

OLEKSII MINTS

**METHODS FOR DETECTING
INFORMATION MANIPULATIONS
IN DIGITAL MEDIA:
THE CASE OF YOUTUBE**

SCIENTIFIC STUDY



UDC 001.8:001.102-049.2:004.738.5

M 73

Reviewers:

Prof. Dr. Sergo Tsiramua (University of Georgia, Tbilisi, Georgia);

Ass. Prof. Dr. Tetiana Gorokhova (Centre for Advanced Internet Studies (CAIS), Bochum, Germany);

Ass. Prof. Dr. Aleksandra Synowiec (Silesian University of Technology, Zabrze, Poland)

The publication was discussed at the meeting of the Academic Council of the SMK College of Applied Science and is recommended for publication (Minutes No. 5, March 10, 2026)

Mints Oleksii

M 73 Methods for detecting information manipulations in digital media: the case of YouTube : Scientific study. Sumy : University book, 2026. 120 p.

ISBN 978-617-521-147-2

The study develops an applied methodology for detecting low-level information manipulation in online discussions and validates it using YouTube comments. The proposed approach combines analysis of temporal anomalies, assessment of emotional sentiment and semantic similarity, and measurement of reply relevance degradation using LLMs.

У дослідженні розроблено прикладну методику виявлення низькорівневих інформаційних маніпуляцій в онлайн-дискусіях та апробовано її на базі коментарів YouTube. Запропонований підхід поєднує аналіз часових аномалій, оцінювання емоційної тональності та семантичної подібності, вимірювання деградації релевантності відповідей із використанням LLM.

UDC 001.8:001.102-049.2:004.738.5

ISBN 978-617-521-147-2

© Mints Oleksii, 2026

© SMK College of Applied Sciences, 2026

© University book, 2026

Table of contents

Introduction	4
1. Literature review	6
1.1. Types of information manipulations	6
1.2. Media platforms	10
1.3. Methods for manipulation detection	14
1.4. Datasets and data collection	18
1.5. Motivation	20
1.6. Conclusions of the literature review	22
2. Methodology	27
2.1. Data preprocessing	30
2.2. Methods for identifying low-level manipulations based on activity anomalies	32
2.3. Methods for identifying low-level manipulations based on reply relevance anomalies	41
3. Results	48
3.1. Data preprocessing	48
3.2. Identification of low-level manipulations on YouTube based on activity anomaly analysis	48
3.3. Identification of low-level YouTube manipulations based on analysis of reply relevance anomalies	64
4. Statistical tests and discussion	70
4.1. Statistical assessment of model quality	70
4.2. Statistical testing of the bot-activity hypothesis	73
4.3. Research discussion	76
4.4. Research limitations	84
Conclusion	87
References	89
Abstract	97
Appendixes	98

Introduction

In modern information society, digital media plays a key role in shaping public opinion. However, alongside positive aspects such as easier, cheaper, and faster access to information, serious issues also arise related to the manipulation of the information space and the creation of manufactured consensus (Woolley, 2023). This phenomenon creates an illusion of unanimity on a specific issue, achieved through targeted use of technologies and coordinated activity in digital media. Instead of organic development of public opinion through free exchange of information and discussion, a manufactured consensus is imposed from outside, leading to a distorted view of the real situation. Manufactured consensus forms through various mechanisms, such as using bots and fake accounts to mimic mass support, artificially boosting activity metrics (likes, comments, reposts/shares), spreading disinformation, and selectively silencing opposing voices. The mechanism through which public consensus (more specifically, its subjective perception) affects an individual's decision-making relies on reflexive control theory (Lefebvre, 2015).

Manipulations of public opinion on digital media have become a serious threat: worldwide, political organizations and governments use social platforms to spread disinformation, undermine trust in institutions, and engage in other forms of information influence (Bradshaw & Howard, 2018). García-Orosa (2021) identified four waves in the development of digital democracy, each accompanied by new tools for communicating with potential voters and influencing them. The latest, fourth wave, at the time of publication, according to García-Orosa, includes elements such as disinformation, social media, bots, and astroturfing.

The beginning of this wave dates back to 2016, when interference by the Russian “troll factory” IRA (Internet Research Agency) was uncovered in the U.S. presidential election and the Brexit referendum (García-Orosa, 2021; Schoch et al., 2022). Among recent global examples of using coordinated media activity to forge manufactured consensus, the

U.S. presidential election in 2016 and the Anti-Vaccine Movement during the COVID-19 pandemic stand out (Woolley, 2023). Similar mechanisms are used by the Russian Federation to justify its military actions (Vasara, 2020). The danger of these activities is that they distort the true picture of public opinion, erode trust in information in the digital space, and contribute to societal polarization (Lazer et al., 2018). Although there are current reasons to identify a subsequent, fifth wave of digital democracy connected to AI use (Vaiqoh & Astuti Nurhaeni, 2024), these influence tools remain relevant and effective.

The creation of coordinated, inauthentic media activity aimed at manufacturing consensus in the modern information landscape is a complex process involving various actors with different goals. Recognizing these actors, understanding their motives, and developing effective methods and tools to identify them are crucial for countering manipulation and disinformation.

1. Literature review

The field of manipulation detection in digital media is rapidly expanding in academic research. The number of publications dedicated to this phenomenon is so large that a structured review becomes necessary. First, we identify the main types of information manipulation recognized by researchers. Then, we examine the features of different media platforms and discuss the reasons for choosing them for manipulation studies. Next, we review methods for studying various types of information manipulations, trace the development of these methodologies, and consider the main approaches to data collection, dataset construction, and their applications. At the end of the review, we explore the motivations behind information manipulation and summarize the key findings of our analysis.

1.1. Types of information manipulations

Academic literature discusses many forms of manipulative activities and their goals. The scope of information manipulations ranges from individuals creating fake accounts for messaging on dating websites to political campaigns that sway attitudes across entire countries. Therefore, without claiming to be comprehensive, we focus on selected types of information manipulations relevant to this study.

Astroturfing

This term first appeared in 1985 in a speech by U.S. Senator Lloyd Bentsen, who was flooded with identical letters as part of a fabricated letter-writing campaign. Today, it is used to describe disguising artificial public support as a grassroots initiative. Astroturfing is a type of coordinated disinformation campaign that creates the illusion of widespread support for certain initiatives or “bottom-up” narratives, as if they come from regular people. This distinguishes it from classical forms of disinformation, where false beliefs are imposed “from above” (through advertising or political propaganda). Besides its use in business, where astroturfing is considered a form of low-budget or “guerrilla” marketing

(Dimitrieska, 2025), it is increasingly employed for political aims. Keller et al. (2020) even introduced the term “political astroturfing” for cases where a group of controlled accounts pretends to be independent citizens and extensively broadcasts prepared messages.

Such campaigns may involve both software bots and genuine users (trolls). Research indicates that astroturfing can be detected through digital traces like message uniformity, synchronized actions, and content-behavior linkage patterns. Alieva et al. (2022), examining a campaign to discredit opposition politician Navalny on Russian-language Twitter, discovered that hundreds of accounts simultaneously promoted hashtags against him, acting as a unified entity. Schoch et al. (2022) also analyzed a campaign by the Russian IRA on Twitter. Their tweets were deliberately crafted and influenced media agendas until they were uncovered. Therefore, many studies focus on coordinated information operations. Methods for analyzing them range from content analysis (identifying which “agenda” is promoted) to graph analysis (how accounts are connected internally and to external websites).

Bots

Initially, this term was used to describe a program created to perform specific tasks in the digital space without direct human involvement. Software bots can automate routine operations, gather information, and interact with users. However, they can also mimic the behavior of real users and be used to manipulate the information environment and public opinion. Today, the term “bot” encompasses many types of automated accounts. Akhtar et al. (2023) identify 10 different types of bots based on their activity goals (for example, Spam bots, Follower bot, Astroturf bot, Trigger bot, etc.). Ellaky et al. (2023) identify 11 bot types across three categories: Benign, Neutral, and Malicious.

In addition, bots are categorized by implementation technology. Researchers differentiate between automatic software scripts (the most primitive bot type today), partially automated accounts (cyborgs) where a person occasionally intervenes, and bots with AI features (Ellaky & Benabbou, 2024). Cyborgs are the hardest to detect because they behave almost like regular users but activate scripts for mass actions at critical moments. For example, a type of cyborg has been documented that suddenly conducts mass messaging and posts very frequently to support a candidate, while remaining inactive at other times (Ellaky & Benabbou, 2024). Traditional bots, which were just simple software scripts, are

now easier to detect, but the rise and development of generative AI have greatly complicated their behavior and, consequently, their detection.

Trolls and “troll factories”

The term “troll” is sometimes used to describe real users (or groups of users) who intentionally provoke conflicts, insults, or disinformation, often pretending to be fictional characters (Marcondes et al., 2024). In the context of astroturfing, however, even real users are more frequently labeled as “bots.” Troll research aims to expose their identities and tactics. One of the first major cases was the revealed network of the IRA “troll factory” from Saint Petersburg. Linvill and Warren (2020) detailed how these operators created thousands of profile accounts and posed as ordinary Americans from different political backgrounds to “engage in dialogue from both sides” and radicalize it. Trolls often work with bots to engage in more subtle psychological manipulation. Therefore, troll detection methods are also similar: researchers look for unusual stylistic features, atypical activity hours, and, most importantly, signs of network coordination (Marcondes et al., 2024). Trolls’ goals may not only be political but also commercial – for example, groups producing fake reviews of competitors’ products.

Fake followers

This type of manipulation focuses on creating an illusion of popularity. Since the number of followers is a key metric of online popularity, the “fake followers” service is highly sought after by both commercial accounts and individual social media users. Akhtar et al. (2023) rank the follower bot second in popularity after spam bots. Cresci et al. (2015) demonstrated that entire bot farms sell follower-boosting services and proposed an effective (as of 2015) method called “Digital DNA” for their detection. Key signs of fake followers include mass account creation simultaneously, minimal activity besides following, identical avatars or bios, and numeric or randomly generated names. In studies of manipulation, detecting inflated follower counts is crucial because it enhances other influence tactics. For example, politicians or brands with millions of followers – whether real or fake – appear more popular and attract genuine followers. In turn, fake popularity can be used to validate false messages, such as in astroturfing campaigns.

Disinformation content (fake news, conspiracy theories)

Along with methods for detecting actors (bots, trolls), a research direction is also developing to recognize manipulative messages and

narratives. In some studies, this task is treated as an application of NLP methods (Althubiti et al., 2022; Mundra et al., 2023). Research that combines content analysis with network analysis is also of interest. For example, Guarino et al. (2020) studied networks of Twitter accounts that shared links to specific propaganda websites. They found that such accounts form clusters around particular domains (for instance, a group of accounts that almost exclusively share links from one pseudo-news site). This allowed them to speak about *propaganda distribution networks*. Cross-platform diffusion is also noted, when the same false information is broadcast simultaneously through several channels – Twitter, Facebook, YouTube, etc. (Pote, 2024). Wilson and Starbird (2020) described specific cases where links to the same disinformation resource were coordinatedly posted on Twitter and Facebook, reinforcing each other's effects.

Other tactics

This includes everything that did not fall into the categories above but has been studied by scholars: *fake trends* (Kausar et al., 2022); *deepfake media manipulations* (Tong et al., 2024; Mints & Sidelov, 2022). The deepfake problem became especially relevant in 2025 due to the development of generative AI models (NanoBanana, Sora, Seedream, etc.) that enable the creation of realistic videos featuring specified characters. The availability of these services and the quality of the generated images and videos already suggest a dramatic increase in the impact of deepfakes on news media (Lundberg & Mozelius, 2025). Finally, in recent studies, researchers have paid attention to virtual influencers – accounts of synthetic personalities that first gain popularity through content created with generative AI and are later used for native promotion of needed ideas (Lee et al., 2025). At present, the main area of application of virtual influencers is marketing, but the boundary between marketing and manipulation is quite thin. Methodologies for studying this phenomenon are still being formed.

Thus, today this topic is a broad interdisciplinary field that covers diverse types of information manipulation in digital media, and the emergence of new types and tools of manipulation will guarantee the intensive development of this field in the future as well. Importantly, countermeasures tested against one type of threat are often applicable to others. For example, clustering methods developed for bot detection also allow successful detection of troll communities, and network

analysis of accounts that spread propaganda helps expose hidden operations regardless of the platform. A challenge in using academic results is that the manipulation landscape constantly changes under the influence of internal and external factors, necessitating continuous updating of observations and knowledge in this area.

1.2. Media platforms

Twitter

The vast majority of studies focus on Twitter, largely because, for a long time, this platform provided open access to data through an API, which many other social networks do not offer (Pote, 2024). Another reason why Twitter became the main platform for studying bots and the mechanisms of disinformation spread is that Twitter is widely used by politicians and other influential figures. Large information campaigns have repeatedly been observed on the platform.

Twitter has been studied, for example, in analyses of the spread of crisis events such as the COVID-19 pandemic (Schoch et al., 2022), and as a tool for influencing voting outcomes, for instance the U.S. elections in 2016 (Kollanyi et al., 2016) and 2020 (Ferrara et al., 2020), the Brexit referendum in 2016 (Howard & Kollanyi, 2017), and elections in other countries as well (Mints, 2025-a).

Due to the openness of the platform's data (before the change of ownership), many methods for bot detection on Twitter have been developed, along with benchmarks for comparing them. For example, the large, open TwiBot-20 dataset contains data on about 230,000 accounts and 33 million tweets for bot recognition tasks (Ellaky & Benabbou, 2024).

At the same time, researchers emphasize that focusing only on "Twitter bots" leaves gaps. Many astroturfing campaigns also rely on real (non-automated) accounts that act synchronously. Therefore, recent Twitter studies shift attention from individual accounts to the detection of coordinated account networks. Schoch et al. (2022) analyzed all exposed Twitter astroturfing campaigns and found that, on average, 74% of accounts in each campaign participated in joint posting (simultaneous tweets/retweets of the same messages), a behavior almost never observed among ordinary users. This indicates that Twitter campaigns leave "network traces" that can be used to detect them.

Facebook and Instagram

Despite Facebook's huge audience, there are fewer academic works on it, mainly due to restricted data access. Nevertheless, there are studies that examine covert manipulations on these platforms as well. For example, Schler & Bonchek-Dokow (2024) studied astroturfing on Facebook during various election campaigns and collected several million comments from politicians' pages. The researchers identified anomalous profiles on these platforms and proposed a method for detecting them using a combination of manual labeling and machine learning.

Among the most typical features of astroturfing on Facebook are unnatural, synchronized activity and the use of repeated comment templates over time. Using these patterns enabled training a classifier with an accuracy of about 92% (Schler & Bonchek-Dokow, 2024). Research also addresses Instagram, where campaigns of state trolling on the platform have been studied. For example, pro-government attacks on the Iranian diaspora on Instagram were analyzed (Schoch et al., 2022).

As noted in the cited sources, analyzing manipulations on Facebook/Instagram, unlike Twitter, requires labor-intensive web scraping (Pote, 2024), which has constrained their study. At the same time, the importance of these platforms is very high, and coordinated operations have been identified there as well. For instance, according to the Oxford Internet Institute project, cases of manipulative influence via Instagram, WhatsApp, Snapchat, and others were recorded in 12 of 48 examined countries (Bradshaw & Howard, 2018).

Reddit and forums

Platforms like Reddit have been studied less often. Reddit is a large news forum with a voting mechanism that, in theory, allows the hidden promotion of desired narratives (for example, through mass registration and coordinated voting by groups of bots or trolls). However, the rating mechanism (karma) makes it difficult for a large group of bots whose opinions would carry weight to appear quickly. Therefore, there is almost no data on large-scale influence operations on Reddit, although individual cases are known in which propagandistic political messages were posted in U.S. subreddits from accounts of unknown origin.

One bold experiment was conducted by Swiss researchers: they secretly launched AI-based bots in the *ChangeMyView* subreddit to test whether machines can influence participants' opinions. The experiment

caused a scandal, but it showed that the audience does not always distinguish a human from a program.

In academic literature on Reddit, studies also appear on detecting disinformation in topic communities. For example, in medical subreddits, bots were used to automatically filter harmful health myths (Sager et al., 2021). In general, Reddit remains a less studied field compared to Twitter or Facebook. This is partly because systematically collecting Reddit data is more difficult (external archives like Pushshift are needed) and partly because there are no scandals as high-profile as elections and bots on Twitter. Nevertheless, a number of studies show coordinated activity on forums, for example, by tracking repeated messages and links posted under different accounts, which can indicate hidden coordination (an analogue of astroturfing).

YouTube

The YouTube platform, as the world's largest video hosting service, is relatively understudied in the context of information manipulation, underscoring the scientific novelty of studies that focus on it. The main difficulty is the closed nature of recommendation algorithms and viewership data. Nevertheless, some researchers call YouTube a "powerful channel" for spreading disinformation worldwide. According to Serrano (2024), in 2022, more than 80 fact-checking organizations publicly accused the platform of enabling false content, including COVID-19 conspiracy theories, false election narratives, and other misinformation.

Among academic studies of propaganda spread through video, one can note the article by Allgaier (2019), which analyzed the ecosystem of climate videos on YouTube and found that algorithms recommended many videos denying climate change, contributing to a distorted picture. In another study (Calvo et al., 2022), a network analysis of YouTube recommendations was conducted, and a viewing graph was built, showing how videos with anti-vaccine disinformation cluster and direct viewers to each other. Thus, YouTube can draw users into information bubbles, but it remains unknown whether this is intentional on the platform's part or the result of users exploiting weaknesses in the recommendation algorithm.

It is also important to note that some sentiment analysis studies focus on comments under YouTube videos (Pradhan, 2021; Cunha et al., 2019). Some researchers point out the presence of "toxic" comments and suggest methods for analyzing them (Obadimu et al., 2019; Döring, 2020).

Thus, although the facts of deliberate propaganda distribution by the YouTube platform have not been proven, there is potential for YouTube to be used to manipulate by creating a manufactured consensus in comments. Therefore, the study of manipulations on YouTube is important for filling gaps in scientific knowledge.

