 194 children's channels, which corresponds to about 30% less overhead.

To complete the analysis, we need to determine the minimum percentage

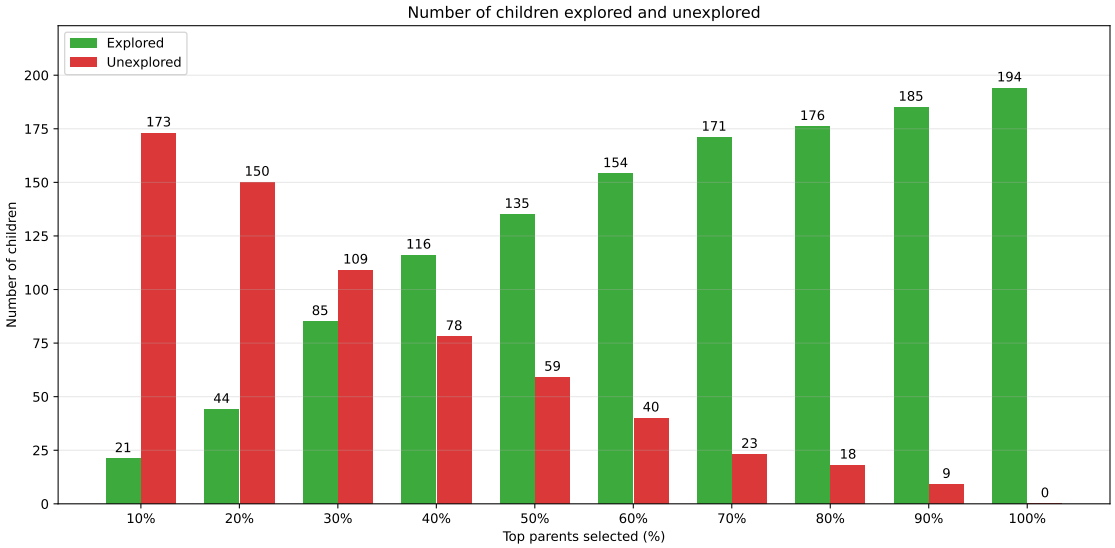

Figure 4.7: Children explored

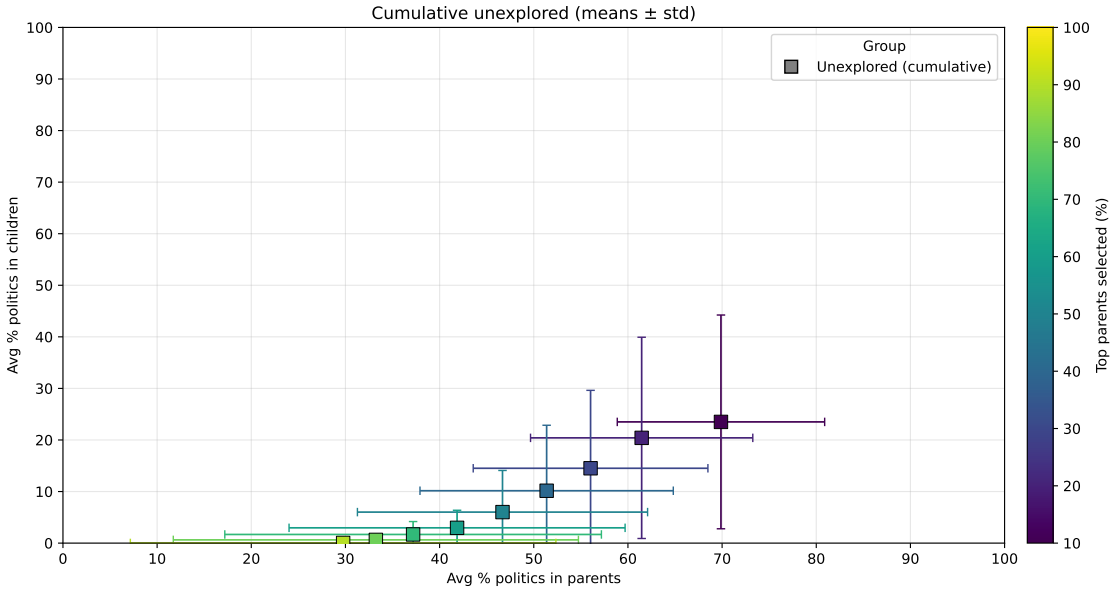

Figure 4.8: 2D scatter plot

of political messages among parents that the crawlers must collect, keeping in mind that the best percentage of top parents to collect is 50%. The crawler will analyze children, considering only those whose parents exceed the threshold. This allows us to maintain a good balance between the collected political messages and computational efficiency.

Figure 4.8 presents a 2D scatterplot. The x-axis shows the average percentage of political messages among parents, while the y-axis represents the average percentage of political messages among their children. The colors of the dots indicate the selected portion of parents, divided into 10% bands, ordered from most to least promising: starting with the most promising channels in dark purple (the best 10%), and progressively descending to the least relevant ones in yellow, reaching 100%. The square unexplored dots follow a cumulative logic and show the average values of children not referred to that specific percentage of parents. The first dark square, for example, indicates the average political percentage of all child channels, which would be ignored if the search were limited to the top 10% of parents. Finally, the cross-shaped error bars indicate the standard deviation, illustrating the dispersion of the data around the mean values for both parents along the X-axis and children along the Y-axis. The figure shows that setting a minimum political message percentage of 40% results in discarding parent groups whose average political message percentage is below 40%, thereby retaining the top 50% of parent political groups.