Telegram u messengers

Telegram, WhatsApp, and similar platforms are more often perceived as private communication channels. However, public Telegram channels and groups are widely used for propaganda and campaign coordination, especially in countries where Telegram is popular (for example, Russia, Ukraine, Iran). Recently, Telegram has become a popular object for academic research. First, it is worth noting the work by Kireev et al. (2025), which presents an effective method for detecting propaganda networks on Telegram. The researchers analyzed 17 million comments from 13 political and news-oriented channels. The study identified two networks of connected accounts: a large pro-Kremlin one and a smaller pro-Ukrainian one. The authors note a feature of the manipulators' work: they did not start topics themselves but posted mass replies to other people's comments using specific keywords (for example, "Putin" or "Zelensky"). At the same time, they posted identical messages in different places, which ordinary users almost never do. As a result, the study detected a dense network of cross-posting from propagandist accounts, which differs sharply from the scattered behavior of ordinary commenters.

In Kireev et al. (2025), a propaganda detection model was also developed, achieving an effectiveness of about 97.6%, exceeding human moderation by 11.6%. At the same time, it should be noted that such a model has a rather narrow specialization.

Analyzing emoji reactions in a Telegram community, Mints (2025-b) also notes a significant number of propagandist messages and an uneven distribution of them over time: important events are accompanied by increased bot activity. At the same time, the author notes a specific feature of Telegram: the self-organization of members of online communities to counter propaganda and other forms of manipulation.

Analyzing messages in Telegram chats in Brazil, Júnior et al. (2021) concluded that this platform is actively used by the country's political groups to promote their agenda and fight opponents.

Thus, Telegram, especially in countries and communities where it is widespread, is actively used for propaganda and manipulation. At

the same time, both academic research and practical countermeasures remain insufficient, and the effectiveness of manual moderation is too unstable (Kireev et al., 2025 indicate that 20% to 95% of “injected” messages were detected manually). The analysis also highlights the relevance of studying propaganda that is carried out not by content authors but by its consumers.

1.3. Methods for manipulation detection

To detect hidden information influence, researchers have developed and applied a wide range of methods. The choice of a method and its expected effectiveness depend on many factors, including the type of task, its complexity, and correct formulation (Mints, 2017), available computing resources, data completeness, and more. Additionally, different types of media platforms have their own specific features in how information influence manifests (Hargittai & Sandvig, 2015). Let us consider the main approaches to their detection.

Analysis of temporal anomalies (activity spikes)

Many manipulative campaigns are characterized by an unnatural activity pattern over time – for example, bursts (burst) of publications, that is, the appearance of many messages (often identical or very similar) within a short period. In many cases, this happens synchronously from different accounts. Detecting such spikes is a simple but effective indicator of coordinated activity. In particular, bots and “cyborgs” often publish posts at an abnormal human rate (dozens per hour, 24/7), which immediately distinguishes them from real users (Pote, 2024).

In addition to posting messages, simultaneous actions by a group of accounts can also indicate coordinated activity. For example, if dozens of accounts retweet the same content within one minute, the probability of coordinated artificial boosting of trends is high. Time-series analysis approaches automate the detection of such anomalies. For example, Abu-El-Rub & Mueen, (2019) developed the *DeBot* algorithm, which clusters tweets by simultaneity and similarity of content, highlighting groups of suspiciously synchronized accounts. Analysis of temporal anomalies can also be applied for event-based detection of astroturfing – for example, recording sharp spikes in mentions of a candidate exactly during a TV debate or immediately after a scandal, which may indicate artificial forcing of the topic.

Sentiment analysis

This method aims to detect a hidden shift in the average sentiment of content, which often accompanies propaganda campaigns. In this case, bots or trolls generate large numbers of messages with a specific emotional tone, usually extremely negative or extremely positive. Audience polarization is also beneficial for manipulators, as people in an emotional state are much more easily influenced.

Sentiment analysis enables measuring the emotional tone of a large number of messages and detecting anomalies – spikes in the negative or positive direction. For example, Hagen et al., (2022) showed that a network of bot accounts on Twitter significantly increased the negative emotional background of a political discussion by spreading mainly outrageous content and fake “likes,” thereby polarizing the audience. In some works, sentiment features are extracted from an account profile: researchers estimate the average emotional tone of its posts and compare it with a norm (Bahrainian & Dengel, 2013; Saif et al., 2012). It was observed that bot farms can act as a “negative amplifier,” increasing anger and distrust toward opponents. In this case, they can be detected by the extreme emotional tone of messages or by the synchronous release of many equally aggressive comments (Ellaky & Benabbou, 2024). Therefore, sentiment analysis is an important tool for manipulation detection.

Network analysis

Approaches in this group consider links between accounts – follows, retweets, replies, and shared use of hashtags and links – to identify clusters of abnormal interaction. The logic of this approach assumes that manipulations are usually organized by groups of accounts acting in coordination, so their connections form distinguishable structures in the graph of a social network (Schoch et al., 2022). Thus, because bots often follow each other or the same few nodes, dense clusters form in the friends graph. Similarly, the retweet and mention graph can be studied: if certain accounts disproportionately retweet each other’s messages, this may indicate pre-planned coordination.

Network analysis methods include calculating centrality metrics, community detection, and identifying “stars” and “echo chambers.” For example, Zannettou et al., (2019) analyzed the interaction network of the IRA “troll factory” on Twitter and found that these trolls formed a separate community with strong mutual ties and weak penetration

into the rest of the network. In Keller et al., (2020), network analysis was used to detect political astroturfing by searching for coordination patterns. Vargas et al. (2020) used network analysis to detect disinformation campaigns on Twitter in which the same narratives were spread under different accounts. It was shown that when a group of accounts is centrally managed, repeated action sequences appear in their activity logs (e.g., the simultaneous use of the same hashtags), enabling the distinction between a fake campaign and spontaneous civic activity.

Thus, network analysis is one of the most powerful tools because it captures the core of manipulation: its collective nature. In addition, network analysis can be effectively combined with other features – for example, first build graphs of “who follows whom” or “who retweets what,” then identify anomalous subgraphs, and then check the content of messages in these subgraphs for propaganda (Kireev et al., 2025).

Machine learning methods

With the development of ML, methods that automatically learn to recognize manipulators based on many features have become more popular. Usually, such approaches represent accounts or content as multi-dimensional vectors (embeddings) and then apply classification or clustering algorithms. For each type of information manipulation, specific approaches and success metrics have been developed. For example, for bot detection – classification accuracy; for campaign detection – completeness of network detection; for content fakes – accuracy of message classification, etc. There are many ML-based methods and approaches for bot detection. Ellaky et al. (2023) describe 8 main approaches and, within them, identify 65 specific methods and algorithms. Therefore, we focus only on the most popular methods.

Unsupervised clustering makes it possible to find natural groups of similar accounts – for example, accounts that generate similar content and have similar time patterns may form a bot network. Thus, the DeBot algorithm uses graph clustering of tweets to identify groups of bots on Twitter (Ellaky & Benabbou, 2024). Solving the task of detecting purchased followers, Cresci et al. (2015) introduced the concept of an account’s “digital DNA” and clustered accounts by the similarity of this Digital DNA.

For content clustering, the content is first transformed into vector embeddings, which are then clustered. In this way, one can, for example, find groups of semantically similar messages published by differ-

ent accounts, which is a common sign of an information campaign. This approach is described in Barbero et al. (2023). In manipulation detection tasks, embedding clustering is more commonly applied to text, although in theory it can be applied to any type of content, including video and audio.

The most diverse group of methods for studying information manipulation is supervised learning. Both classical algorithms (decision trees, SVMs, and random forests) and deep neural networks are used. Varol et al. (2017) tested Random Forests, AdaBoost, logistic regression, and decision trees classifiers for bot detection on Twitter. The best classification performance, with an AUC of 0.95, was achieved by the Random Forest algorithm. (Varol et al., 2017). Modern studies use more advanced methods, for example, the ensemble method ESC (Ensembles of Specialized Classifiers), which combines several models for different feature types (content, network, temporal, etc.) and reaches $AUC > 0.99$ in tasks of isolating political bot networks (Sayyadiharikandeh et al., 2020). An important trend is the introduction of coordination-detection methods: modern models aim to learn to recognize not only individual bots but also coordinated groups. Techniques are used that consider several aspects at once (for example, the work of Xiao et al., (2022) combines self-supervised learning, attention-based polarity inference, and adversarial training). The key limitation of supervised learning is the need for a labeled dataset to begin training. Labeling, that is, matching reference answers to input data vectors, is extremely labor-intensive, especially for large datasets. Therefore, the number of studies significantly exceeds the number of datasets used.

It is almost impossible to predict in advance which ML method will be most effective (in terms of accuracy, resource intensity, development costs, etc.). Therefore, a significant part of the studies is devoted to comparing different ML methods on specific tasks. For example, Mostefai et al. (2025) compare the effectiveness of Artificial Neural Network with Multi-Layer Perceptron, Support Vector Machine (SVM), and Multinomial Naive Bayes (MNB) for Identifying Textual Propaganda on Social Media. This and similar studies simplify the task of building an ML-based manipulation-detection system.

1.4. Datasets and data collection

When studying manipulations in digital media, an important issue is the choice of data source and the methods used to collect the data. The key question is how to obtain a representative set of content and accounts that includes examples of manipulations. A typical choice here is to either collect data independently for a specific study or use public datasets. Let us consider both options and their further details.

Independent data collection is implemented either by using the official APIs of the studied platforms or through web scraping.

Data collection via official APIs

Many studies on Twitter, YouTube, and Telegram use the open APIs of these platforms for data sampling. For example, for a long time, Twitter allowed users to download tweets for free by keywords, hashtags, or from specific users. The availability of an accessible and user-friendly API explains the large number of academic publications based on the Twitter platform [Alieva et al. (2022), Cresci et al. (2015), Feng et al. (2021), Pratama & Rakhmawati (2019), etc.]. Using an API makes it possible to obtain data immediately in a structured format (for example, JSON), and some APIs also provide metadata that is not even visible in the user interface. For some platforms, API use is strongly limited. For example, the Reddit API allows downloading only the last 1000 posts and does not support date-based search. Telegram, although it does not have a public API for channel history, still allows data export via program scripts using the open Telethon library. Therefore, this platform is also accessible for connecting to public channels and exporting chat histories. In Kireev et al. (2025), comments from 13 channels were collected in this way. However, not all platforms have APIs, and even where they exist, their use may be limited. Thus, after the change of ownership, Twitter's access to platform data became significantly restricted.

Web scraping and unofficial methods

For platforms without an open API or with limited data access (Facebook, Instagram, some forums), researchers have to parse HTML pages or use paid scraping services that mimic an ordinary user. This method of data collection is labor-intensive and often requires bypassing restrictions (e.g., CAPTCHA and authorization). In addition, it is impossible to obtain any metadata in this way, as with APIs. In their study of Facebook astroturfing, Schler & Bonchek-Dokow (2024) note that they

collected comment data manually using a developed script. Reddit data are often obtained from the Pushshift archive, which parses Reddit and provides an almost complete copy of comments and posts over many years (Pushshift), but access to these data is limited and is provided on the basis of a signed agreement. In general, scraping enables the collection of valuable datasets, but it is associated with several limitations, requires more resources, and calls for ethical reflection on the results.

Data stored in media-platform archives is usually incomplete, as it has already undergone moderation, and some messages have been deleted. Therefore, in some cases, when it is necessary to track in real time the response of social media to the unfolding of a socially sensitive event, researchers build a dataset using specialized software for monitoring media sources. The main principle is to cover a relevant slice of data where the risk of manipulation is high (for example, before elections, during public crises, or around a scandal). Then, manual labeling is often used with experts, crowdsourcing, or LLMs to mark sample examples of propaganda, bots, and so on.

Publicly available datasets

Because collecting and labeling one's own datasets is difficult, many studies use publicly available datasets as their data source. Some platforms publish their own data on detected abuses. For example, since 2018, Twitter has maintained an archive of information operations by publishing identifiers and content of accounts linked to state propaganda campaigns (Schoch et al., 2022). These data became a source for many studies. They were also used by Zannettou et al. (2019), Schoch et al. (2022), and other researchers. Facebook, in reports on "coordinated inauthentic behavior," also sometimes discloses examples of removed networks, though it does not provide raw data for verification. Such archives are valuable because they provide known class labels – we know for sure that these accounts were manipulators, because they were blocked by the platform administration. This makes it possible to train algorithms and conduct a retrospective analysis of influence. On the other hand, publicly available archives usually contain outdated data.

Scientific datasets

In the process of comparing the effectiveness of algorithms for manipulation detection and bot activity, the research community has identified some reference datasets that are regularly used for this purpose. These include TwiBot-20, which contains a balanced sample of

bots and ordinary Twitter accounts (Feng et al., 2021). The Cresci-2015 dataset is also widely known: it includes real accounts, purchased bots, and manual spam bots, and it serves as a test for bot detection algorithms (Cresci et al., 2015). Gilani-17 is another dataset collected in 2017, containing various indicators for bot and human accounts, and it is often mentioned in different studies (Gilani et al., 2017). For troll analysis, Twitter IRA (2018) is often used, with a volume of 298 Gb.

Similar reference datasets exist for other tasks as well. Finally, for content tasks, such as identifying propaganda in text, there are corpora of labeled news and messages. A limitation of using reference datasets is that tested methods and algorithms are often specifically tuned to a particular dataset (its parameter set, data structure, and specific features). This enables high accuracy on a particular dataset, but it does not yield strong results beyond it.

1.5. Motivation

In a comprehensive report for the Council of Europe, Wardle & Derakhshan (2017) note that the key motives behind intentional distortion or fabrication of information are political advantage, financial profit, and the intention to create conflict or attract attention (the so-called “for fun” motivation or ideological provocation). Other researchers generally share this view, identifying political, ideological, or economic motives among the factors driving actors who initiate information manipulation on social media (Lazer et al., 2018; García-Orosa, 2021; Akhtar et al., 2024; Woolley, 2023). This list does not cover all possible options (for example, Wilson & Maceviciute (2022) identify as many as seven categories of motives for creating misinformation), but it includes the main ones. It is interesting that, historically, some manipulation methods had a rather clear purpose. For example, astroturfing, soon after its appearance, was used as a marketing tool and, therefore, for purely economic goals (García-Orosa, 2021). However, this specialization later became disrupted and is now almost absent.

Political motivation of inauthentic media activity is typical for state structures that aim to build a positive image of the authorities, promote their political agenda, or discredit the opposition (Wilson & Maceviciute, 2022; Bradshaw & Howard, 2018; Zannettou et al., 2019). Intelligence

services can also use these methods in information wars to destabilize situations in other countries (Vasara, 2020). In turn, inauthentic media activity, often based on the use of bots and algorithms, may be initiated both by state actors to strengthen control over the information space and by private individuals or groups to achieve their own goals (Kireev et al., 2025; García-Orosa, 2021; Akhtar et al., 2024). In general, political actors such as political parties and social movements often use methods of manufacturing consensus to promote their ideas and mobilize supporters. An example of inauthentic media activity driven by political motives is the 2016 U.S. presidential election (Kollanyi et al., 2016; Hughes & Waismel-Manor, 2021).

Inauthentic media activity driven by *ideological motives aims to promote a belief system, worldview, or set of values*, often to shape social norms and influence cultural discourse (Wilson & Maceviciute, 2022; Wardle & Derakhshan, 2017). For this purpose, media platforms are used to spread ideological content, attract supporters, and create communities around shared beliefs. This may include promoting specific narratives, demonizing opposing ideologies, and strengthening group identity (Woolley, 2023; Zannettou et al., 2019; Synowiec, 2025). For example, a religious group may use social networks to spread its beliefs and attract new members, or an activist group may use online platforms to challenge dominant ideologies. An example of inauthentic media activity driven by ideological motivation is the Anti-Vaccine Movement (Calvo, Cano-Orón, & Llorca-Abad, 2022; Sager et al., 2021). Wardle & Derakhshan (2017) note that in some cases such motivation takes the form of “ideological provocation” and may be carried out simply for fun. The absence of a clear motive complicates the process of isolating such manipulations.

Economic motivation lies behind the media activity of commercial organizations, including corporations and PR agencies, which use these methods to promote goods and services, build a positive brand image, and compete with rivals (Wilson & Maceviciute, 2022). Inauthentic media activity in this context may appear as artificially inflated popularity metrics and the creation of fake reviews and comments (Cresci et al., 2015; García-Orosa, 2021).

It is also important to consider individual actors such as bloggers, opinion leaders, and activists, who can use their popularity and influence to promote certain ideas and build manufactured consensus around them (Woolley, 2023; Lee et al., 2025). Their motivation, in addition to

what is described above, may be based on a desire for *personal popularity or recognition*. Individual actors may rely both on coordinated actions with their supporters and on the use of bots and other tools of inauthentic media activity (Woolley, 2023; Akhtar et al., 2024).

Correct identification of the motivation behind information manipulation contributes to a better understanding of the related processes and, therefore, helps find more effective ways to counter them.

1.6. Conclusions of the literature review

Fig. 1 shows a scheme of correspondence between different types of information manipulations and their initiators.

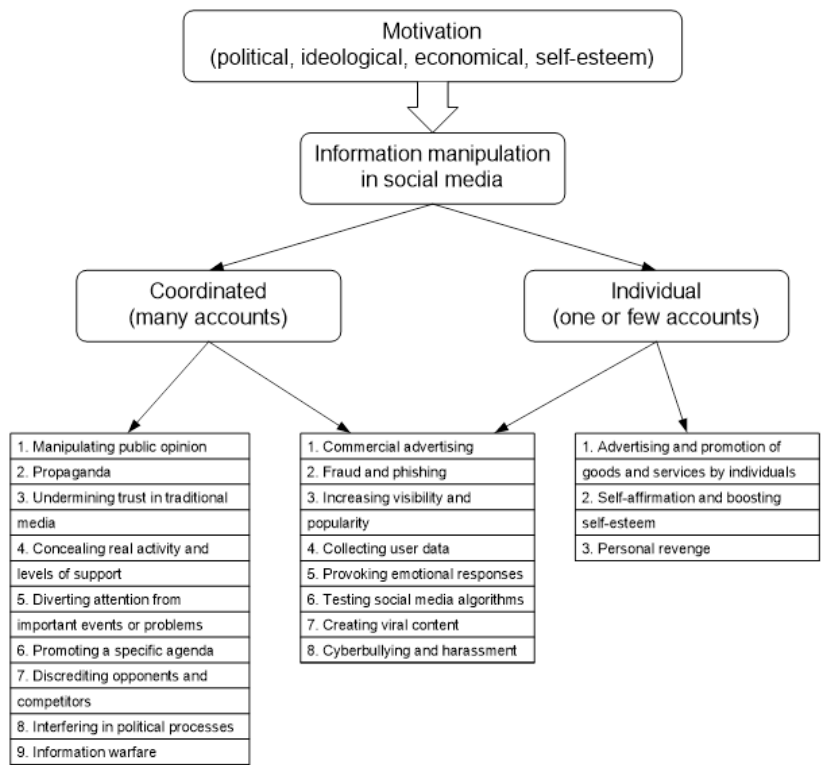


Fig. 1.1. Types of information manipulations from the perspective of motivation

It is important to understand that the boundaries between these categories may be blurred, and one actor may pursue several goals at the same time. However, in general, Fig. 1.1 clearly shows that coordinated activity driven by political or ideological motives requires the largest number of accounts.

For accounts that participate in coordinated media activity, we will use the term “bot,” regardless of who or what stands behind the account – a program, AI, or a human troll. This corresponds to the common connotation of a person who behaves like an automated program: acting mechanically, repeating the same actions, or following someone’s instructions without critical reflection. Many studies (for example, Ellaky et al., 2023; Hegelich & Janetzko, 2016) note that the boundary between bots and trolls is blurred: organized groups of real users acting “according to a manual” are almost indistinguishable from bots in their effect. In the media space, it is practically impossible to distinguish the actions of such a person from the work of a program (especially one using generative artificial intelligence). Therefore, we will focus on developing methods for bot detection of any nature.

The increasing complexity of bot behavior (human-driven or AI-based) has simultaneously made their detection more difficult. The literature review above shows that in a number of studies, high accuracy in identifying information manipulation (above 95%) has been achieved, but it should be noted that these studies rely on network analysis and are conducted on datasets collected from dozens to hundreds of accounts. The volumes of such datasets are measured in the tens and sometimes hundreds of gigabytes (for example, the Twitter IRA (2018) dataset has a volume of 298 GB). From a practical point of view, creating and analyzing such a dataset is difficult to implement and requires significant resources (including time, financial, computing, and intellectual).

Therefore, in our study, we will focus on detecting such anomalies associated with signs of inauthentic media activity that can be identified based on analysis of comparatively small volumes of information. In particular, these include:

- coordinated activity spikes, identical publication times, repeated actions, and other unnatural activity patterns;
- use of identical (or similar) content by different users, promotion of the same narratives;

- a characteristic user profile: minimal personal information, stock photos, recently created accounts, generated nicknames;
- irrelevant posting (for example, spreading political narratives in comments under neutral posts, while ignoring their topic and discussion history).

Since manipulative influence is always goal-oriented, signs of achieving these goals (for example, emotional escalation) may also serve as markers of bot activity. It is known that an audience in an emotional state of arousal becomes more suggestible and more controllable (Bronstein et al., 2019).

Each of these features alone does not guarantee that an account is a “bot,” but their combination increases the likelihood. Obviously, different types of information manipulations correspond to different sets of anomalies and, accordingly, different detection methods.

As already noted, information manipulations can be carried out both by content authors – “top-down,” and by content consumers – “bottom-up” (fig. 1.2).

A content author is potentially more influential, since each post is viewed by a large number of followers. However, building an account to a substantial level of influence is a complex, long-term process. We will refer to manipulations by content authors as *high-level* or *top-down* manipulations.

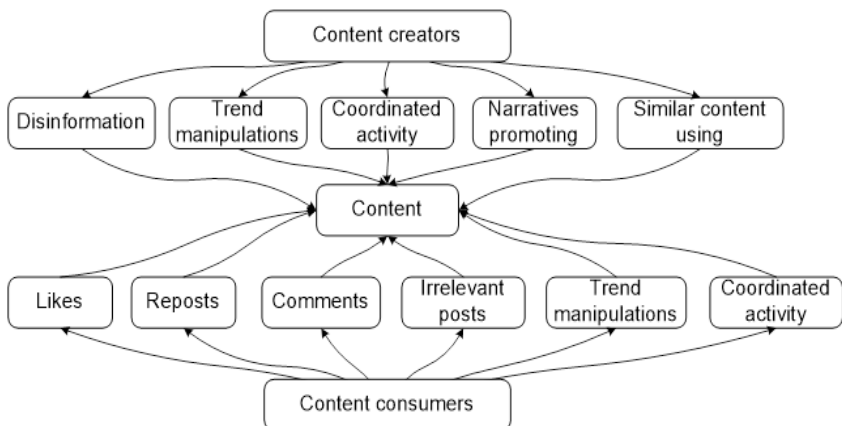


Fig. 1.2. Signs of information manipulations with content

On the other hand, each content consumer (readers, followers) has limited influence individually, but a follower account does not require prior development, so it is much easier to create and use to boost likes, reposts/shares, commenting, and other activity around content. In this case, the required effect is achieved by involving a large number of accounts. The strongest effect is observed when authors' coordinated activity is combined with consumers' coordinated activity (Linvill & Warren, 2020). We will refer to manipulations by content consumers as *low-level* or *bottom-up* manipulations.

It should be noted that the number of influential accounts on any media platform is limited due to the difficulty of developing an account to a sufficient level of influence. The narratives promoted by such accounts are relatively easy to track and identify – it is enough to regularly check information for accuracy. As for “bottom-up” manipulations, their detection is a difficult task for several reasons. First, actors of such manipulations imitate the behavior of ordinary community members, making it difficult to separate inauthentic from organic activity. Second, the effect of such influence does not appear at the level of individual messages but through the combined impact of many weak signals distributed over time and across different participants. Third, standard moderation methods focused on individual sources or specific influencers become less effective when the overall background of the discussion and the structure of network interactions are important. It is exactly low-level manipulations that are subconsciously perceived by other content consumers as the “voice of the people,” and therefore, when used skillfully, they work most effectively, especially in totalitarian and post-totalitarian communities where a “be like everyone” mentality was historically imposed.

The complexity of detecting bottom-up manipulations in social media stems from the large volume of generated data, the rapid spread of information, and the diverse content formats. For example, in the social network X (Twitter) alone, more than 100 gigabytes of text data are generated annually. This creates a need to apply various methods and data analysis technologies, mainly described above. In addition, different media platforms have distinct features for low-level manipulation. Thus, on YouTube and other visual content platforms, where user connections are weakly expressed, the main form is coordinated activity in comments. For social networks based on user interaction (Facebook, Instagram, X), reposts/shares, likes, and other engagement indicators

are most important, though comments also matter. Finally, on platforms based on text content (Telegram, Reddit, Discord channels), activity in comments is almost the only tool for low-level manipulation and the manufacture of consensus.

The goal of this study is to develop methods for detecting low-level manipulations in digital media and to test them using a platform from the list above.

2. Methodology

As mentioned earlier, different media platforms have their own specific ways in which information influences appear. It is evident that these features also lead to differences in the methodologies used to study various platforms. Although the general idea of detecting low-level manipulations is fairly universal, the implementation details depend on the platform. Therefore, we first justify the choice of a media platform for our study. Our main selection criteria include the platform's popularity among users, data access policies, and the potential for using the platform to manipulate public opinion. From this perspective, we examine the primary platforms discussed in the previous section.

Reddit. The platform architecture includes an open, documented API (in particular via the PRAW library); however, it is limited in search depth and request frequency. The platform is most widespread in the USA, although the share of active users from Europe is also estimated as high (about 36%, according to SEO Sandwich (2025)). The number of platform users is growing. The platform's vulnerability to manipulative influence is moderate. On the one hand, the content rating mechanism ("karma") and the hierarchical structure of comments allow coordinated groups to manufacture consensus by marginalizing alternative viewpoints and promoting selected narratives. On the other hand, the same mechanisms make effective external attacks by bots almost impossible, because the karma of new users, and therefore their weight in the community, is equal to zero. 82% of Reddit users identify as tech-savvy (SEO Sandwich, 2025), making this platform less typical than other social media platforms. It should also be noted that most discussions are conducted in English, which limits its broad use in non-English-speaking countries.

X (Twitter). The platform has a well-developed API for analyzing posts, but access to historical data after the change of ownership and rebranding has become significantly restricted and is now offered on a paid basis. The platform's audience in the EU is estimated at 94.8 million MAU, and there is a trend of decline (X Internet Unlimited Company,

2025). The functionality of hashtags and nested replies (threads/replies) creates a favorable environment for imitating mass support or aggression (astroturfing, dogpiling). The platform's algorithmic model, oriented toward engagement, tends to amplify artificially created trends, expanding the reach of manipulative content. Thus, X has a high vulnerability potential.

Telegram. The messenger API (accessible via the Telethon library) provides broad capabilities, enabling access to almost all platform functions. This simplifies the deployment of bot networks for data scraping and activity simulation. The European audience of the platform (meaning geographical Europe) is about 270 million users, but the number of EU users is less than 45 million (World Population Review, n.d.). The platform is especially widespread among users from Eastern Europe. A simple registration procedure, full anonymity, and the absence of strict verification allow the creation of false public opinion through mass boosting of reactions, views, and coordinated comments. Therefore, Telegram is highly vulnerable to manipulation.

Discord. The platform's API is primarily designed to support real-time operations, making retrospective analysis of posts difficult. According to the Discord Trust & Safety Team (2026, Feb.17) the Discord audience in Europe is less than 45 million users. Vulnerability to external manipulations can be classified as moderate. Although the platform is subject to coordinated attacks ("raids"), its decentralized architecture and server isolation (guilds) localize potential damage. Effective community-level moderation prevents the global spread of destructive narratives.

Facebook. Data access via the Graph API is strictly regulated, making platform analysis either almost impossible or relatively expensive (when using third-party data parsing services). The platform has an audience of 263.6 million users in Europe (Social Samosa, 2025). Vulnerability to manipulations is moderate. The real-name policy, complex registration procedure, and ranking algorithms that prioritize first-circle social ties create barriers for large-scale attacks, limiting the effective area of manipulations to public groups and pages.

Instagram. The platform is characterized by a closed API ecosystem (Graph/Basic Display) designed for business analytics, which significantly restricts third-party data access. Despite leading reach in the EU (281.8 million MAU, according to the Social Samosa (2025)), vulnerability to low-level manipulations is assessed as low. The visual dominance

of content and the lack of ergonomic tools for automating posting hinder the platform’s use for manufacturing consensus, except for some primitive forms, such as automated boosting of quantitative engagement metrics.

YouTube. The service provides an official Data API for extracting metadata and comments, but it operates within a quota system, making it difficult to extract global data. According to the Google Ireland Limited (2025, Aug.18) it has the largest audience in Europe among the considered platforms (~480.7 million). The platform uses comment ranking algorithms that take into account account history and user behavior patterns, which makes bot activity in comments more difficult. On the other hand, the spread of manipulations is supported by the fact that creating new accounts is relatively easy. There is also a large shadow market of ready-made accounts. Therefore, the platform’s vulnerability to manipulative activity should be assessed as moderate.

Consolidated information on different media platforms is presented in Table 2.1.

Table 2.1 provides a basis for an evidence-based choice of a platform for testing the methodological provisions of the study. A critical aspect is the availability of free access to textual content posted on the platform via an API. Among the considered platforms, only YouTube, Reddit (partly), and Telegram support this function. Among these platforms, YouTube has the largest audience, while Telegram is the most vulnerable to manipulations. Both platforms are of high interest for analyzing and detecting manipulative activity in digital media. In this study, we limit the analysis to YouTube because, first, this platform has

Table 2.1

Comparison of media platforms

	data access through API	user audience (EU, mln)	Manipulation vulnerability
Youtube	yes	481	moderate
Facebook	no	263	moderate
Instagram	no	281	low
Reddit	yes	~154	moderate
Discord	restricted	<45	moderate
Telegram	yes	270	high
X	yes, paid	95	high

the largest audience and, second, the lower variability of manipulation methods allows them to be examined in sufficient detail within the scope of this research.

Working with data in our study consists of three main stages: data preprocessing; manipulation detection based on activity anomalies; manipulation detection based on reply relevance anomalies. Next, we consider the methodology of these stages.

2.1. Data preprocessing

The data preprocessing stage precedes almost any data analysis (Mints, 2017). Data analysis tools cannot work directly with media resources, so it is first necessary to download the required dataset from the platform and create a local copy, and then process the data as needed for the method of further analysis.

The YouTube platform provides registered developers with access to a built-in API that simplifies the collection of primary data. The structure and description of the data obtained via this API are presented in Table 2.2.

After saving the primary dataset, it is necessary to hide user data. Although comments are published openly, publishing research results without data anonymization contradicts the ethics of scientific research (Rossi et al., 2022). Information that indirectly allows user identification is contained in the fields `authorDisplayName`, `authorProfileImageUrl`, `authorChannelUrl`, `authorChannelId`. From all these fields, we use only the first one in the research process.

Since, for identifying anomalies in user activity, it is critically important to preserve information about the correspondence between the author-text entities, instead of full anonymization, we apply pseudonymization, which we conduct in accordance with the methodology described in (Rossi et al., 2022). In the pseudonymization process, the source data file is processed by a program script that replaces each unique username in the `authorDisplayName` column with a pseudonym in the format `user-N`, where `N` is an ordinal number. The dataset obtained after pseudonymization can be used in further studies.

The final step of the first stage is calculating the main statistical characteristics of the dataset, which enables conclusions about its suitability

Table 2.2

Structure of the YouTube comments dataset obtained via API

Fields	Type	Description	Example
etag	text	Unique identifier for the resource version, used for caching and validation.	QBCC2pSQ_ Q9zAEr32pwbOkJZZWk
parentid	text	ID of the parent comment if this entry is a reply; empty if it is a top-level comment.	Ugyk6GzI_Yt_ ln6zpa94AaABAg
id	text	ID of the comment, including thread and reply information.	Ugyk6GzI_Yt_ ln6zpa94AaABAg. 9jdhZD5dGwI9jdwyh5q2YG
textDisplay	text	Final version of comment text after editing (if any).	user's comment
textOriginal	text	Comment text as initially submitted by the user.	user's comment
authorDisplayName	text	Display name chosen by the user for their YouTube account	@user
authorProfileImageUrl	text	URL to the author's profile image (thumbnail)	https://yt3.ggpht.com/POi...
authorChannelUrl	text	URL of the author's YouTube channel (if exist)	http://www.youtube.com/@user
authorChannelId	text	Unique identifier of the author's YouTube channel	UCjure4kHd- pjHf2wJmglv1
likeCount	Num	Number of likes the comment received at the time of data collection	0
publishedAt	Date	Date and time when the comment was originally posted (UTC)	2022-12-15 06:35:41
updatedAt	Date	Date and time when the comment was last edited (UTC)	2022-12-15 06:35:41

for further processing. Such characteristics include the number of observations (comments), the number of unique users, the covered period, the comment structure, and so on.

2.2. Methods for identifying low-level manipulations based on activity anomalies

To identify manipulations based on activity anomalies, we use a combined toolkit. General data analysis, including statistical processing, analysis of users' temporal activity, sentiment analysis, similarity analysis, and clustering, is implemented using Orange Data Mining, v.3.39 (Bioinformatics Laboratory, University of Ljubljana, 2025). This software uses a workflow-based approach to modeling data processing, from data import to output of results in the form of charts, tables, or data for further analysis. The overall data analysis workflow is presented in Appendix 1.

The presented workflow makes it possible to identify the following activity anomalies:

- activity spikes and time-coordinated commenting;
- the emotional tone of comments and its dynamics;
- posting of similar comments.

Below, we consider the workflow elements used to solve specific tasks.

User activity analysis is needed primarily to identify periods of increased comment activity. Exactly such periods are characterized by the widest coverage and are therefore the most attractive for manipulation. To analyze user activity by month, we discretize the data by the publishedAt field with a 1-month window (Fig. 2.1). The results are visualized as a histogram, which makes it easy to detect periods of anomalous activity. As noted above, such periods may be associated with bot activity and therefore warrant the highest level of detailed analysis.

Analysis of the semantic (emotional) tone of comments and the dynamics of their changes

To perform this analysis, the following branch of the workflow is used (Fig. 2.2).

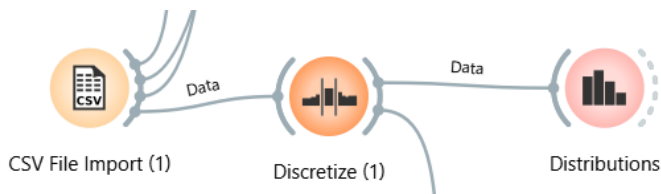


Fig. 2.1. User activity analysis workflow

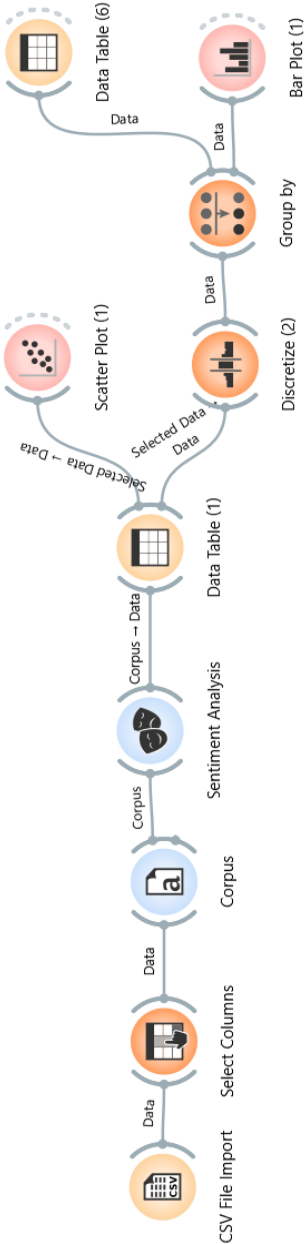


Fig. 2.2. Sentiment analysis workflow

This workflow contains the following key elements:

Select columns – selecting necessary columns only (textOriginal, publishedAt) from the dataset.