4.2.2 BERTopic and LDA topic analysis

As detailed in the methodological framework in Section 3.4.1 and Section 3.4.2, the optimal topic modeling configuration was determined by evaluating Coherence, Diversity, and Silhouette scores.

Figure 4.9 illustrates the top words per topic produced by BERTopic with paraphrase-MiniLM-L6 as sentence transformer, `min_cluster_size` of 30 for HDBSCAN, and `n_components`, `n_neighbors`, `min_dist` respectively of 10, 10, and 0.0 for UMAP. This configuration was selected because it achieved the best diversity score among all tested combinations.

Figure 4.10 was obtained by LDA topic detection optimized for the best Coherence (C_v) score as detailed in Section 3.3.1, with the following hyperparameters: `n_topics` and `learning_decay` set to 33 and 0.7, respectively. For both models, in addition to selecting the configurations with the highest metric values, the final choices were validated through a manual qualitative analysis performed by two researchers.

We observe that moving from a transformer-based tool such as BERTopic to a probabilistic one such as LDA slightly reduces word coherence. Because of this decrease, a further quantitative analysis is presented in the next section.

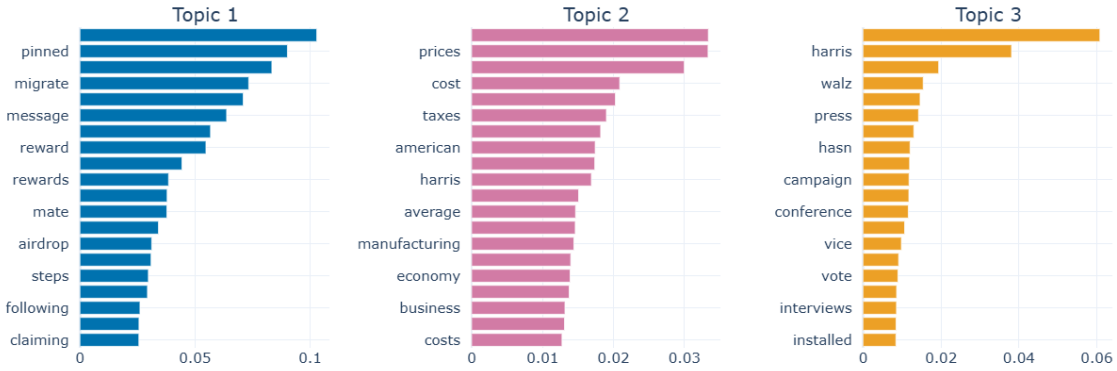

Figure 4.9: Top words per topic for BERTopic

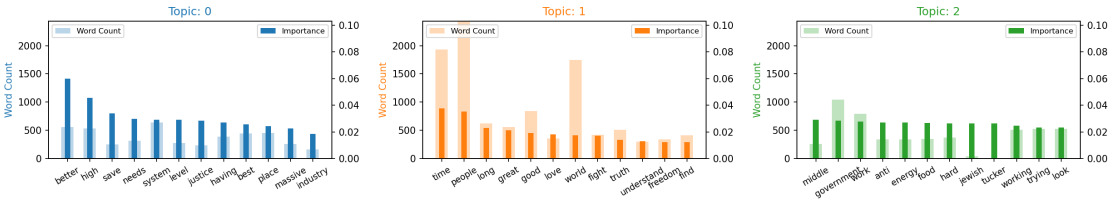

Figure 4.10: Top words and word count per topic for LDA

4.2.3 Validating LDA against the BERTopic Baseline

As described in Section 3.4.2, we evaluated whether LDA can replace BERTopic by comparing the topic assignments produced by their respective optimal configurations detailed in Section 4.2.2 on the same set of documents using a contingency table, the Adjusted Mutual Information (AMI), and the Adjusted Rand Index (ARI).

The evaluation yields an AMI of 0.28 and an ARI of 0.12. Although these numerical results are modest, they should be interpreted qualitatively. Due to its large dimensions, the full contingency table is omitted from this document. However, an analysis of the table reveals that documents assigned to a single LDA topic are predominantly partitioned into a small, consistent subset of BERTopic topics, whereas the document counts in other BERTopic topics are negligible. This means that each BERTopic topic consistently maps to the same LDA topic, rather than being scattered randomly across the entire LDA partition. The low AMI and ARI values are therefore not a sign of disagreement between the two methods but rather a consequence of their different granularity: BERTopic generates more fine-grained topics, effectively breaking down the broader themes identified by LDA into highly

specific sub-topics, and pairwise metrics penalize this many-to-one mapping even when the underlying semantic alignment is consistent.

4.2.4 LLM-as-a-judge results

This section evaluates how the LLM's classification differs from the manual topic classification as political or non-political performed by two researchers. The manual classification serves as the ground truth for this analysis. GPT-5-Nano was selected because it offers the lowest token cost and can correctly classify topics as political or non-political.