Corpus – creates a collection of text documents from the input source data. This is a standard operation for text mining.

Sentiment Analysis – calculating the emotional tone of a text (sentiment of text). This task does not have a single unambiguous solution, because even different people may perceive the emotional load of the same text differently. Nevertheless, sentiment analysis methods exist, and among them, several different approaches can be distinguished. Lexicon-based methods rely on dictionaries of emotionally colored vocabulary and on calculating their total score in a text. Machine learning assumes training models on labeled datasets where each phrase or sentence has an emotional label. Modern neural network approaches use deep learning and transformers, which allow for taking context and hidden word dependencies into account. For preliminary data analysis, it is reasonable to limit the work to lexicon-based methods, which are the fastest both in computation and in implementation, as they do not require additional preparatory work. These methods use specially created dictionaries that map lexical n-grams to their emotional scores. The simplest methods use unigrams, while more complex ones use bigrams and trigrams. Orange Data Mining software includes five sentiment analysis methods, but only one, *Multilingual Segment*, supports analysis of Russian-language texts. This is a unigrams-based multilingual sentiment lexicon described in (Chen & Skiena, 2014). As an output of the Sentiment Analysis widget, we obtain a data table enriched with a sentiment field that contains an emotional score for a post on a scale from -100 to +100, where -100 corresponds to the most negative sentiment and +100 to the most positive.

The obtained scores are saved in tabular form for further processing and are also visualized using a Scatter Plot, allowing a clear assessment of the range of emotionality across different periods. A substantial change in this range may be interpreted as a result of bot activity.

Another visualization tool in this workflow is a bar diagram of the mean sentiment level during the observed period. Unlike the Scatter Plot, this diagram shows the mean sentiment level for different time intervals.

Analysis of both diagrams helps determine how the level of emotions in comments changed during the studied period. It is established

that “emotional escalation” is one of the goals of manipulation (Ellaky & Benabbou, 2024), and the presence of signs of a sharp change in the emotional level of comments within certain time intervals can be interpreted as anomalous activity.

Finding and identifying similar comments (Similarity analysis)

The methodology for identifying identical or very similar comments in this study is based on the sequential execution of four key stages, each of which performs its own task and provides the basis for the next step in analysis. Fig. 2.3 shows the sequence and brief content of these stages, and Fig. 2.4 shows their implementation as an Orange workflow.

Let us consider these stages and the features of their implementation in the context of the study.

Stage 1. Text Preprocessing

At the preprocessing stage, textual data undergo comprehensive cleaning and normalization. First, “noise” elements are removed from comments: extra symbols, links, emojis, and other artifacts that do not carry semantic value. Next, tokenization is performed, that is, splitting

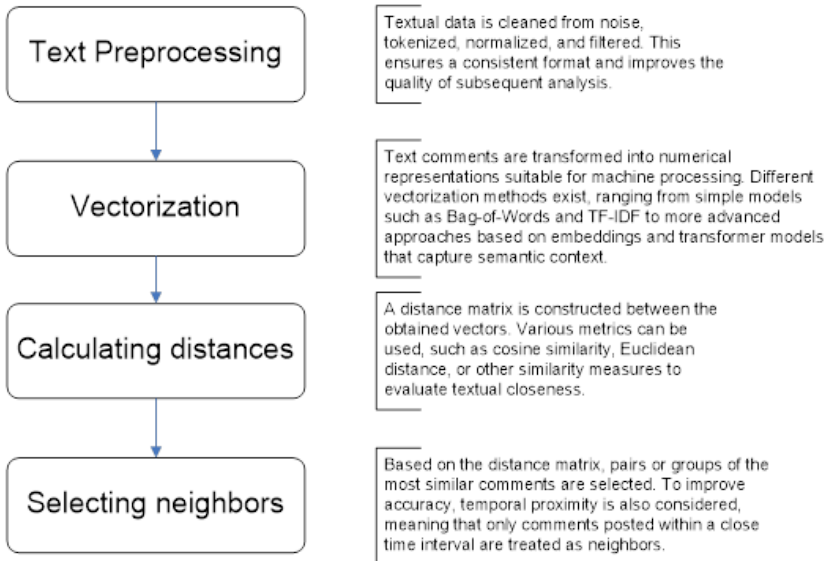


Fig. 2.3. Key steps for finding similar comments

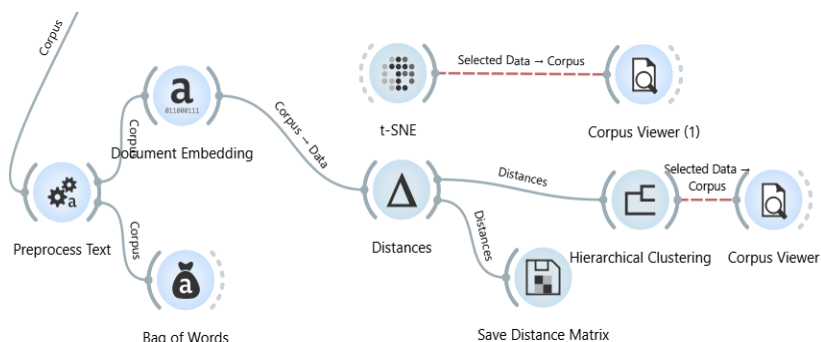


Fig. 2.4. Orange workflow for finding similar comments

the text into separate words or stable word combinations. Since Russian words are on average longer than English words, we use the Regexp pattern: `\w{4,}`. That is, for further analysis the text keeps sequences consisting of four or more word characters in a row. Then, tokens are normalized: they are reduced to a standard form via lemmatization with the UDPipe Lemmatizer for Russian. This smooths differences between word forms. With the additional filtering, stop-words and other irrelevant elements are removed. As a result, the textual data obtains a uniform form, which significantly increases the reliability of further calculations.

Stage 2. Vectorization

The next step is converting textual data into a numeric representation. In this study, two vectorization methods were tested. The Bag-of-Words method is based on counting word frequencies. Its advantages are simplicity of implementation, fast performance, sensitivity to exact lexical matches in comment composition, and good interpretability. Its limitation is that it ignores context and text structure, because the same set of words in a different order may produce the same vector.

The second method is Document Embedding, implemented using the multilingual SBERT (Sentence-BERT) model, which allows building vector representations of texts in different languages, including Russian. Unlike Bag-of-Words, this method captures semantic relations between words and can detect semantic similarity even when the wording differs. The model outputs a vector of 384 numeric features, and this number is independent of the length of the original text. The strong side of

SBERT is context handling and high accuracy in detecting paraphrases, but this method requires higher computing resources and depends on the quality of the pretrained model. In addition, the interpretability of such a model is much worse than that of a Bag-of-Words model.

In the Orange workflow shown in Fig. 2.4, the choice of one of these methods is made by creating a connection between its output and the Distances widget's input. Further comparison of the two approaches allows evaluation of which is better suited to the task under the specific data conditions.

Stage 3. *Calculating distances*

After obtaining vector representations of the comments, a distance matrix is calculated. As a metric, the study uses the normalized Euclidean distance, enabling the estimation of the degree of difference between vectors on a unified scale. Normalization is needed so that data with different value ranges can be compared correctly. The result of this stage is a symmetric matrix in which each cell reflects the degree of proximity of a pair of comments. The smaller the distance, the higher the probability that the comments duplicate or paraphrase each other. Since Orange Data Mining does not contain tools for filtering data from a Distance Matrix, this matrix is saved as an Excel file for further processing by an external program script.

In addition to this traditional approach, we considered an alternative method that projects data into a two-dimensional space using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm. In the study, we use t-SNE only for vectors obtained by the Document Embedding method. t-SNE enables visualization of multidimensional embeddings and their arrangement on a plane so that semantically similar comments are located near each other (Policar & Zupan, 2024; Godec et al., 2021). This method does not replace distance-matrix calculation but serves as an auxiliary tool: it simplifies the detection of dense clusters of similar texts, reveals hidden patterns, and can serve as a check of the correctness of the chosen metric.

Stage 4. *Selecting neighbors*

The final stage involves finding sets of comments that can be treated as identical or strongly similar. For this purpose, we use two groups of methods. The first group includes *visual* methods for message clustering, which allow a clear assessment of whether clusters of similar comments exist in the data and enable identification of main discussion topics. The

second group consists of *formal* clustering methods, which are better suited for use in an automated manipulation detection system.

In this study, we use such visual clustering methods as hierarchical clustering and t-SNE. Hierarchical clustering is less resource-intensive and can effectively find texts that contain similar or identical fragments. However, its visual effectiveness decreases as the corpus grows. t-SNE is more convenient for large text corpora, but it is also more resource-intensive. Building a two-dimensional mapping of vectors from the analyzed dataset after Document Embedding takes 10–15 minutes on the author's computer, whereas hierarchical clustering is performed in seconds.

Although visual cluster representations are convenient for a preliminary human assessment of data and allow identifying the presence and nature of coordinated activity within a specific dataset, their use becomes impractical for automatic batch analysis due to high time costs and high requirements for the analyst's qualifications. In addition, the methods considered above do not provide sufficient flexibility in configuring the clustering process, resulting in clusters that become "contaminated" with data unrelated to coordinated activity. To determine whether these limitations can be addressed, we formulated, as an alternative, a formal approach to message clustering based on graph structure discovery in the data (Fig. 2.5).

Let us consider the main components of this approach.

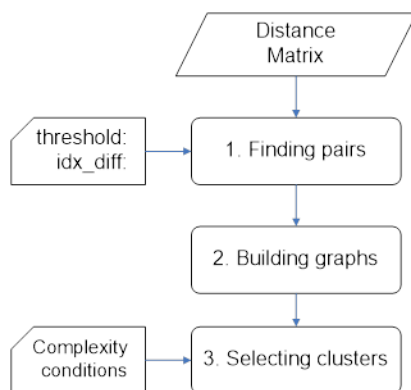


Fig. 2.5. Workflow for clustering based on finding graph structures in data

1. Finding pairs.

At this stage, a special algorithm iterates over the distance matrix and selects pairs of comments with a small vector distance. In other words, we search for comments that are almost identical in content. To control this process, an external parameter threshold is used to set a limit on the maximum distance between vectors. The larger the value of this parameter, the more pairs of comments will be found and the stronger they may differ from each other.

A preliminary analysis of the distance-matrix processing results showed that pairs sometimes include comments separated by more than several months, which contradicts the features of coordinated activity identified above. Therefore, we added an additional condition requiring comments to be published in close context. For this purpose, an additional parameter, `idx_diff`, was introduced into the algorithm, which sets a limit on the maximum distance between comments in a pair. The term “distance” in this case describes not the difference between vector values, but the difference between comment publications, which is expressed either in time units or as a maximum allowed difference between comment indices. For example, `idx_diff=50` specifies that the components in a pair must be no more than 50 messages apart. This criterion allows filtering out random coincidences and focusing on cases where similar comments appear in the same discussion within a limited time interval.

2. Building graphs.

The result of searching for pairs of neighboring messages is a table of the following form:

```
Row: 3, Column: 2, Distance: 0.00  
Row: 21, Column: 19, Distance: 0.00  
Row: 30, Column: 8, Distance: 4.31  
Row: 41, Column: 19, Distance: 1.13  
Row: 41, Column: 21, Distance: 1.13
```

Such a record indicates that the comments in the third row and the second column have identical meanings. Comments in row 21 and column 19 are also identical. The comment in row 41 is very similar to the comments in columns 19 and 21, and so on.

However, for the purposes of this study, it is necessary to identify not pairs but groups of similar messages, that is, clusters. For example, if message a) is similar to message b), and message b) is similar

to message c), then they form the cluster “abc”. In the example above, the messages form the following clusters:

Cluster 1: 3,2

Cluster 2: 21,19,41

Cluster 3: 30,8

Clusters of two messages are most often caused by technical errors, for example, accidental duplication of a comment. However, clusters that unite three or more comments are already an anomaly. Therefore, the larger the cluster size (the number of messages in it), the higher the probability of coordinated activity.

To form clusters based on a set of message pairs, we use tools from graph theory. Indeed, if we treat message pairs as graph edges, then connecting such pairs into clusters is equivalent to a graph traversal. During clustering, it is necessary to find all graphs that are not connected to each other and can be built from the set of message pairs. This process is illustrated in Fig. 2.6.

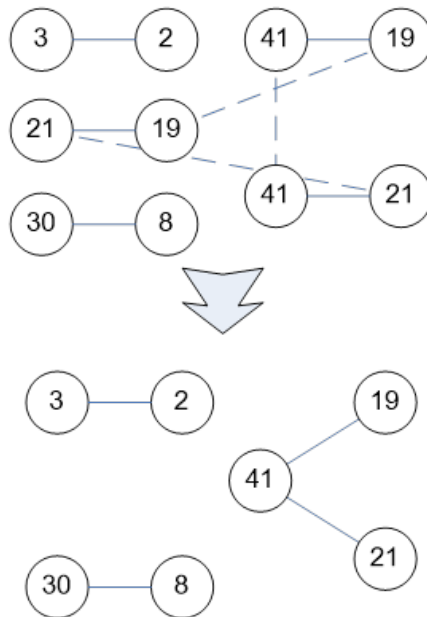


Fig. 2.6. Message clustering by finding graph structures in the data

To find graph structures in data, we use the Depth-first search algorithm (Tarjan, 1972). This algorithm traverses all vertices in the data space, searching to the maximum possible depth before moving to the next vertex and, accordingly, the next graph. This guarantees correct cluster formation of any complexity.

3. As an output of the previous stage, we obtain a set of clusters composed of message pairs. Different clusters may contain different numbers of messages. At the same time, situations occur quite often where comments are duplicated due to technical failures or accidental reasons. Obviously, if two completely identical comments are found in a sample of several thousand comments, this hardly indicates large-scale manipulations. Therefore, before further analysis, it is reasonable to limit cluster complexity. At a minimum, clusters formed by accidental message duplication should be excluded. For this purpose, an additional complexity-limited condition is introduced into the algorithm. The condition serves as a lower bound; that is, graphs with complexity less than or equal to the specified value are filtered out. Complexity is measured as the number of vertices included in the graph (cluster). Thus, if it is necessary to remove from further analysis clusters formed by accidental message duplication, the parameter $\text{complexity}=2$ is set.

2.3. Methods for identifying low-level manipulations based on reply relevance anomalies

So far, we have treated comments as independent semantic units. However, in reality, comments reflect interaction between people. On YouTube, users can comment on the video itself or reply to other users' comments. In communication between ordinary users, their messages are linked to each other or to the content they comment on. Studies show that bots often post irrelevant comments (Marcondes et al., 2024). This feature is not "strong" enough to classify individual comments, but in statistical analysis, a substantial deviation from baseline values in the average relevance of a reply comment to its parent comment can be used as an indicator of bot activity. Below, we formulate approaches to manipulation detection based on this type of anomaly.

In the dataset structure used earlier (Table 2.1), each row contains the text of a user comment and a reference to the parent content. For the

research, it must be transformed so that the parent content is present in each row, along with the comment text. Analysis of the structure of Table 2.1 shows that the hierarchy of YouTube comments is formed using the fields *parentid* and *id*. At first glance, this approach allows representing a comment structure of any nesting level, because the *id* of one comment can serve as *parentid* for another. However, as it turned out, YouTube stores hierarchy information only up to the second level. Dataset analysis enabled identification of the principles for forming *parentid* and *id* for comments at different levels (Fig. 2.7).

All comments that are replies to second-level comments are technically also shown in the dataset as second-level comments, meaning that their *parentid* matches the *parentid* of the parent comment.

This approach may be justified from YouTube's perspective because it improves the interface's usability. However, in our study, this creates difficulties in correctly identifying "parent text – reply" pairs.

To improve the accuracy of identifying such pairs, we utilize an additional feature of parent comments on YouTube: the text of comments at the third and subsequent levels (fields *textDisplay* and *textOriginal*) begins with @@, followed by the username to whom the comment is addressed. By comparing this name with the values in the *authorDisplayName* field of previous comments in the dataset, we can determine the parent comment with a fairly high probability. We only mention this as a fairly high probability because situations where a user changes the

Level 0: core video

↓
1st level comments
(*id* = *parentid*)

↓
2nd level comments
(*id*=*parentid*.2ndlev_*id*)

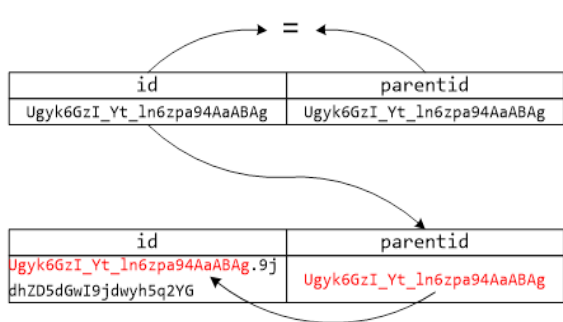


Fig. 2.7. Forming the *parentid* and *id* fields for YouTube comments of different levels

username or posts several comments with different content one after another cannot be tracked.

Thus, the general algorithm for determining the parent comment (which is added to the dataset in the `parent_text` field) can be formulated as follows:

1. If `parentid = id`, this is a 1st level comment. As `parent_text` we set the video description.
2. If `id` contains a dot, this is a reply, or a 2nd level comment. If `textOriginal` does not start with `@@` symbols, then the “parent” comment can be determined precisely by the `parentid` field, which references the root id of the top-level comment of the target branch. That is, to find the parent text in this case, we need to:
 - take the `parentid` value,
 - find in the dataset the comment with exactly the same `id`,
 - take its `textOriginal` – this will be `parent_text`.
3. If the comment text (`textOriginal`) starts with `@@` symbols, then this is a reply not to a 1st level comment but to another reply of the second or next levels. In this case, `parentid` still points to the “root” of the branch. To determine the most probable parent, we use a heuristic:
 - determine the username being replied to, which comes after `@@` in the `textOriginal` field, for example, `@@user1703`;
 - scan the dataset sequentially from the current comment to older ones until a user with the same name is found in the `authorDisplayName` field;
 - set this nearest comment as `parent_text`.

After forming the dataset, it is pseudonymized in the same way as described earlier in subsection 2.1, by replacing usernames with anonymous names in the format user N.

Next, we move to the analysis of connectedness between comments and replies. Research results (Al Tawil et al., 2024) show that traditional NLP methods are weaker in detecting semantic nuances than more advanced neural tools. At the same time, the complexity of analyzing the connectedness of “question–answer” message pairs is higher than that for sentiment analysis considered in the previous section. Therefore, we use only models based on transformer neural networks below.

Model selection

We formulate the following assumptions for model selection:

1. Correct work with the Russian language, because it is the main language in the studied dataset
2. Different complexity and speed, because it is necessary not only to obtain a result but also to find the optimal way to obtain it
3. Ability to operate in an automatic mode, in accordance with the program
4. Ensuring reproducibility of results, which implies the use of publicly available and, if possible, free models.

These conditions are satisfied by several transformer-based neural networks, including two main architectures: BERT (Devlin et al., 2019) and GPT (Vaswani et al., 2017).

Within the BERT architecture, we selected the following models:

Rubert-tiny2 (Cointegrated, n.d.). This is one of the simplest and fastest models using the BERT architecture. This result was achieved by distilling several “large” models, including BERT-Multilingual, RuBERT, LaBSE, and others. The model uses **29.4M** parameters and outputs an embedding vector of 312 parameters. Rubert-tiny2 does not contain an additional layer that specializes the model for a specific task, so it is a baseline solution. For example, in our case, an additional layer is needed to obtain conclusions about the probability of matching “parent text – reply.” One of the solution options used in this study is calculating cosine similarity.

GrigoryT22/cross-encoder-ru + LaBSE-en-ru (GrigoryT22, n.d.). This choice is explained by the following factors. First, this model uses more parameters than Rubert-tiny2 and therefore can potentially solve more complex tasks. Second, the model includes a classifier trained on “question–answer” pairs, which matches the tasks of our research. The base of the cross-encoder is a modified LaBSE-en-ru model. The total number of parameters of the model and the classifier is **132M**.

Within the GPT architecture, we selected Google’s Gemma3 model family. The main selection criterion was that these models be free to use, run locally on computers, and have relatively low computational requirements. The following models were used:

Gemma3:1b (Google, 2025-a). At the time of the study, this was the most lightweight model in the Gemma family. The model has **1B** param-

eters, which is small by GPT standards, but almost 8 times larger than GrigoryT22/cross-encoder-ru. Since this model belongs to the generative pre-trained transformer (GPT) architecture, working with it differs from that of BERT-like models in how data is provided and results are obtained. The model size allows installing it locally using the Ollama platform and calling it from Python code without using the Internet.

Gemma3:27b (Google, 2025-b). This is the most powerful model in the Gemma family. Although, by the number of parameters (27B), it is significantly (more than 20 times) smaller than full-size models such as Gemini, ChatGPT, Grok, test results show that the quality of Gemma3:27b's outputs is comparable to GPT3.5, which once formed the basis of the first version of ChatGPT. This model can also be deployed locally, but it requires much more computational resources than the previous one. An acceptable option is to use a "cloud" model with API access. In this case, it should be considered that the "cloud" model has limits on the number of free requests – no more than 30 requests per minute (RPM) and no more than 14400 requests per day (RPD) (Google, n.d.).

Let us consider the features of each of these models for detecting relevance anomalies.

Rubert-tiny2 is the simplest model among those considered in terms of parameters. The model cannot determine the relevance of a reply to a parent comment by itself, so we estimate this value approximately by calculating the cosine similarity of the embeddings of the parent comment and the reply using the formula:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}} \quad (2.1)$$

This approach somewhat contradicts the established practice of using BERT-like models, which involves processing the obtained embeddings with a cross-encoder trained on a labeled dataset. However, in our study, this *bi-encoder-cross-encoder* combination is already represented by the next model, while Rubert-tiny2 is used "as is" to assess the feasibility and potential of this approach.

The GrigoryT22/cross-encoder-ru model is also based on the BERT architecture, but it includes not only a bi-encoder but also a cross-

encoder trained on a Russian-language dataset for recognizing “question–answer” pairs. This is quite close to our task of determining the relevance of comments. The model takes a pair of text strings as input and returns a pair of logit values, for example, $[-3.3953, 3.1857]$. They can be interpreted as the “confidence level” of a binary classification model for the two possible outcomes: [0] or [1], respectively.

To convert logit values into probability estimates, we use the *softmax* transformation (2.2):

$$p_1 = \frac{e^{z_1}}{e^{z_0} + e^{z_1}} = \frac{1}{1 + e^{(z_0 - z_1)}}, \quad (2.2)$$

Gemma3:1b uses the General Pretrained Transformer (GPT) architecture, so the work follows a “prompt–answer” mode. Therefore, to use the model, two tasks must be completed: preparing a prompt with instructions and integrating input-file data into those instructions, with the results saved.

Gemma3:27b requires significant resources for local deployment. If these requirements exceed the available computer configuration, it can be run remotely via Google AI Studio. The model prompt, to ensure comparability of results, should correspond to the prompt for Gemma3:1b.

After processing the data with various models, it is important to determine which one performs better at identifying irrelevant comments. Since the dataset was not labeled, we lack prior information about correct and incorrect model outputs and therefore cannot use traditional methods for evaluating classification results. Instead, we aim to identify the best models indirectly.

We assume that manipulative activity varies in intensity across comments of different levels. This hypothesis is supported by the differences in accessibility among comment levels: creating first- and second-level comments is easier than producing third-level comments. It also seems evident that, to reduce costs and simplify bot operations, manipulative activity will focus on specific comment levels. Therefore, we need to analyze the relevance scores generated by various models and select those models that show statistically significant differences across comment groups for further examination. Subsequently, we will also conduct statistical tests to identify the presence of temporal relevance anomalies and assess their significance.

Thus, the overall research scheme can be presented as follows (Fig. 2.8).

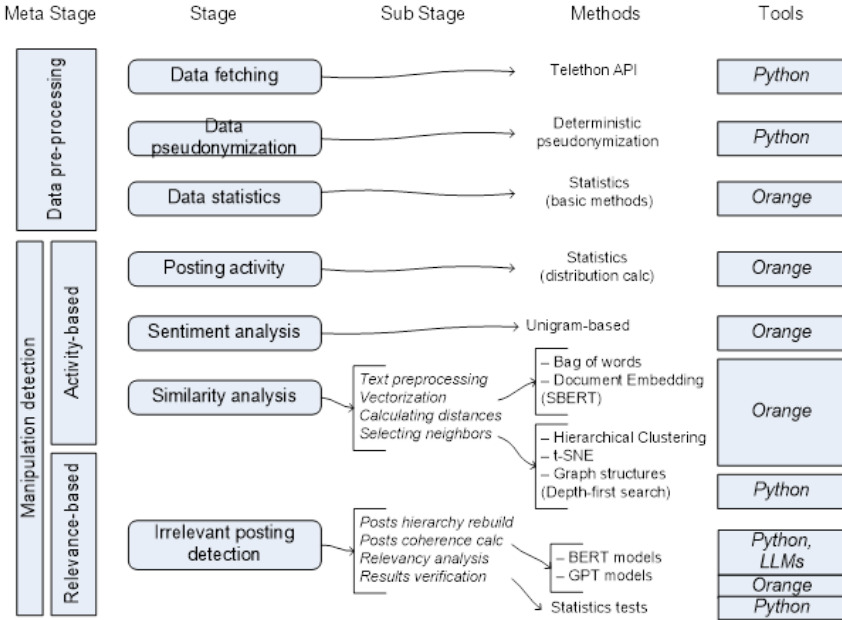


Fig. 2.8. Research design

The scheme in Fig. 2.8 allows comparison of the methods used in the study and the tools for their implementation with the main and sub-stages of the research. Next, we consider the obtained results.

3. Results

3.1. Data preprocessing

As a testing object for the research methodology, we chose comments on a YouTube video (Sunvideo, 2022). The original video features neutral visuals of Mariupol, with no voice-over narration or provocative inserts of any kind. It was uploaded to YouTube about two months before the start of the Russian military invasion, which makes it possible to compare comment activity across different time periods. At the time of data collection, it had a total of 345,000 views.

To download the comments, a Python program (v. 3.10.0) was written in the Jupyter Notebook environment using the YouTube API. The program code is provided in Appendix 2. As a result of fetching, a local copy of the data was created, containing 3,700 comments posted from 31.12.2021 to 20.08.2025. The absolute majority of comments were written in Russian.

To pseudonymize usernames, a separate Python program was also created (Appendix 3). It reads the source CSV file, finds the `authorDisplayName` column, and replaces each unique username with a pseudonym in the user-N format, where N is an ordinal number. The script uses a Python dictionary to map original names to pseudonyms. This ensures that the same real username is always replaced by the same pseudonym (user-1, user-2, ...) throughout the file. The script automatically finds the index of the `authorDisplayName` column by its header, making it robust to changes in column order in the file. The result is saved to a new .csv file, while the original file remains unchanged. As a result, 1,735 unique users were pseudonymized.

3.2. Identification of low-level manipulations on YouTube based on activity anomaly analysis

Posting activity

The analysis of user activity was preceded by the discretization of the data by the `publishedAt` field with a 1-month period (see the work-

flow in Fig. 2.1). The results of data visualization using a distribution histogram are shown in Fig. 3.1

An analysis of Fig. 3.1 shows that user activity in commenting on the video is extremely unevenly distributed. A strong spike is observed in December 2022, which accounts for 92.4% of the total number of comments over the entire period. Obviously, this spike is related to YouTube’s algorithms, which internally promote the video. Nevertheless, even for this platform, such a large number of comments (compared to the rest of the period) is anomalous. It can be assumed that this period is key to detecting coordinated media activity.

Sentiment analysis

We visualize the sentiment analysis results using a scatter plot diagram (Fig. 3.2)

The diagram shows that the emotional tone of comments changed substantially over time. In the initial period after the video was posted on YouTube, most comments are neutral or positive. However, in December 2022 strong negativity appears in comments, which in time coincides with the period of the highest user activity.

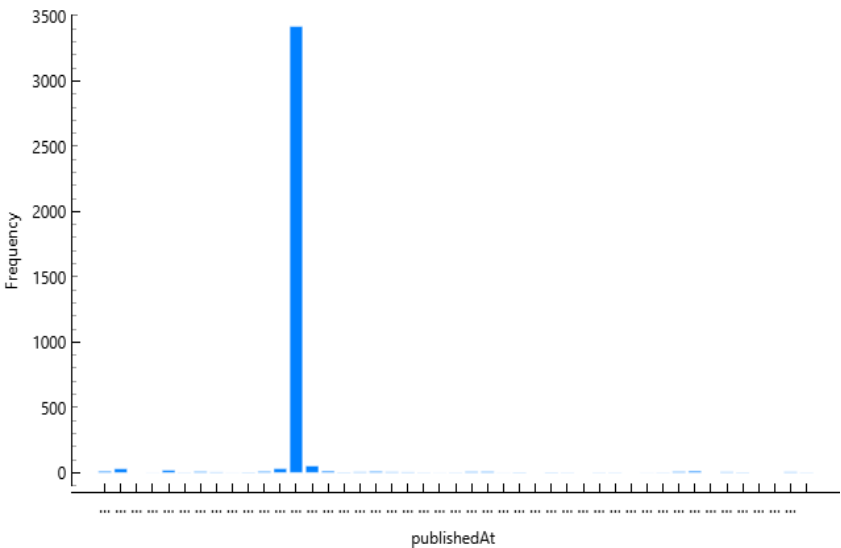


Fig. 3.1. Dynamics of user activity by months

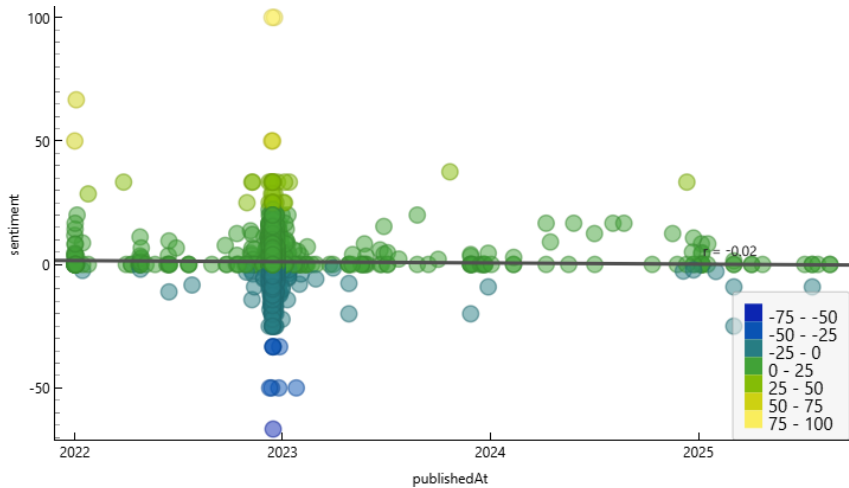


Fig. 3.2. Scatter plot of sentiment vs published time

The diagram in Fig. 3.2 allows assessment of the range of emotionality across different time periods. To calculate averaged monthly sentiment values, the lower branch of the Sentiment analysis workflow is used (Fig. 2.2). The “Discretize” widget converts the `publishedAt` variable from a continuous to a categorical type with a Time interval of 1 month. The next widget groups data by the `publishedAt` variable and aggregates columns with the “mean” aggregation. As a result of these transformations, we obtain a table containing 38 rows (by the number of months in the analyzed period in which at least one comment was posted), and each row contains the mean sentiment value. The last widget, “Bar Plot,” visualizes the calculation results as a diagram (Fig. 3.3).

The diagram shows that the emotional tone of comments changed substantially over time. In the initial period after the video was posted on YouTube, most comments are neutral or positive. However, in December 2022 strong negativity appears in comments, which in time coincides with the period of the highest user activity.

The diagram in Fig. 3.2 allows assessment of the range of emotionality across different time periods. To calculate averaged monthly sentiment values, the lower branch of the Sentiment analysis workflow is used (Fig. 2.2). The “Discretize” widget converts the `publishedAt` variable

from a continuous to a categorical type with a Time interval of 1 month. The next widget groups data by the `publishedAt` variable and aggregates columns with the “mean” aggregation. As a result of these transformations, we obtain a table containing 38 rows (by the number of months in the analyzed period in which at least one comment was posted), and each row contains the mean sentiment value. The last widget, “Bar Plot,” visualizes the calculation results as a diagram (Fig. 3.3).

An analysis of Fig. 3.3 shows that the mean sentiment level changed quite substantially. At the beginning of the observed period, that is, immediately after the video was posted, comments with a moderately positive emotional tone dominated. Then the emotional level reached its maximum for the entire observed period. In time, this corresponds to the first month after the start of the full-scale military invasion. Comments in this period mostly contain words of support and indeed have a positive emotional tone. However, later polarization of views increases in user discussions, more negative comments appear, and as a result, the mean sentiment level falls. In the period of the highest commenter activity (December 2022), the mean sentiment level, despite the maximum amplitude, is only slightly above 0. In later periods, the mean sentiment level increases, though it remains unstable.

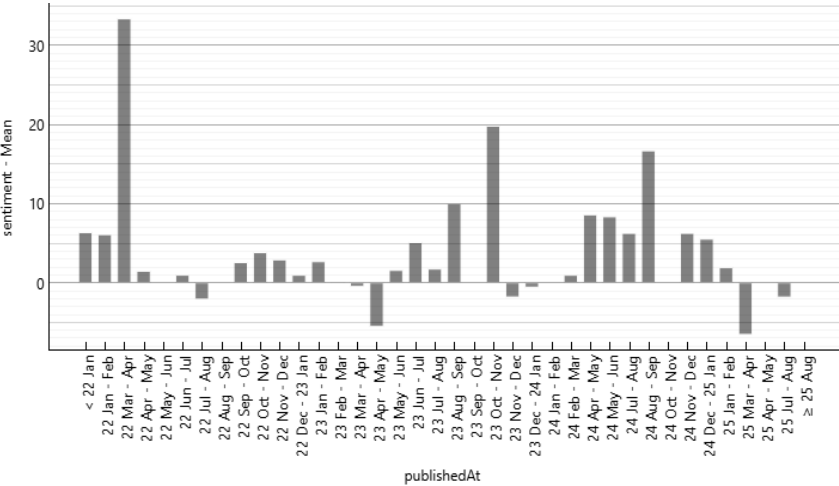


Fig. 3.3. Bar diagram of mean sentiment level during the observed period

However, this increase shown in Fig. 3.3 coincides with a decrease in commenter activity. In some months, no new comments appeared at all, or only a few were added, which affects the statistical significance of the monthly mean values.

Sentiment analysis showed that the period of the highest user activity (December 2022) is also characterized by the largest amplitude of emotional scores in comments. According to Ellaky & Benabbou (2024), this is one of the indicators of bot activity, especially given that the mean sentiment level in this period is only slightly above 0. Nevertheless, this indicator can be treated only indirectly. The process of obtaining more reliable evidence of information manipulations is considered below.