Table 4.2: Example of topic representation for LLM classification (Level 0)

Topic 0 keywords: states, united, iran, gets, sent, israel, nuclear, missiles, personal, details, response, iranian, phil, attacks, missile, germany, email, totally, drones, taylor
Top documents: 1. "united states officials believe 400-500 drones and missiles will be launched at israel. the united states predicts that about 85% of drones and missiles will be intercepted..." 2. "more footage of the iranian missiles impacting israel, presumably the ramon airbase" 3. "the united states of america is going to declare war on [redacted]" Random documents: 1. "watch: cnn has a meltdown over our coverage of the taylor swift psyop" 2. "united by history, heritage, and hope there's not a thing the united states and ireland can't do together. potus"

Table 4.2 illustrates the data format used to prompt the model for a topic in Level 0. This format comprises the 20 most significant keywords, the three most relevant messages, and three randomly selected messages.

The results of the comparison between the human and LLM topic classification are summarized in Table 4.3.

Of the 58 topics, human annotators classified 46 as political and 12 as non-political, whereas the LLM classified 46 as political. Using the metrics defined in Section 3.4.3, the recall is 95.65%, demonstrating that the LLM successfully identifies almost all the topics that human annotators labeled as political. The precision is 95.65%, indicating that the LLM reliably avoids misclassifying non-political topics as political. The F1-score is 95.65%, which

Table 4.3: Confusion matrix: LLM classification vs human ground truth (58 topics).

	Human: Political	**Human: Non-Political**
LLM: Political	44 (TP)	2 (FP)
LLM: Non-Political	2 (FN)	10 (TN)

indicates the overall quality of the classification. These results quantitatively validate the LLM approach for classification.

4.3 US Dataset Crawling Results

Having validated the individual components, this section presents the end-to-end crawling performance on the US Dataset. Compared to Table 4.1, which only filters by language, the automated pipeline applies the full preprocessing described in Section 3.2. This results in a visible reduction in seed channels across all experiments, confirming that the preprocessing phase effectively discards a significant number of low-quality channels and messages, thereby reducing the computational effort of the subsequent pipeline steps.

4.3.1 Comparison of the thresholds for the US dataset

In Table 4.4, an overview of different threshold experiments is shown.

Table 4.4: Threshold comparison: US dataset from July 1st to August 31st, 2024

Metric	**20%**	**40%**	**60%**	**80%**
Number of levels	5	4	5	8
Total time (hours)	8.42	6.06	5.36	5.36
Total channels explored	13,699	11,651	10,392	9,115
Political channels explored	7,371	6,943	4,553	2,435
Non-political channels explored	6,328	4,708	5,839	6,680
% Political channels	53.8%	59.6%	43.8%	26.7%
Total messages	4,057,296	3,306,133	2,970,392	2,735,058
Political messages	1,286,756	1,510,056	1,526,168	1,428,932
% Political messages	31.7%	45.7%	51.4%	52.2%

The number of levels indicates how many iterations the crawler performed before stopping, where each level corresponds to a complete round of the pipeline applied to a new set of child channels. The total number of channels explored and the channel political fraction indicate the model's effectiveness in filtering political channels from non-political ones. Also, the execution time for each threshold experiment is reported to estimate the overall completion time across the threshold experiments.

20% Threshold US Dataset

As illustrated in Table 4.5, the percentage of political messages is 100% in the seed channels and decreases at the second level, which has the highest number of political groups but a lower percentage of political messages than level 1. This result is expected, as level 1 is generated from level 0, which has groups with a higher percentage of political messages, according to the US Dataset creation procedure proposed in Blas et al. [2024].

Notably, this approach yields a comparable number of political channels to the 40% threshold experiment as shown in Table 4.4, but has a significantly higher absolute number of political messages.

Table 4.5: Metrics per level from July 1st to August 31st 2024 (20% threshold)

Level	Channels	Pol. channels	Non-pol. channels	Mean pol. msgs. %
0	92	92	0	100.00
1	3,131	2,587	544	82.63
2	9,648	4,632	5,016	48.01
3	800	60	740	7.50
4	28	0	28	0.00
Total	**13,699**	**7,371**	**6,328**	**53.81**

40% Threshold Analysis US Dataset

For the 40% threshold experiment as seen in Table 4.6, the total number of channels explored has decreased by 2,048 channels from the 20% experiment in Table 4.6, and approximately 428 channels, which were considered political in the 20% threshold experiment, are no longer considered political.

Therefore, if the crawler's aim is not to obtain channels that cover only a small portion of the discussion topics, but rather to obtain a sufficient

number of channels on the specified topic, the 40% threshold experiment is a good compromise.

In summary, the 40% threshold strikes an optimal balance between performance and accuracy, avoiding the severe loss of relevant channels observed at the 60% threshold in Table 4.7.

Table 4.6: Metrics per level from July 1st to August 31st 2024 (40% threshold)

Level	Channels	Pol. channels	Non-pol. channels	Mean pol. msgs. %
0	92	91	1	98.91
1	3,131	2,271	860	72.53
2	7,083	4,581	2,502	64.68
3	1,345	0	1,345	0.00
Total	**11,651**	**6,943**	**4,708**	**59.59**

60% Threshold Analysis US Dataset

The 60% threshold experiment of Table 4.7 exhibits a significant decrease in the number of political channels from the 40% threshold represented in Table 4.6; the number goes from 6,943 in the 40% experiment to 4,553 in the 60% experiment, which means there are many political groups missing.

This threshold should be used to have a general idea of the political topics discussed in the dataset, similarly to the 80% experiment summarized in Table 4.8.

Table 4.7: Metrics per level from July 1st to August 31st 2024 (60% threshold)

Level	Channels	Pol. channels	Non-pol. channels	Mean pol. msgs. %
0	92	81	11	88.04
1	3,056	1,300	1,756	42.54
2	6,147	3,065	3,082	49.86
3	1,045	107	938	10.24
4	52	0	52	0.00
Total	**10,392**	**4,553**	**5,839**	**43.81**

80% Threshold Analysis US Dataset

Table 4.8 illustrates how the 80% version of the experiment is ideal for providing a general overview of the discussion topics in the shortest possible computation time.

Table 4.8: Metrics per level from July 1st to August 31st 2024 (80% threshold)

Level	Channels	Pol. channels	Non-pol. channels	Mean pol. msgs. %
0	92	84	8	91.30
1	3,130	467	2,663	14.92
2	2,018	366	1,652	18.14
3	677	554	123	81.83
4	2,233	758	1,475	33.95
5	850	200	650	23.53
6	112	6	106	5.36
7	3	0	3	0.00
Total	**9,115**	**2,435**	**6,680**	**26.72**

Threshold Experiment Conclusions

Table 4.9: Final summary for threshold

threshold	Channels coverage	Channels precision	Levels	Summary
20%	excellent	low	4	Too permissive
40%	high	good	4	Suggested
60%	average	good	5	valid alternative
80%	high	excellent	8	Too restrictive

In summary, the 20% threshold is too permissive; although the probability of missing a political channel is nearly zero, it is prone to including unrelated content.

The 40% threshold offers a better balance between coverage and precision. Conversely, the 60% threshold may exclude several political channels and should be used only to provide a general overview of the discussion topics.

Finally, the 80% threshold provides only a very approximate view because, due to the exclusion of many channels, their corresponding topics are also lost.

4.3.2 Temporal Comparison Experiment

Table 4.10 summarizes two time windows experiments with a fixed threshold at 40%: the first between July 15th and August 15th, 2024, and the second from March 1st to March 31st, 2024.

Table 4.10: Time Comparison (40% threshold)

Metric	**July-August**	**March**
Levels explored	5	5
Total channels	10,660	9,105
Political channels	5,869	6,516
% Political channels	55.1%	71.6%
Total messages	2,080,844	1,516,166
Political messages	824,783	731,170
% Political messages	39.6%	48.2%

Table 4.10 highlights that 71.6% of channels were classified as political in March, compared to 55.1% during the summer. This noticeable difference stems from the intense political fervor within these groups at the time. With users highly focused on current events, a larger number of chats naturally crossed the required threshold to be flagged as political. The overall increase in the percentage of political messages strongly supports this. In short, tracking these percentage spikes offers a clear and practical way to identify moments of intense political debate.

Table 4.11: Political events detected during the period from July to August 2024

Event	**Keywords**
attempted assassination of Donald Trump 13 lug	assassination, secret, service, rally
VP J.D. Vance choice 15 lug	vance, trump, president
VP Tim Walz choice 6 ago	walz, people, know
Haniyeh death 31 lug	haniyeh, ismail, hamas, leader
Riots in UK end of July	starmer, british, riots, police
Endorsement Elon Musk	elon, musk, trump, donald

From Tab. 4.12 and 4.11, we see that the topics identified by the crawler

Table 4.12: Political events detected in the period of March 2024

Event	Keywords
State of the Union (7 mar)	state, union, address, biden
Super Tuesday (5 mar)	super, tuesday, haley, republican
The terrorist attack in Moscow (22 mar)	crocus, moscow, terrorist, isis
Immigration crisis USA	border, migrants, illegal, immigrants
Francia and Russia tensions	macron, french, france, scholz
Georgia court case	georgia, willis, judge, case
Nikki Haley primaries	haley, nikki, presidential

across different time windows are distinct yet closely related to events that occurred in those periods. Therefore, to understand the most important topics and discussion groups of a given period, it suffices to use the messages from that period, and the crawler will automatically assign greater importance to these topics than to those referring to more general periods.

4.3.3 Dataset-Wide Crawling at the 40% Threshold

In this section, the results for the best 40% threshold compromise over the entire time window are explored. The number of political channels and messages, and the percentage of political messages in the total, for each level are shown in Table 4.13 to indicate the method's quality.

As illustrated in Table 4.13, the proportion of political channels drops significantly after Level 2, given that the broader US Dataset contains a substantial amount of non-political noise beyond the immediate seed network.

Table 4.14 details the differences between the channels excluded by the crawler and the ones successfully explored.

In Table 4.14, it is clear that the majority of the channels were not visited because their messages were not collected by the dataset authors or they did not contain English messages.

To rigorously evaluate the effectiveness of the filtering mechanism, the LDA topic detection pipeline was applied to the 4,203 valid channels bypassed during the crawling process. Table 4.15 presents the results of this analysis. It is evident that only a very small fraction of unvisited channels were political, and the vast majority of discarded messages were non-political, thereby confirming the validity of the crawling strategy.