Similarity analysis

The Orange workflow (Fig. 2.4) provides several options for performing the key steps for finding similar comments:

- for vectorization: Bag-of-Words and Document Embeddings (using the SBERT model);
- for clustering (selecting neighbors): hierarchical clustering, t-SNE, and graph structures.

In total, these options can be combined in 6 ways. During the study, all combinations were tested, but below we consider in detail only those that showed sufficiently good results. The others are mentioned briefly.

First, we consider the results of applying visual methods for message clustering (hierarchical clustering and t-SNE).

The hierarchical clustering method based on Bag-of-Words vectors did not yield very good results. Fig. 3.4 shows that this method assigns a high weight to the nickname of the user to whom the reply is addressed (as shown in Subsection 2.3, such nicknames have remained in the `textOriginal` field due to the specifics of the YouTube API).

Nevertheless, even in the fragment of the full hierarchy tree shown in Fig. 3.4, the presence of rather large clusters with similar comments is visible, which can be treated as a sign of bot activity.

Fig. 3.5 shows hierarchical clustering based on Euclidean distances between vectors obtained after document embeddings. The hierarchy depth is 10 levels. Among the available ways to define the distance between clusters (Single linkage, Average linkage, Weighted linkage, Complete linkage, Ward linkage), after a series of experiments, Ward linkage was chosen, which computes the increase of the error sum of

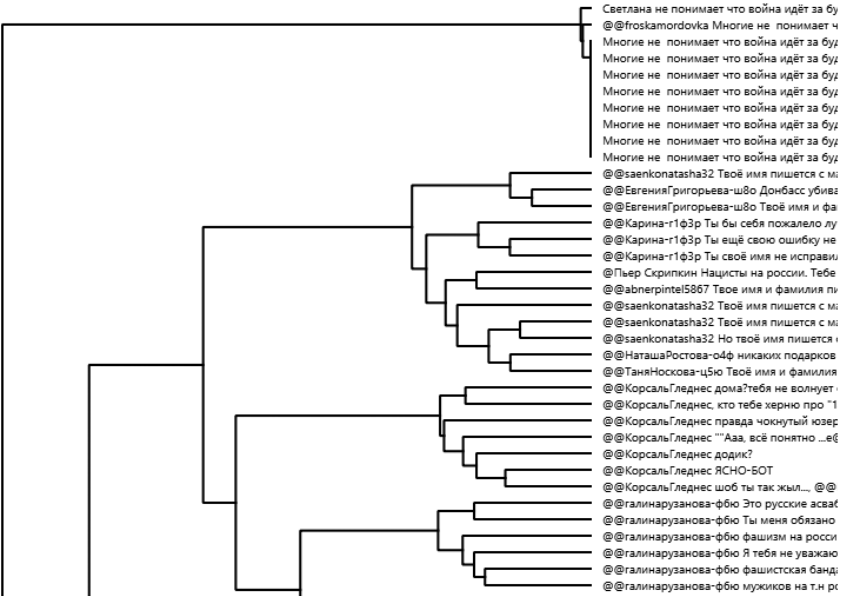


Fig. 3.4. Part of the full comments hierarchy after the sequence “Bag of Words → Hierarchical clustering”

squares. In other words, Ward’s minimum variance criterion minimizes the total within-cluster variance (Orange Data Mining, n.d.).

Fig. 3.5 also illustrates the limitations of using hierarchical clustering for analyzing large datasets. With Ward linkage, the full hierarchy diagram becomes difficult to analyze due to its large size. If other ways of defining inter-cluster distance are used, noise increases because the algorithm begins to place messages that differ quite strongly in the same clusters.

Fig. 3.6 shows the content of the selected cluster.

67% of messages in cluster 1 were posted by a single author using the pseudonym: user-339. The remaining messages were later posted by other authors. Although these messages are very similar in formal features, their content does not classify them as manipulative due to the lack of a clear manipulation goal. Instead, this cluster exemplifies the phenomenon of self-organization in online communities (Polupanova et al., 2025). This example highlights the importance of additional analysis

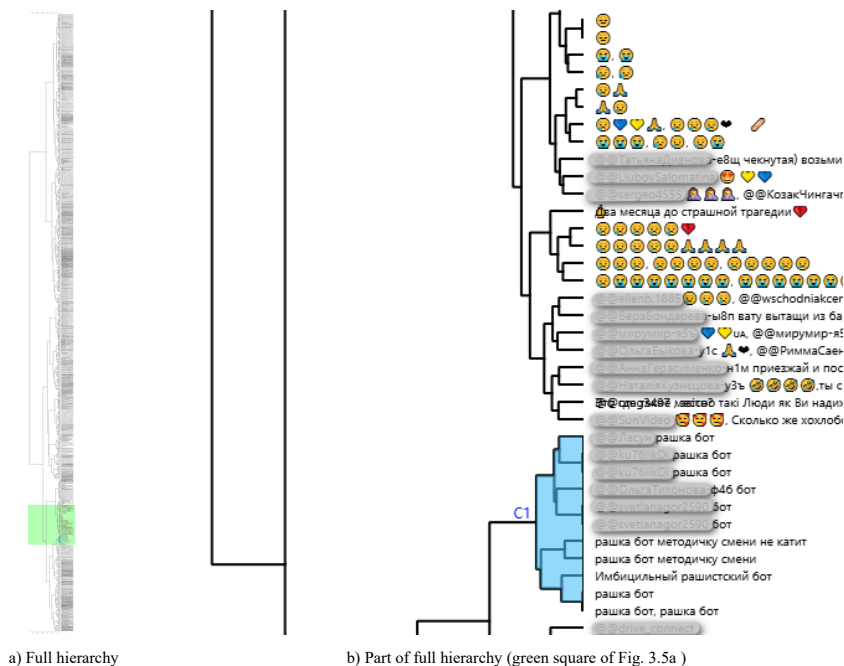


Fig. 3.5. Results of comment clustering after the sequence Document Embedding $\rightarrow$ Hierarchical clustering.

of the results and underscores the difficulty of developing fully autonomous methods to combat manipulation and coordinated activity.

A careful analysis of the cluster tree also reveals groups of comments that convey narratives of official propaganda. One example is shown in Fig. 3.7.

Visual search in a branched hierarchical cluster tree is a very labor-intensive task. For a more general visual analysis of the data, we use their two-dimensional representation after t-SNE processing of Document Embedding (Fig. 3.8).

In Fig. 3.8, several clusters of different sizes are clearly visible. Cluster analysis showed that they indeed contain messages with similar meaning. Moreover, the multilingual SBERT model used here for text vectorization can detect semantically similar messages even when written in different languages, and even when meaning is expressed through

1	user-339 (1)	textOriginal: [REDACTED] рашка бот
2	user-339 (2)	authorDisplayName: user-339
3	user-339 (3)	Cluster: C1
4	user-339 (4)	textOriginal: рашка бот
5	user-339 (5)	authorDisplayName: user-339
6	user-339 (6)	Cluster: C1
7	user-339 (7)	textOriginal: Имбицильный рашистский бот
8	user-339 (8)	authorDisplayName: user-1219
9	user-1219 (1)	Cluster: C1
10	user-1375 (18)	textOriginal: [REDACTED] ф46 бот
11	user-1566 (1)	authorDisplayName: user-1375
12	user-1566 (2)	Cluster: C1
		textOriginal: [REDACTED] бот
		authorDisplayName: user-1566
		Cluster: C1
		textOriginal: [REDACTED] бот
		authorDisplayName: user-1566
		Cluster: C1

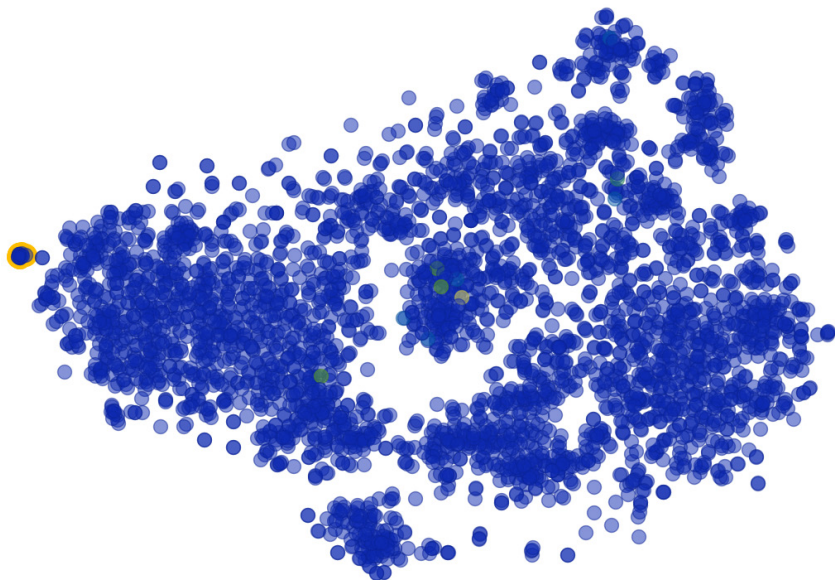
Fig. 3.6. Content of cluster 1

1	user-1343 (1)	textOriginal: Этот город построили коммунисты до развала СССР 19 августа 1991 года.
2	user-1343 (2)	authorDisplayName: user-1343
3	user-1343 (3)	textOriginal: Этот город построили коммунисты до развала СССР 19августа 1991 года.
		authorDisplayName: user-1343
		textOriginal: Этот город построили коммунисты до развала СССР 19 августа 1991 года.
		authorDisplayName: user-1343

Fig. 3.7. Content of cluster 2

emojis. This is illustrated in Fig. 3.9, which presents an example of the content of a medium-sized cluster where messages express regret about what happened.

Let’s examine one of the small clusters marked with an orange border on the left side of Fig. 3.8. As the content analysis of this cluster reveals, it includes messages that can be classified as manipulative, specifically



*Fig. 3.8. Results of comment clustering after the sequence
Document Embedding $\rightarrow$ t-SNE*

messages aimed at drawing attention to a particular religious movement (Fig. 3.10).

Other clusters can be found in a similar way. However, visual analysis is very labor-intensive, does not guarantee the discovery of all clusters, and can therefore be used only for a preliminary assessment of the level of anomalous activity.

Next, we consider the results of implementing the approach described in Section 2.2, which is based on analyzing the distance matrix and searching for clusters within it as graph structures. This approach enables automatic detection of all message clusters and provides additional options for controlling the search process. We also compare its effectiveness with visual approaches.

As input data, we use the distance matrix calculated earlier in the Orange workflow (Fig. 2.4). We use two distance matrices created from the same dataset but with different vectorization methods. The distance matrix based on the Bag-of-Words method is called *bagw*.

[illegible]

Fig. 3.9. Content of cluster 3 from the two-dimensional t-SNE representation

1	user-927 (1)	textOriginal:	Светлана не понимает что война идёт за будущее Руси и для этого есть много примеров, напишите предсказания старцы Иоанн Одесского и ведение Марины его же. Богу не нужна эта ему надо чтобы люди молились, а не делали кровавый майдан как и октябрьская революция с разрешением храмов, и, там и там вышло в войну и все остановилось, даже пцу это не векова? и там нет спасения
2	user-927 (2)	authorDisplayName:	user-927
3	user-927 (3)	textOriginal:	Многие не понимают что война идёт за будущее Руси и для этого есть много примеров, напишите предсказания старцы Иоанн Одесского и ведение Марины его же. Богу не нужна эта ему надо чтобы люди молились, а не делали кровавый майдан как и октябрьская революция с разрешением храмов, и, там и там вышло в войну и все остановилось, даже пцу это не векова? и там нет спасения
4	user-927 (4)	authorDisplayName:	user-927
5	user-927 (5)	textOriginal:	Многие не понимает что война идёт за будущее Руси и для этого есть много примеров, напишите предсказания старцы Иоанн Одесского и ведение Марины его же. Богу не нужна эта ему надо чтобы люди молились, а не делали кровавый майдан как и октябрьская революция с разрешением храмов, и, там и там вышло в войну и все остановилось, даже пцу это не векова? и там нет спасения
6	user-927 (6)	authorDisplayName:	user-927
7	user-927 (7)	textOriginal:	Многие не понимает что война идёт за будущее Руси и для этого есть много примеров, напишите предсказания старцы Иоанн Одесского и ведение Марины его же. Богу не нужна эта ему надо чтобы люди молились, а не делали кровавый майдан как и октябрьская революция с разрешением храмов, и, там и там вышло в войну и все остановилось, даже пцу это не векова? и там нет спасения
8	user-927 (8)	authorDisplayName:	user-927
9	user-927 (9)	textOriginal:	Многие не понимает что война идёт за будущее Руси и для этого есть много примеров, напишите предсказания старцы Иоанн Одесского и ведение Марины его же. Богу не нужна эта ему надо чтобы люди молились, а не делали кровавый майдан как и октябрьская революция с разрешением храмов, и, там и там вышло в войну и все остановилось, даже пцу это не векова? и там нет спасения
10	user-927 (10)	authorDisplayName:	user-927
11	user-927 (11)	textOriginal:	Многие не понимает что война идёт за будущее Руси и для этого есть много примеров, напишите предсказания старцы Иоанн Одесского и ведение Марины его же. Богу не нужна эта ему надо чтобы люди молились, а не делали кровавый майдан как и октябрьская революция с разрешением храмов, и, там и там вышло в войну и все остановилось, даже пцу это не векова? и там нет спасения
12	user-927 (12)	authorDisplayName:	user-927

Fig. 3.10. Content of cluster 4 from the two-dimensional t-SNE representation

The distance matrix based on the Document Embedding method is called `embed`.

The algorithm for clustering comments by finding graph structures in the distance matrix is implemented as a custom Python script (Appendix 4).

The key statistical characteristics of the matrix contents are shown in Table 3.1.

Table 3.1
Key statistical characteristics of data from matrices `bagw` and `embed`

Statistics	<code>bagw</code>	<code>embed</code>
max	189,5914289	45,05860466
min	0	0
mean	30,64280818	19,27469679
standard deviation	14,14836893	3,548212226

The smaller the number describing the distance between two comments, the greater their similarity. To find pairs of comments that are close to each other, we set a `threshold` value for “similarity” and compare distances between comments with this value. If the distance is smaller than the threshold, the pair of comments is considered identical in meaning. To limit the time context of the search, we set a maximum value for the difference between comment indices (`idx_diff`).

During the experiments, we studied the number and composition of clusters found by the graph-structure approach under different values of the parameters `threshold` and `idx_diff`. The results of selecting pairs of identical-by-meaning comments and forming clusters for some of the tested combinations are shown in Tables 3.2 and 3.3 (the number of clusters with more than two elements is given in parentheses).

A comparative analysis of Tables 3.2 and 3.3 shows a substantial difference between the results obtained with different vectorization methods. When searching for pairwise matches in the distance matrix built on Bag of Words, increasing `Threshold` leads to a very rapid growth of the number of pairs (12 times when `idx_diff` = 50 and 17 times when `idx_diff` = 100). A sample analysis of these pairs showed that they are often quite distant in meaning. In addition, the Bag-of-Words method

Table 3.2

Selected comments pairs depending on Threshold, while `idx_diff=50`

Threshold	bagw		embed	
	pairs	clusters	pairs	clusters
1	180	73(14)	104	56(3)
5	321	75(18)	136	63(5)
7.5	754	73(16)	195	79(14)
10	2231	63(14)	556	153(45)

Table 3.3

**Selected comments pairs depending on Threshold,
while `idx_diff=100`**

Threshold	bagw		embed	
	pairs	clusters	pairs	clusters
1	246	70(12)	104	56(3)
5	544	64(8)	141	68(5)
7.5	1332	62(5)	217	87(19)
10	4154	54(6)	828	148(39)

fails to capture the semantic meaning of emojis, leading to classification errors in messages where emojis play a key role.

Substantial differences should also be noted between clustering results based on the `bagw` and `embed` matrices. In the first case, when `Threshold` increases, the number of clusters remains almost unchanged (when `idx_diff = 50`) or even decreases (when `idx_diff = 100`), despite a substantial increase in the number of found comment pairs. In clustering based on the `embed` matrix, increasing the threshold leads to an increase in the number of identified clusters (including clusters that contain more than two comments).

Analysis of the experimental results showed that message vectorization using Document Embedding ultimately provides higher-quality clustering.

Using the data from Tables 3.2 and 3.3, we compare clustering results for the `bagw` matrix (`Threshold = 5`, `idx_diff = 100`) and the `embed` matrix (`Threshold = 10`, `idx_diff = 50`). In both cases, approximately the same number of message pairs is selected (544 and 556, respectively).

However, their clustering results differ substantially. In the `bagw` matrix, 64 clusters are identified, of which 8 are “large” (containing more than two comments). In the `embed` matrix, 153 clusters are identified, of which 45 are “large.” At the same time, the maximum size of clusters identified in the `bagw` matrix is much larger than in the `embed` matrix. The cluster composition is provided in Appendix 5.

We compare the clustering quality of the `bagw` and `embed` matrices using the first “large” cluster as an example. The clustering algorithm parameters and the cluster composition are as follows:

- 1) Matrix `bagw` (`Threshold=5`, `idx_diff=100`)
Cluster 2: [19, 21, 34, 41]
- 2) Matrix `embed` (`Threshold=10`, `idx_diff=100`)
Cluster 2: [8, 9, **19, 21**, 23, 28, 30, **34, 41**]

When clustering the `bagw` matrix, we had to limit the `Threshold` to 5 because increasing it to the next value of 7.5 increased the size of this cluster from 4 to 243 messages. At the same time, the semantic content of these comments differed substantially, and what they had in common was only a small length. Most likely, this effect is caused by the limited precision of the Bag-of-Words vectorization method, which is unable to capture text meaning.

Let us consider the composition of these clusters. The full list of comments included in them is shown in Fig. 3.4–3.5. To avoid confusion with earlier clusters identified using graphical methods, we continue the continuous numbering within the study.

18	user-2 (6)	textOriginal:	Спасибо большое!)))	
19	user-2 (7)	authorDisplayName:	user-2	19
20	user-2 (8)	textOriginal:	Спасибо большое!)))	
21	user-2 (9)	authorDisplayName:	user-2	21
22	user-2 (10)	textOriginal:	Спасибо большое!!!)))	
23	user-2 (11)	authorDisplayName:	user-2	34
24	user-14	textOriginal:	Вот спасибо большое!)))	
25	user-2 (12)	authorDisplayName:	user-2	41

Fig. 3.4. Content of cluster 5 identified in the `bagw` matrix

19	user-2 (7)	textOriginal: Спасибо большое за отзыв! С наступающим!	8
20	user-2 (8)	authorDisplayName: user-2	
21	user-2 (9)	textOriginal: Благодарю)))	9
22	user-2 (10)	authorDisplayName: user-2	
23	user-2 (11)	textOriginal: Спасибо большое!)))	19
24	user-14	authorDisplayName: user-2	
25	user-2 (12)	textOriginal: Спасибо большое!)))	21
26	user-15	authorDisplayName: user-2	
27	user-16	textOriginal: Спасибо большое!))) За ссылкой написал в ФБ)	23
28	user-2 (13)	authorDisplayName: user-2	
29	user-17 (1)	textOriginal: Благодарю за отзыв!	28
30	user-2 (14)	authorDisplayName: user-2	
31	user-17 (2)	textOriginal: Спасибо большое за отзыв!)))	30
32	user-18	authorDisplayName: user-2	
33	user-19	textOriginal: Спасибо большое!!!)))	34
34	user-2 (15)	authorDisplayName: user-2	
		textOriginal: Вот спасибо большое!)))	41
		authorDisplayName: user-2	

Fig. 3.5. Content of cluster 5 identified in the embed matrix

As can be seen, cluster 5 identified in the bagw matrix contains only those comments that are almost identical in content and vocabulary. The cluster identified in the embed matrix contains comments united by a common meaning, but much more variable in wording. For example, comment 9 does not contain any common words with comments 8, 19, 21, 23, 30, 34, 41, but it is semantically similar to them, which was correctly captured by the Document Embedding method.

We note that cluster 5, like previously identified cluster 1 (Fig. 3.6), is not related to coordinated activity and contains replies from the video author to other users.

Let us consider how effectively graph-structure-based clustering can recover the clusters previously identified by visual methods. The results of searching for equivalents within clusters found by different methods are presented in Table 3.6.

An analysis of Table 3.6 shows that clustering based on the embed matrix produced better results than clustering based on the bagw matrix. All the same clusters as in the visual analysis were found (except cluster 3), as well as several additional clusters.

Table 3.6

Contents of clusters found by different methods

№	Visual clustering	Clustering by finding graph structures	
		Matrix embed	Matrix bagw
1	[564, 566, 567, 568, 569, 570, 572, 573, 2421, 2995, 3018, 3305, 3306]	Threshold =1: [564, 566, 573] Threshold =10: [564, 566, 568, 569, 573]	Threshold =1: [564, 566, 573, 609] Threshold ≥5: overclustering
2	[2741, 2742, 2744]	Threshold =5, 7.5, 10 [2741, 2742, 2744]	no such clusters
3	[383, 450, 971, 1012, 1018, 1077, 1142, ...]	no such clusters	no such clusters
4	[1692, 1695, 1697, 1698, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707]	Threshold =1: [1695, 1697, 1698, 1701, 1702, 1703, 1704, 1705, 1706, 1707] Threshold =5: [1692, 1695, 1697, 1698, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707]	Threshold =1: [1695, 1697, 1698, 1701, 1702, 1703, 1704, 1705, 1706, 1707] Threshold =10: [1692, 1695, 1697, 1698, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707]
5	no such clusters	Threshold=10: [8, 9, 19, 21, 23, 28, 30, 34, 41]	Threshold=5: [19, 21, 34, 41]
6	no such clusters	Threshold=7.5: [1777, 1778, 1780]	Threshold=10: [1777, 1778, 1780]
7	no such clusters	Threshold=7.5: [2225, 2227, 2262, 2263]	no such clusters
8	no such clusters	Threshold=7.5: [3059, 3065, 3077, 3078, 3080] Threshold=10: [3056, 3059, 3063, 3065, 3074, 3077, 3078, 3080]	Threshold=10: [3059, 3065, 3074, 3077, 3078, 3080]
9	no such clusters	Threshold=7.5: [3066, 3069, 3071, 3072, 3086]	Threshold=10: [3071, 3072, 3086]
10	no such clusters	Threshold=7.5: [3060, 3076, 3084]	no such clusters

The effectiveness of clustering by finding graph structures depends to a large extent on the threshold value:

At `Threshold = 1`, most of the found clusters consist of duplicate comments resulting from unintentional user errors and are not a sign of coordinated activity. Their distinguishing feature is that the sender, recipient, and content match, and the ordinal numbers follow one another. Besides duplicates, at this threshold value, comments were found that partially form clusters 4 and 1.

At `Threshold = 5`, the results did not change substantially. Cluster 2 was identified, as well as two additional comments from cluster 4.

At `Threshold = 7.5`, further expansion of found clusters and the appearance of new clusters are observed. At the same time, those that contain messages with the same meaning but written in different words are still incomplete.

At `Threshold = 10`, the most complete version of cluster 1 was found. A number of new clusters were also identified, which contain signs of coordinated activity.

3.3. Identification of low-level YouTube manipulations based on analysis of reply relevance anomalies

The algorithm for adding parent content information to the dataset was implemented in Python. The script is provided in Appendix 6.

During the dataset transformation, the video description is used as `parent_text` for first-level comments. The following text was used (given here in translation; the original language matches the language of most comments): «This video shows peaceful Mariupol in December 2021, just before New Year's. Soon, war will come to the city, many people will be killed, and many buildings will be destroyed. But for now, everything is beautiful and intact, and everyone is alive. Christmas trees, the Drama Theater, and other New Year's locations are shown.»

Running the script produced the following statistics for comments of different levels:

- 1,195 1st-level comments (the parent content is the video description);
- 837 2nd-level comments (the parent content is 1st-level comments);

- 1,668 3rd-level comments (the parent content is 2nd or 3rd level comments).

Because running the models used in this part of the study requires significant computational resources, it is necessary to specify the author's computer system characteristics. All models were run on a computer with the following configuration: CPU: i7-8750H 2.2GHz, RAM 24Gb, GPU: GTX 1050 4Gb. Some models' run parameters (for example, batch size) were optimized specifically for this configuration and should be selected individually when reproducing results on computing systems with different characteristics.

For each of the considered models (Rubert-tiny2, GrigoryT22 cross-encoder-ru, Gemma3:1b, Gemma3:27b), Python code was written that accounts for the specifics of each model. In particular, this concerns model and data loading, calling the model, receiving and saving results, and performance optimization. Running the code adds a `reply_prob` field to the dataset. It contains the model's estimate of the probability that the text in the `textOriginal` field is a reply to the text in the `parent_text` field. The resulting scripts are quite large, so they are not included in the Appendixes, but can be provided by the author upon request.

Rubert-tiny2 is the most "lightweight" and fastest of all BERT models for Russian. It does not include an additional layer for detecting textual relations, so we determine them indirectly by computing cosine similarity. Analysis of the outputs showed that the range of cosine distances is relatively narrow. Therefore, to convert cosine similarity outputs into a probability-like form, linear scaling (Min-Max Scaling) was applied. This procedure maps an arbitrary similarity score range onto the standard unit interval [0;1].

$$P = clip\left(\frac{s-s_{min}}{s_{max}-s_{min}}, 0, 1\right), \quad (3.1)$$

where s is the original cosine similarity value;

s_{max} and s_{min} are defined as the 95th and 5th percentiles of the cosine similarity value range;

$clip$ is a function that truncates values outside the unit interval, preserving the probability semantics.

The dataset processing time with Rubert-tiny2 was 11s.

Experiments with GrigoryT22/cross-encoder-ru showed that the model is sensitive to the order of input texts and, in some cases, for

the order [text1, text2] it may return “not related,” whereas for [text2, text1] it may return “related.” This is explained by the fact that the model was trained on “question–answer” data and therefore accounts for the order of text. In our case, there is no strict separation into questions and answers in comments. Moreover, during a dialogue, a comment at the next level may be formulated as a question to the previous one. Therefore, to remove order sensitivity, we introduced a bidirectional check. This means that each pair of comments was passed through the model in both forward and reverse order, and the larger result was used as the final output. The same scaling procedure as for the previous model was applied to the outputs using formula (3.1). To speed up processing, comment pairs were passed in batches. It was found that the optimal batch size for the used computer configuration is 8–32 pairs.

The dataset processing time with GrigoryT22/cross-encoder-ru was 2m:05s.

When working with GPT-architecture models, a “Zero-shot prompting” approach was used. It assumes that the model is configured solely via a prompt, without additional training or examples (DAIR.AI, 2025; Wei et al., 2021). A practical difficulty was obtaining numerical output from models primarily designed to generate text. After several experiments, the following prompt was synthesized (given here in translation; the prompt language matches the language of the comments):

```
«You are given N pairs: a parent comment and a proposed reply.\n»
«For each pair, return a single number from 0 to 1 with a decimal
point – the probability that the second text is a reply to the
first.\n»
«Return exactly N numbers, one per line, with no words or
explanations.\n»
«0 – definitely not a reply; 1 – definitely a reply.\n»
«Consider pragmatics: expressions of thanks, agreement/
disagreement, clarification, direct address, and quoting indicate
a higher probability.\n»
«A short, slogan-like phrase indicates a low probability.\n»
«Important: each answer must be a number only (e.g., 0.83).»
```

For Gemma3:1b, the local API of the Ollama program was used, which emulates chat mode. The Python script sends instructions to the model on behalf of the user, parses the response, and saves it to the dataset in the new `reply_prob` field.

To speed up processing, instructions, as with BERT models, are sent in batches. The optimal batch size was experimentally set to 8 comment pairs.

Because GPT models consume much more resources and work much more slowly than BERT models, initial experiments and script debugging were performed on a small sample (200 rows) from the full dataset. Only after satisfactory results were obtained, the model was run on the full set of comments.

The dataset processing time with Gemma3:1b was 1h:7m:05s.

For local execution of Gemma3:27b, the computer used in the study lacked sufficient resources. Therefore, the model was run using Google Cloud resources. The study tasks are sufficiently covered by the Gemini API free tier offers. Since Google limits the number of requests for free-tier projects (30 RPM, 14400 RPD), it was necessary to select API call parameters to ensure high performance while staying within the limits. As a result, a stable and sufficiently fast option was achieved with the following parameters:

```
BATCH_SIZE_MAX = 10
MAX_PROMPT_CHARS = 6000
SLEEP_BETWEEN = 5.0
RATE_LIMIT_RPM = 20
```

The script runtime with Gemma3:27b was 1h:44m:58s. However, this time cannot be used to compare model performance, because Gemma3:27b was run on Google Cloud resources, while the local computer was used only for auxiliary tasks.

Thus, after processing the input dataset with the considered models, four files were obtained: *yt_sim_rubert.csv*; *yt_sim_grigoryt22_bidir.csv*; *yt_sim_gemma1b.csv*; *yt_sim_gemma27b.csv*.

For further analysis, a combined set of research tools is used. For fast execution of basic data procedures, including sorting, filtering, grouping, visualization, and calculation of main statistical characteristics, Orange Data Mining is used (Fig. 3.6). For statistical calculations and hypothesis testing, Python with the corresponding modules is used.

Using this workflow, we calculate statistics on the reply relevance scores produced by different models across comment levels (Table 3.7).

The data in Table 3.7 show substantial differences in the relevance scores obtained with different models. Moreover, not only are the absolute score values different, but the overall conclusions are as well. For

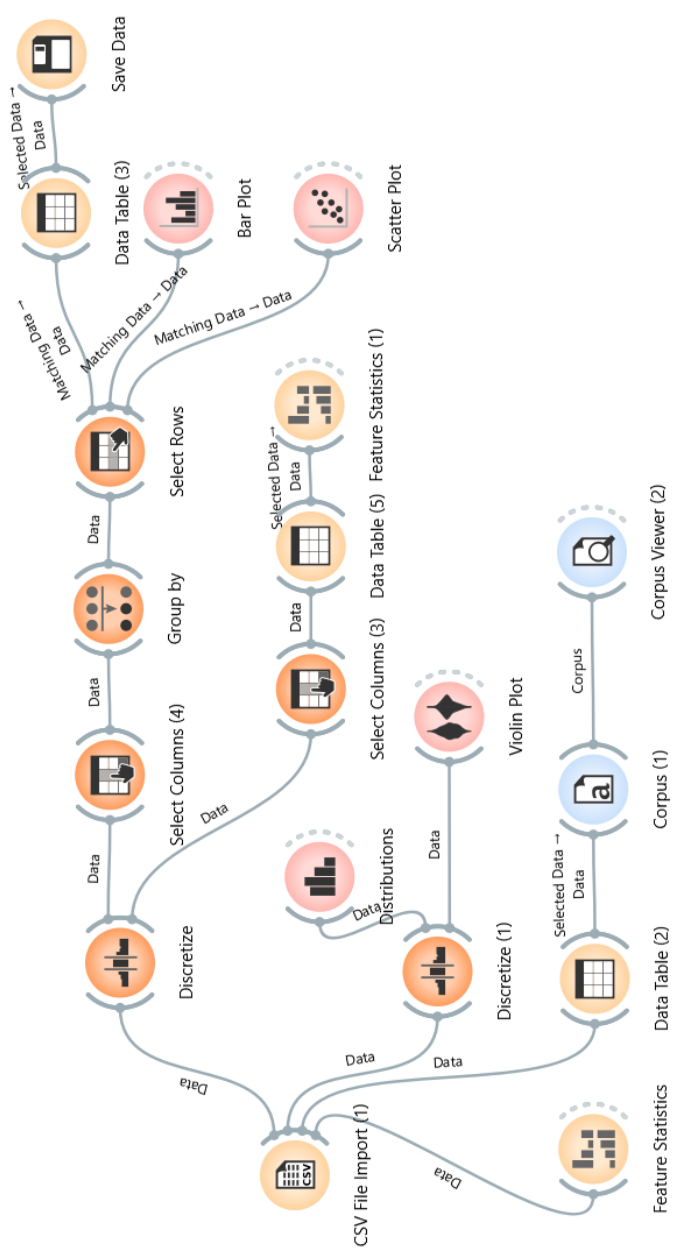


Fig. 3.6. Workflow for analysis comments relevance

Table 3.7
Statistics on the relevance of replies in the context of comment levels and models

Comment group (level)	Number of comments	Models							
		Rubert-tiny2 (29.4M)		GrigoryT22 cross-encoder-ru (132M)		Gemma3:1b (1B)		Gemma3:27b (27B)	
		mean	st.dev	mean	st.dev	mean	st.dev	mean	st.dev
1 st level	1195	0.47122	0.11588	0.29247	0.39306	0.68089	0.34536	0.57044	0.3254
2 nd level	837	0.50258	0.11575	0.29722	0.39601	0.67651	0.36114	0.42567	0.31705
3 rd levels	1668	0.52510	0.11865	0.345	0.41808	0.62669	0.38584	0.442	0.30949

example, the BERT-group models assessed 1st-level comments as the least relevant and 3rd-level comments as the most relevant. The GPT-group models produced the opposite result.

In the next section, we compare these models and determine which produces higher-quality estimates.

4. Statistical tests and discussion

4.1. Statistical assessment of model quality

Let us attempt to statistically evaluate how effective the selected models are at estimating the relevance of replies to their parent comments. As shown in Table 3.7, the average relevance scores for various comment groups in the Gemma3:27b model vary quite significantly (favoring top-level comments), while for the other models, this difference is less marked. We will examine whether such differences could be caused by random factors. For this purpose, we formulate the null hypothesis as follows:

H_0 : all groups have the same distribution (the same medians).

As an alternative hypothesis, H_1 , we assume that, in at least one group, the median (or the distribution shape) differs from those of the others.

To test the hypothesis, we use the non-parametric *Kruskal–Wallis* rank test (Agresti, 2018). This test does not require normal distributions or equal variances. It works only with rank information: the values of all observations across groups are pooled, sorted by size, and replaced with ranks. Then the mean ranks in each group are compared, and the test statistic H is calculated using formula (4.1):

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1), \quad (4.1)$$

where k – the number of groups;

N – the total number of observations;

n_i – the number of observations in group i ;

R_i – the sum of ranks in group i .

For $n_i \geq 20$ an approximation is valid where the distribution of H is approximated by the χ^2 distribution with $f = k - 1$ degrees of freedom. That is, the H_0 is rejected if $H \geq \chi^2_{\alpha}(k-1)$. This happens when the mean ranks in the groups differ strongly.

The results of the *Kruskal–Wallis* test calculations are shown in Table 4.1.

Table 4.1

Kruskal-Wallis statistics for the models studied

Model	H-statistics	p-value	Inference ($\alpha=0,05$)
Rubert-tiny2	125.874209	$4.643e^{-28}$	reject
GrigoryT22 cross-encoder-ru	2.040443	0.360515	accept
Gemma3:1b	6.618620	0.036541	weak reject
Gemma3:27b	159.775028	$2.02e^{-35}$	reject

Thus, at the significance level $\alpha = 0.05$, we have no grounds to reject the H_0 hypothesis only for one model: GrigoryT22/cross-encoder-ru, and at the significance level 0.025, for two models: GrigoryT22/cross-encoder-ru and Gemma3:1b. The differences between groups identified by the Rubert-tiny2 and Gemma3:27b models turned out to be statistically significant.

At the same time, an analysis of Table 3.7 shows that, in essence, these two models give opposite assessments. The results of Rubert-tiny2 indicate that the smallest number of relevant comments belongs to the top level. The results of Gemma3:27b indicate that, on the contrary, the top level contains the largest number of relevant comments. Therefore, we need to determine which of these models produces correct results and which one makes errors.

To resolve this contradiction, we refer to the model outputs (the files *yt_sim_rubert.csv* and *yt_sim_gemma27b.csv*) and analyze comment pairs that received extreme *reply_prob* scores.