Table 4.13: Metrics per 40% threshold

Level	Channels	Pol. Channels	Msgs.	Pol. Msgs.	Topics	Pol. Topics
0	137	129	110,229	96,595	94	90
1	3,653	2,881	6,221,081	3,259,425	393	258
2	9,673	7,174	9,425,135	5,852,784	300	209
3	2,680	300	2,141,923	359,458	300	36
4	97	2	14,157	912	23	1
Total	**16,240**	**10,486**	**17,912,525**	**9,569,174**	–	–

Table 4.14: Coverage of the strategy

Category	Channels
Total channels in dataset	127,140
Channels visited	23,889
Non visited channels	103,251
Referred to non visited channels:	
Without folders in dataset	97,243
Without messages files	232
Without English messages	825
With English valid messages	4,203

Table 4.15: Comparison: Visited channels and non-visited channels

Metric	Visited	Non visited
Analyzed channels	16,240	4,203
Political channels	10,486	22
Non political channels	5,754	4,181
Total messages	17,912,525	4,755,501
Political messages	9,569,174	50,177
Total topics	1,110	300
political topics	594	6

The keywords of the topics referred to the excluded channels, reported in Table 4.16, further confirming their non-political nature.

In summary, the strategy applied to all the dataset messages with a threshold

Table 4.16: Most relevant topics in non-visited channels, ordered by number of messages.

Topic	Classification	Messages	Top Keywords
276	Non-political	419,226	balance, wallet, transactions, owner, info
47	Non-political	265,391	birdeye, useful, solscan, liquidity, time
8	Non-political	223,145	owns, market, supply, wild, like

of 40% is highly efficient, capturing 99.5% of the political channels and excluding 99.5% of unreached non-politically related channels. These represent isolated political channels that are not connected to the seed network. Unvisited channels are dominated by cryptocurrency and trading-related content, confirming that they do not contain relevant political information.

4.4 TGDataset Results

This section presents the results of the graph analysis and the crawling process performed on the TGDataset.

4.4.1 TGDataset Graph Analysis Results

As detailed in Section 3.5.2, the following analysis is performed on the unweighted directed graph formed by the forwarding relationships. Table 4.17 summarizes the main topological metrics of this graph.

The density corresponds to approximately 2.1 million edges, out of a theoretical total of 2.36 billion, for a complete graph. Nodes have an average in-degree and out-degree of 43.2, with outliers having more than 100 connections. The largest WCC contains 48,222 nodes, corresponding to approximately 99.2% of the nodes, meaning the network is almost entirely a single connected component. The largest SCC contains 36,386 nodes, corresponding to approximately 74.8% of the nodes, indicating that a significant fraction of groups are mutually reachable. The other three components, in order of size, have 17, 15, and 12 nodes, so there is no need to include the nodes of all major SCCs in the initial seeds to ensure coverage of all channels, as nodes from the second-largest SCC onward are negligible. Reciprocity is

Table 4.17: Topological metrics of the TGDataset forwarding graph.

Metric	**Value**
Edges	2,099,813
Nodes	48,630
Density	8.88×10^{-4}
Average in/out-degree	43.2
Number of WCCs	384
Largest WCC	48,222 nodes (99.2%)
Number of SCCs	11,622
Largest SCC	36,386 nodes (74.8%)
Reciprocity	21.6%
DAG	No

21.6%, which means that only 21.6% of the relationships are bidirectional. This means that the number of groups that forward to each other is very low compared to the total, indicating that most channels are unaware of each other. The graph is not a DAG, so cycles exist as expected in a social network.

Figure 4.11 shows the PageRank score distribution among nodes.

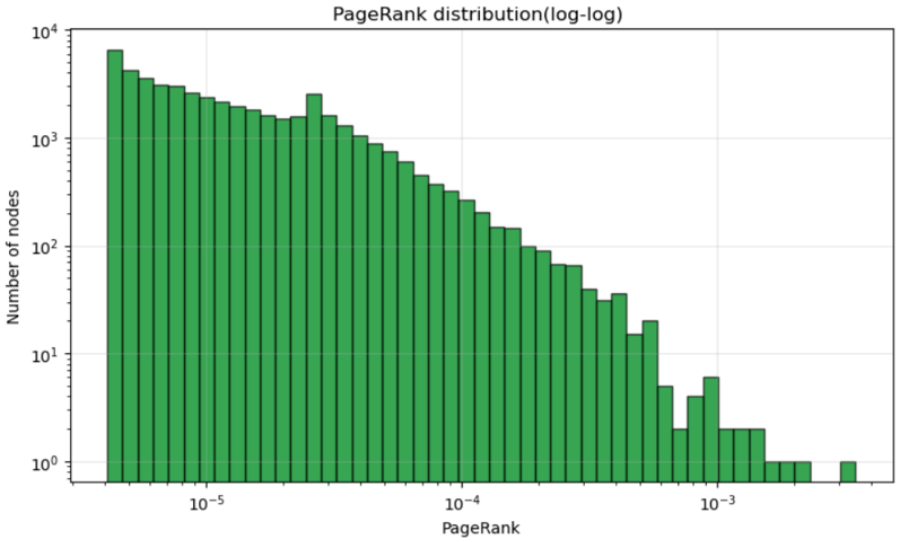

Figure 4.11: PageRank value histogram (log-log scale).

The PageRank score graph exhibits a power-law distribution, indicating

the presence of preferential attachment, in which a new channel tends to forward messages to channels that other groups have previously forwarded. This demonstrates that individual channels can significantly change the network, even at a global level. Therefore, it is important for a crawler to cover all relevant channels, as missing any would result in the loss of a large portion of the informative content.

The group of top hubs is distinct from that of top authorities, meaning that, in general, the groups tend to have a single role: some agglomerate information through message forwards, while others create the information that the former draw from. This separation implies a largely unidirectional information flow from authorities to hubs, which benefits the crawler as it can discover new groups at each level of the traversal. Note that the crawler is immune to cycles by design, since it maintains a set of all previously visited channels and excludes them from subsequent levels.

4.4.2 TGDataset Crawling Results

As described in Section 3.5.2, two seed configurations are tested: Pure, where all seeds are labeled as *Videogame Modding*, and Mixed, where 50% of the seeds are labeled with other topics. Table 4.18 and Table 4.19 report the global results across all thresholds.

Table 4.18: Global classification results for Pure seeds (stratified sampling).

Threshold	Precision	Recall	F1 Score
10%	50.1%	76.3%	60.5%
15%	68.8%	72.7%	70.7%
20%	74.2%	70.5%	72.3%
30%	85.3%	60.7%	70.9%
40%	84.7%	62.3%	71.8%

Naturally, as the threshold increases, the number of false positives decreases because the crawler becomes more selective, selecting only groups with a high percentage of related messages.

The number of false negatives increases with the threshold, as the threshold required by the crawler to collect them increases, leaving some groups labeled as *Videogame Modding* with a lower percentage of messages.

Most false negatives are due to the classification method, which imposes a sufficiently high proportion of messages on the selected topic, whereas the

Table 4.19: Global classification results for Mixed seed types (stratified sampling).

Threshold	Precision	Recall	F1 Score
10%	52.9%	73.5%	61.5%
15%	51.0%	75.4%	60.8%
20%	62.5%	72.8%	67.3%
30%	77.6%	61.0%	68.3%
40%	81.3%	58.3%	67.9%

dataset authors also labeled groups with few modding-related messages and much noise as *Videogame Modding*.

For each experiment, the numbers of false positives and false negatives remain similar, underscoring that the crawler can maintain a similar level of quality even when starting from groups covering topics other than the topic of interest.

The majority of false positives are due to the fact that a more general *Gaming* topic is set up, rather than *Videogame Modding*, to determine whether the crawler can also pick up all the subtopics of the initial topic, introducing channels that discuss *Gaming* but not specifically *Videogame Modding*.

Others, however, are false positives because, although they contain messages about *Videogame Modding*, the authors still labeled them as *Carding*, as this topic is very similar and often overlaps with *Videogame Modding* in the broader *Gaming* context.

Since topics such as *Videogame Modding* often involve a wealth of terminology, slang, and extremely short messages, they can be difficult for a deterministic algorithm such as LDA to interpret.

If it is not known a priori whether the seed sample is pure or mixed, a threshold between 20% and 30% can be chosen, as they contain the best F1-score for both the pure and mixed cases.

On the other hand, if the crawler operates on a pre-defined pure seed sample, it may be appropriate to use the 20% threshold, as it yields the highest F1 score. The same reasoning applies to the 30% threshold in the case of mixed seeds.

Chapter 5

Conclusion

In this chapter, the main achievements of this thesis are summarized, the main limitations, the practical applications, and the possible future work are discussed.

5.1 Summary of findings

This work successfully addressed the research questions formulated in Section 1.2, demonstrating the viability of an optimized Telegram crawler.

RQ1 & RQ2: Regarding the US Dataset, it is visible how the 40% threshold is optimal to correctly filter out unrelated channels while collecting the argument-inherent ones, saving significant computational resources. Furthermore, the crawler can be configured with different accuracy levels: for general groups susceptible to spam, it is essential to lower the threshold to collect groups effectively, while a higher threshold is more than sufficient if you want a general idea of the topics covered.

RQ3: The integration of an LLM-as-a-judge demonstrated that the crawler can reliably understand a prompt and identify related channels.

RQ4: The crawler successfully identifies the central topic of a group at runtime by employing Latent Dirichlet Allocation (LDA) optimized via the Tree-Structured Parzen Estimator (TPE). This probabilistic approach proved computationally lightweight and effective for real-time classification of short text messages, without the overhead of transformer-based models.

5.2 Limitations

The LDA algorithm has limitations in creating coherent arguments with short messages or when groups use a lot of slang.

The developed crawler is limited to Telegram and does not explore other social media or messaging platforms.

Currently, the crawler only filters English messages, potentially excluding important groups.

The developed crawler cannot fetch messages in real time and is limited to crawling from a dataset.

5.3 Potential Applications

Many malware and zero-day exploits are distributed and sold on social networks such as Telegram. Cyber Threat Intelligence groups can use the crawler to identify such groups and report them to the relevant authorities. The developed crawler can also be used as an OSINT tool to identify and search for potential malicious activity within the Telegram ecosystem. This can be achieved by properly configuring seed channels and using a well-designed prompt.

As demonstrated by the US Dataset experiment, the crawler is highly efficient at providing a general idea of the issues discussed during a political election. It can therefore be used by data scientists to identify groups that contain fake news and to study the influence of social media on the outcomes of political campaigns.

Companies can use the crawler to protect their intellectual property from illegal use. They adapt the prompt to their brand to find and report to authorities groups dedicated to software piracy, counterfeit goods, or illegal carding and fraud operations.

5.4 Future work

Future research could integrate my work with the crawler developed in Perlo et al. [2025] to automate the crawling process, starting from seeds collected using TGStat. This would enable the creation of a crawler that processes real-time messages, thereby expanding its potential.

Furthermore, the use of Dynamic Topic Models (DTM) would allow the crawler to continuously update topic distributions as new messages arrive

in every analyzed channel, eventually changing their evaluation over time. This would allow the crawler to adapt to ever-evolving conversations without having to restart the crawling process from scratch.

Furthermore, extending the crawler's capacity for multimodal analysis could significantly expand the search scope, given the increasing use of images and videos on social media.

Bibliography

Pablo Jost and Leyla Dogruel. Radical Mobilization in Times of Crisis: Use and Effects of Appeals and Populist Communication Features in Telegram Channels. *Social Media + Society*, 9(3):20563051231186372, July 2023. ISSN 2056-3051. doi: 10.1177/20563051231186372. URL https://doi.org/10.1177/20563051231186372.

Leonardo Blas, Luca Luceri, and Emilio Ferrara. Unearthing a Billion Telegram Posts about the 2024 U.S. Presidential Election: Development of a Public Dataset, October 2024. URL http://arxiv.org/abs/2410.23638. arXiv:2410.23638 [cs].

Massimo La Morgia, Alessandro Mei, and Alberto Maria Mongardini. TG-Dataset: Collecting and Exploring the Largest Telegram Channels Dataset. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pages 2325–2334, Toronto ON Canada, July 2025. ACM. ISBN 979-8-4007-1245-6. doi: 10.1145/3690624.3709397. URL https://dl.acm.org/doi/10.1145/3690624.3709397.

Massimo La Morgia, Alessandro Mei, Alberto Maria Mongardini, and Jie Wu. It's a Trap! Detection and Analysis of Fake Channels on Telegram. In *2023 IEEE International Conference on Web Services (ICWS)*, pages 97–104, July 2023. doi: 10.1109/ICWS60048.2023.00026. URL https://ieeexplore.ieee.org/abstract/document/10248275.

Elisabeth Steffen. More than Memes: A Multimodal Topic Modeling Approach to Conspiracy Theories on Telegram, March 2025. URL http://arxiv.org/abs/2410.08642. arXiv:2410.08642 [cs].

Ergon Cugler de Moraes Silva. TelegramScrap: A comprehensive tool for scraping Telegram data, December 2024. URL http://arxiv.org/abs/2412.16786. arXiv:2412.16786 [cs].

Alessandro Perlo, Giordano Paoletti, Nikhil Jha, Luca Vassio, Jussara Almeida, and Marco Mellia. Topic-wise Exploration of the Telegram Group-verse. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pages 1792–1801, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 979-8-4007-1331-6. doi: 10.1145/3701716.3717506. URL https://dl.acm.org/doi/10.1145/3701716.3717506.

Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. BTM: Topic Modeling over Short Texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941, December 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2014.2313872. URL http://ieeexplore.ieee.org/document/6778764/.

Luca Vassio, Tailai Song, and Gabriele Ciravegna. Machine Learning for Networking ML4N.

R. Devika, Subramaniyaswamy Vairavasundaram, C. Sakthi Jay Mahenthar, Vijayakumar Varadarajan, and Ketan Kotecha. A Deep Learning Model Based on BERT and Sentence Transformer for Semantic Keyphrase Extraction on Big Social Data. *IEEE Access*, 9:165252–165261, 2021. ISSN 2169-3536. doi: 10.1109/ACCESS.2021.3133651. URL https://ieeexplore.ieee.org/abstract/document/9641788.

Irene Benedetto, Moreno La Quatra, Luca Cagliero, Luca Vassio, and Martino Trevisan. TASP: Topic-based abstractive summarization of Facebook text posts. *Expert Systems with Applications*, 255:124567, December 2024. ISSN 09574174. doi: 10.1016/j.eswa.2024.124567. URL https://linkinghub.elsevier.com/retrieve/pii/S0957417424014349.

Junaid Rashid, Syed Muhammad Adnan Shah, and Aun Irtaza. Fuzzy topic modeling approach for text mining over short text. *Information Processing & Management*, 56(6):102060, November 2019. ISSN 03064573. doi: 10.1016/j.ipm.2019.102060. URL https://linkinghub.elsevier.com/retrieve/pii/S030645731930041X.

Itai Himelboim, Marc A. Smith, Lee Rainie, Ben Shneiderman, and Camila Espina. Classifying Twitter Topic-Networks Using Social Network Analysis. *Social Media + Society*, 3(1):2056305117691545, January 2017. ISSN 2056-3051. doi: 10.1177/2056305117691545. URL https://doi.org/10.1177/2056305117691545.

Yohji Akama and Atina Husnaqilati. A dichotomous behavior of Guttman-Kaiser criterion from equi-correlated normal population, January 2023. URL http://arxiv.org/abs/2210.12580. arXiv:2210.12580 [math].

Giordano Paoletti, Jussara M. Almeida, Luca Vassio, Marcos André Gonçalves, and Marco Mellia. Join the Chat: How Curiosity Sparks Participation in Telegram Groups. *Proceedings of the International AAAI Conference on Web and Social Media*, 19:1493–1509, June 2025. ISSN 2334-0770. doi: 10.1609/icwsm.v19i1.35884. URL https://ojs.aaai.org/index.php/ICWSM/article/view/35884.

Federico Barravecchia, Luca Mastrogiacomo, and Fiorenzo Franceschini. Categorizing Quality Determinants in Mining User-Generated Contents. *Sustainability*, 12(23):9944, January 2020. ISSN 2071-1050. doi: 10.3390/su12239944. URL https://www.mdpi.com/2071-1050/12/23/9944.

Federico Barravecchia, Giacomo Panzuti, and Luca Mastrogiacomo. ANALISI DELLA DIGITAL VOICE OF CUSTOMERS TRAMITE TOPIC MODELLING E CARTE DI CONTROLLO A MEDIA MOBILE.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research Volume: 3*, 2003.

Daniele Mazzei, Reshawn Ramjattan, Daniele Mazzei, and Reshawn Ramjattan. Machine Learning for Industry 4.0: A Systematic Review Using Deep Learning-Based Topic Modelling. *Sensors*, 22(22), November 2022. ISSN 1424-8220. doi: 10.3390/s22228641. URL https://www.mdpi.com/1424-8220/22/22/8641. Company: Multidisciplinary Digital Publishing Institute Distributor: Multidisciplinary Digital Publishing Institute Institution: Multidisciplinary Digital Publishing Institute Label: Multidisciplinary Digital Publishing Institute.

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].

Felipe Viegas, Antonio Pereira, Washington Cunha, Celso França, Claudio Andrade, Elisa Tuler, Leonardo Rocha, and Marcos André Gonçalves. Exploiting Contextual Embeddings in Hierarchical Topic Modeling and Investigating the Limits of the Current Evaluation Metrics. *Computational Linguistics*, pages 1–41, March 2025. ISSN 0891-2017. doi: 10.1162/coli_a_00543. URL https://doi.org/10.1162/coli_a_00543.

Aly Abdelrazek, Yomna Eid, Eman Gawish, Walaa Medhat, and Ahmed Hassan. Topic modeling algorithms and applications: A survey. *Information Systems*, 112:102131, February 2023. ISSN 03064379. doi: 10.1016/j.is.2022.102131. URL https://linkinghub.elsevier.com/retrieve/pii/S0306437922001090.

Bayode Ogunleye, Tonderai Maswera, Laurence Hirsch, Jotham Gaudoin, and Teresa Brunsdon. Comparison of Topic Modelling Approaches in the Banking Context. *Applied Sciences*, 13(2):797, January 2023. ISSN 2076-3417. doi: 10.3390/app13020797. URL https://www.mdpi.com/2076-3417/13/2/797.

Xiaobao Wu, Thong Nguyen, and Anh Tuan Luu. A survey on neural topic models: methods, applications, and challenges. *Artificial Intelligence Review*, 57(2):18, January 2024. ISSN 1573-7462. doi: 10.1007/s10462-023-10661-7. URL https://doi.org/10.1007/s10462-023-10661-7.

Shuhei Watanabe. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance, September 2025. URL http://arxiv.org/abs/2304.11127. arXiv:2304.11127 [cs].

Annibale Panichella. A Systematic Comparison of search-Based approaches for LDA hyperparameter tuning. *Information and Software Technology*, 130:106411, February 2021. ISSN 0950-5849. doi: 10.1016/j.infsof.2020.106411. URL https://www.sciencedirect.com/science/article/pii/S0950584920300069.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html.

Leonardo Blas, Luca Luceri, and Emilio Ferrara. Unearthing a Billion Telegram Posts about the 2024 U.S. Presidential Election: Development of a Public Dataset, 2025. URL https://www.ssrn.com/abstract=5018893.

Michal Danilák. langdetect: Port of google's language-detection library to python, 2014. URL https://pypi.org/project/langdetect/. Available on PyPI.

Matthew Hoffman, Francis Bach, and David Blei. Online Learning for Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/hash/71f6278d140af599e06ad9bf1ba03cb0-Abstract.html.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *J. Mach. Learn. Res.*, 14(1):1303–1347, 2013. ISSN 1532-4435.

Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl_1):5228–5235, April 2004. doi: 10.1073/pnas.0307752101. URL https://www.pnas.org/doi/10.1073/pnas.0307752101.

Silvia Terragni, Elisabetta Fersini, Bruno Giovanni Galuzzi, Pietro Tropeano, and Antonio Candelieri. OCTIS: Comparing and Optimizing Topic models is Simple! In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 263–270, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-demos.31. URL https://aclanthology.org/2021.eacl-demos.31.

Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the Space of Topic Coherence Measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 399–408, Shanghai China, February 2015. ACM. ISBN 978-1-4503-3317-7. doi: 10.1145/2684822.2685324. URL https://dl.acm.org/doi/10.1145/2684822.2685324.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15. Pasadena, CA USA, 2008.