The analysis showed, in particular, that a large number of top-level comments in which users thanked the video author for the beautiful footage and for reminding them of their home city were marked by Rubert-tiny2 as low-relevance, whereas Gemma3:27b assigned them high relevance scores. This feature of Rubert-tiny2 can be explained by the fact that the model processes comment texts separately and produces only their vector representations, which we then compare. Therefore, in more complex cases when the compared phrases are not semantically similar, the model makes errors, leading to a lower mean relevance score for 1st-level comments. Thus, preference should be given to Gemma3:27b, which is quite logical, because this model is the most complex among all those considered.

The distribution of relevance assessment results by comment level produced by Gemma3:27b is shown in Fig. 4.1. The diagram clearly shows that most 1st-level comments are assessed as relevant by the model, whereas for 2nd- and 3rd-level comments, the opposite pattern is observed.

Next, we consider the relevance assessment results obtained from Gemma3:27b in the context of detecting manipulations and bot activity.

Fig. 4.2 shows the distribution of mean relevance scores over the observation period. These data are provided in tabular form in Appendix 7.

An analysis of Fig. 4.2 shows that the mean comment relevance changed substantially during the observed period. In the first months after the video was published, the mean relevance score was about 0.6, but in December 2022, it declined sharply, first to 0.48 and then to 0.3.

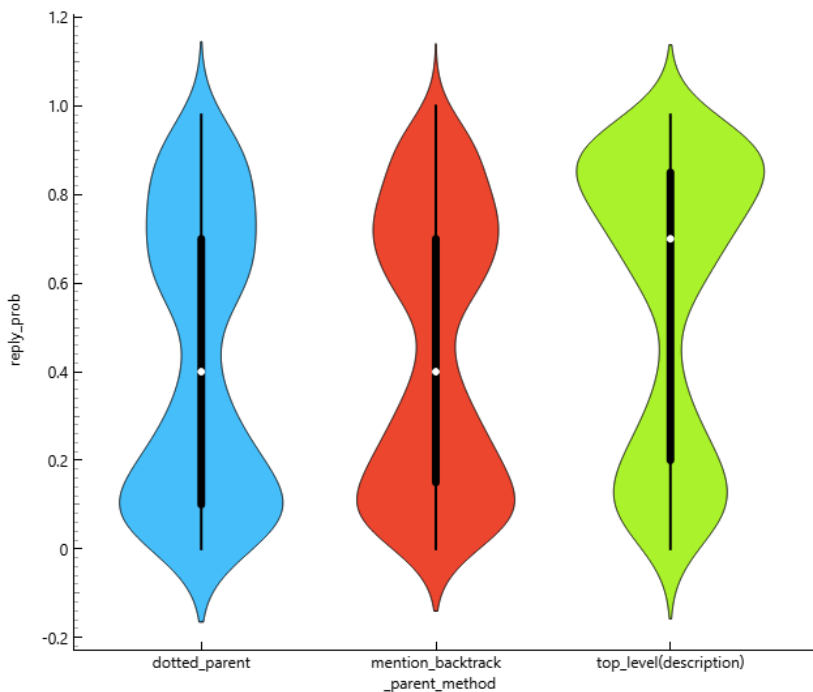


Fig. 4.1. Gemma3:27b. Violin diagram for the comments' relevance assessments

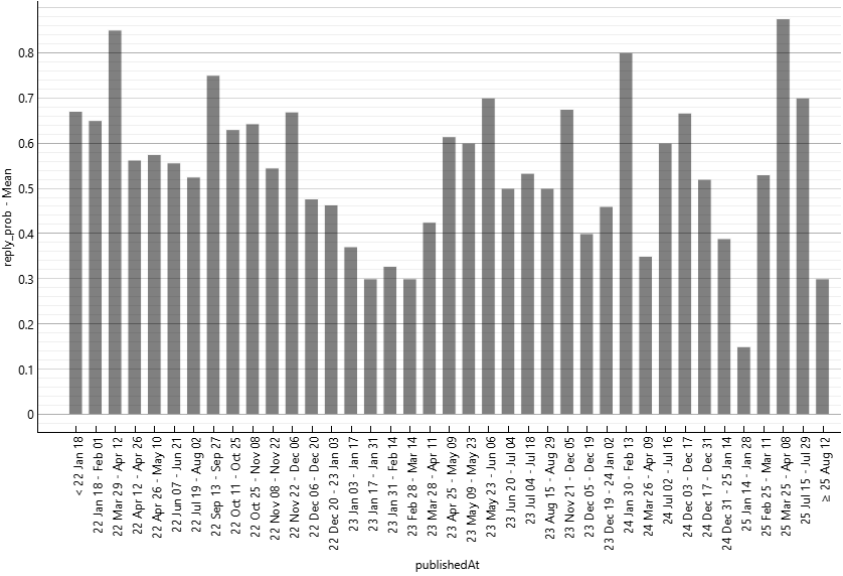


Fig. 4.2. Gemma3:27b. Distribution of the comments' relevance assessments by time

Comparing these observations with the data in Fig. 3.1 and Fig. 3.2, we can see that the decrease in relevance coincides with the increase in comment activity.

Thus, in different time periods, the mean relevance scores produced by the same model differ substantially. We will check whether such changes can be explained by random fluctuations or not.

4.2. Statistical testing of the bot-activity hypothesis

We compare two observation periods. For the first period, we take the first three months after the video was uploaded, corresponding to January–March 2022. Most of these comments were made before the war. In the second period, we take observations in the interval 06.12.2022–20.12.2022, which contains the highest commenting activity. Numerical data for the analysis are provided in Appendix 7.

Since we have no a priori information about what the mean relevance level of replies is “normal,” we formulate the null hypothesis as follows:

H_0 : the true probability of meaningful replies equals the average level in one of the two considered periods, and deviations from it are caused by random fluctuations.

As the main input for analysis, we use the aggregated indicator `reply_prob`, interpreted as the individual probability that a comment was written by a human and represents a meaningful reply. For each time period, the mean value of `reply_prob`: $\bar{p}$ and the total number of comments (n) are known. When testing statistical hypotheses, the “number of meaningful replies” is approximated by $k \approx \bar{p}n$ and modeled using a binomial distribution. This approach is standard in statistical inference for proportions: when individual observations can be treated as independent Bernoulli trials with an approximately constant probability of success within a given period, the number of “successes” can be modeled as a binomially distributed variable $X \sim \text{Bin}(n, p)$ (see Casella & Berger, 2002; Agresti, 2018; Wasserman, 2004).

For definiteness, we compute the baseline level using the first three observation periods. Their weighted mean is $\hat{p}_{base} = 0.6818$ with a total number of comments $n_{base} = 44$, which corresponds to $k_{base} \approx 30$ “successes”. The alternative period from 06.12.2022 to 20.12.2022 has $n_{test} = 2907$ and $\hat{p}_{test} = 0.4766$, hence $k_{test} \approx 1386$.

Because we do not know which of these values is “normal” and which is “anomalous,” we test the null hypothesis in both possible directions.

The *first direction* sets the true probability under the null hypothesis as $H_0: p = \hat{p}_{base} = 0.6818$ and checks whether the drop of $\hat{p}_{test}$ to 0.4766 is statistically significant, i.e., $H_1: p < \hat{p}_{base}$.

We perform the test using a binomial test for the probability that the random variable X_{test} follows a binomial distribution with parameters n_{test} and $\hat{p}_{base}$, i.e., $X_{test} \sim \text{Bin}(n_{test}, \hat{p}_{base})$. The calculated p-value of the binomial test represents the probability that the observed number of successes X is less than or equal to k_{test} : $pval_1 = \Pr\{X \leq k_{test}\}$. This probability turned out to be extremely small: $pval_1 \approx 1.50 \times 10^{-115}$.

The p-value computed with a z-test was almost the same. The standardized z-test statistic is:

$$z_1 = \frac{\hat{p}_{test} - \hat{p}_{base}}{\sqrt{(\hat{p}_{base}(1 - \hat{p}_{base})/n_{test})}} \approx -23.75$$

Thus, the observed deviation from the null hypothesis is almost 24σ . The probability of such a deviation is $pval_1^{(z)} \approx 5.41 \times 10^{-125}$.

The exact one-sided Clopper–Pearson bound for the observed values in the period 06.12.2022–20.12.2022, computed at the 95% level, is $[0, 0.4922]$, which is also entirely below the baseline level 0.6818.

Therefore, based on three different statistical checks, the null hypothesis in the first direction is confidently rejected at the significance level $\alpha = 0.05$ (as well as for any other reasonable significance level). The difference between the mean values of `reply_prob` in the observed periods is statistically significant and cannot be attributed to random fluctuations.

The *second direction* of testing assumes “reverse” calculations: the value of `reply_prob` in the period 06.12.2022–20.12.2022 is treated as the norm, $H_0: p = \hat{p}_{test} = 0.4766$, and its higher value in the baseline sample is interpreted as a potentially random deviation. The alternative hypothesis is $H_1: p > \hat{p}_{test}$.

A binomial test for $X_{base} \sim \text{Bin}(n_{base}, \hat{p}_{test})$ gives $pval_2 \approx 0.004816$.

The standardized z-test statistic is:

$$z_2 = \frac{\hat{p}_{base} - \hat{p}_{test}}{\sqrt{(\hat{p}_{test}(1 - \hat{p}_{test})/n_{base})}} \approx 2.725$$

In this case, the p-value corresponds to the probability of observing a 2.725σ deviation from the null hypothesis and equals:

$$pval_2^{(z)} \approx 0.0032$$

The exact one-sided Clopper–Pearson bound for the observed values in the baseline period, computed at the 95% level, is $[0.5481, 1.0000]$, i.e., it lies above the mean value in the alternative period of observation (0.4766).

Thus, the p-values obtained in different ways, $pval_2 \approx 0.0048$ and $pval_2^{(z)} \approx 0.0032$, provide grounds to reject the null hypothesis in the second direction both at the common significance level $\alpha = 0.05$ and at the stricter level $\alpha = 0.01$. The higher share in the baseline sample is statistically significant and is unlikely to have occurred by chance if the “norm” were 0.4766.

A joint interpretation of both tests allows us to conclude that the sharp decrease of the mean `reply_prob` value during the period of high commenting activity is inconsistent with the hypothesis of a constant

true share of meaningful replies at the level of either of the considered periods.

At the level of formal statistics, both directed tests reject the corresponding null hypotheses, both at the standard significance level $\alpha = 0.05$ typically used in social sciences and at the stricter level $\alpha = 0.01$. We can conclude that the observed asymmetry is due to exogenous factors, and the increased activity in December 2022 indeed led to a systematic decrease in the probability of a “human meaningful reply.” This can be considered as support for the hypothesis of increased bot activity during this period.

4.3. Research discussion

The results reported in Section 3.2 indicate substantial anomalies in both the temporal dynamics of commenting activity and the emotional background of the discussion. First, the distribution of comment frequency over time was extremely uneven. Fig. 3.1 shows a sharp activity burst in December 2022, when about 92.4% of all comments across the entire observation window were posted in that month. A spike of this magnitude is outside typical user behavior on YouTube and should be treated as anomalous. It is plausible that the onset of the burst was triggered by the platform’s algorithmic promotion of the video; however, subsequent inspection of the data revealed additional signals consistent with coordinated activity, suggesting that the initial exposure effect may have been amplified by exogenous actors. In general, such concentrated bursts appear unnatural and may indicate external influence. Similar patterns are described in prior work: for example, Schoch et al. (2022) show that in political information attacks, a large share of activity often occurs within compressed time intervals that do not resemble organic discussion growth. In our case, the December peak became the key period for further anomaly detection and for tracing potential manipulation.

Along with the rise in the number of comments, the emotional tone also changed. The *sentiment analysis* demonstrates a noticeable shift in the emotional background during the anomalous activity period. In the first weeks after publication, the dominant reactions were neutral or moderately positive, as shown in Fig. 3.2, where early comments cluster around neutral/positive values. This is typical for newly uploaded

content, when viewers mostly thank the author or express positive emotions. In December 2022, however, the peak in activity coincides with an increase in negative comments. The emotional range expands sharply, with both highly negative and highly positive messages appearing simultaneously, while the overall average sentiment weakens. Fig. 3.3 shows that the mean monthly sentiment in December 2022 is only slightly above zero (close to neutral), even though the amplitude of sentiment values in that month is the largest in the entire period. In other words, the discussion becomes polarized at that point: extreme negativity and extreme positivity coexist and “flatten” the mean tone toward neutrality.

Such emotional instability, combined with an abrupt activity burst, can serve as an indirect indicator of bot involvement or organized groups. Ellaky & Benabbou (2024) discuss sharp oscillations of emotional tone in anomalous periods as one of the signals associated with bot activity. In our data, the peak period is indeed accompanied by an inflow of negative remarks and by a loss of the distinctly positive tone seen earlier, which may reflect an attempt to steer the discussion into a more conflictual direction. At the same time, as Ellaky & Benabbou emphasize, this signal is not proof in itself: negative comments can also arise as an organic response to external events. For that reason, emotional metrics alone are insufficient for a reliable conclusion, and a more comprehensive content-oriented analysis is required.

It is also important to acknowledge that our sentiment analysis relied on a universal, but relatively coarse, unigram-based multilingual sentiment lexicon described in Chen & Skiena (2014). Its accuracy was sufficient for this dataset, which includes many strongly polarized statements; however, for broader applications, it is preferable to use richer n-gram approaches or more accurate AI-based methods, especially when subtle pragmatic cues and context-dependent polarity matter.

To refine the manipulation-related conclusions, we then searched for repeated or semantically similar messages. Several vectorization and clustering strategies were tested, and the practical outcome was clear: methods that capture semantics provide markedly better results than purely lexical approaches. Specifically, the comparison between Bag-of-Words and SBERT-based embeddings showed that the Bag-of-Words approach produces a rapidly growing number of “similar” pairs as the similarity threshold increases, yet many of those pairs are semantic false positives. Bag-of-Words does not capture the meaning of emojis, and it

tends to merge very short comments simply because their vocabulary is sparse; as a result, the method may cluster messages that differ in meaning but are similar in length and surface form. In our experiments, at a similarity threshold of 7.5, Bag-of-Words collapsed hundreds of short comments into a single cluster that shared no coherent topic beyond brevity, which illustrates the limits of lexical matching for social-media text. By contrast, document embeddings (SBERT) yielded distance-based pairs that were genuinely close in meaning, even when written in different words, in different languages, or with key content conveyed through emojis. This leads to an unambiguous inference: embedding-based vectorization substantially improves duplicate and paraphrase detection, and it is particularly well-suited to heterogeneous social-media content.

Semantic clustering was then used to extract groups of potentially suspicious messages. Both visual methods (hierarchical clustering and t-SNE) and an algorithmic search for clusters in the distance matrix (a graph-based method) were applied. The visual analysis provided valuable initial insight. When inspecting the full dendrogram built from embeddings (Fig. 3.5), we observed several large branches comprising highly similar comments. A representative example is cluster 1 (Fig. 3.6), which contained many comments of a similar format, with 67% of them produced by a single author (*user 339*). However, a closer reading showed that, despite their formal similarity, these messages did not have an explicit manipulative objective: they were friendly responses from the video author to viewers rather than spam or propaganda. This cluster is better explained by community self-organization, where an active author replies to multiple users in a similar style, and it is not evidence of external interference. This aligns with Polupanova et al. (2025), who note that intense, homogeneous activity can emerge organically in online communities in response to events or content. The key implication is methodological: not every homogeneous cluster is a “bot attack,” and content inspection is necessary to separate organic coordination from artificial amplification. Cluster 1 thus illustrates the risk of false positives, which remains a major challenge for automated manipulation-detection systems.

At the same time, the visual methods also exposed clusters that are difficult to explain organically. Fig. 3.7 presents cluster 2, in which comments clearly reproduce the official propaganda narrative. These messages were posted in replies to different users, yet they are semantically

near-identical and contain repeated propaganda slogans supporting a particular position. A set of comments carrying an identical ideological payload across multiple dialogues is a strong signal of coordination. The probability that similar propaganda statements appear independently across different threads in nearly the same form is low; such a pattern is more consistent with bots disseminating prepared text or with synchronized troll activity. Comparable astroturfing signatures are described by Keller et al. (2020) and Schoch et al. (2022), who show that accounts involved in campaigns frequently reuse the same messages or hashtags verbatim. Our cluster 2 is consistent with these findings and suggests coordinated narrative injection in YouTube discussions.

Another visually detected cluster is the small group highlighted on the t-SNE map as cluster 4 (Fig. 3.8; content in Fig. 3.10). It contains messages that are best classified as manipulative promotional content: the comments consistently direct attention to a specific religious movement unrelated to the video. In effect, this is covert advertising or spam inserted into the discussion. This cluster can also be treated as manipulation, albeit of a different type: the goal here is not political but outreach-oriented, aiming to redirect attention to external content. Methodologically, the detection logic remains the same: a coherent group of semantically similar messages that is misaligned with the video context serves as a warning signal of potential manipulation.

Despite their usefulness, the visual approaches (dendrogram inspection or t-SNE “clouds”) are labor-intensive and partly subjective. In large datasets, the analyst may miss some anomalous clusters or, conversely, overinterpret structures that are not truly meaningful. This motivated the implementation of an automated graph-based algorithm that detects clusters as connected components in the distance matrix. The results show that the graph approach, especially when applied to the embedding-based distance matrix, can recover almost all visually evident meaningful groups and can additionally surface new clusters. The threshold parameter plays a decisive role in what type of anomaly is detected. At very low thresholds=1, the method mostly finds literal duplicates which often result from accidental double-posting and therefore represent noise rather than manipulation. As the threshold increases to the 5–7.5 range, broader clusters of semantically similar messages emerge, including the propaganda and promotional clusters described above. At high thresholds=10, the method yields the most complete coverage

of repeated narratives in the dataset. In practical terms, this supports a layered screening strategy: low thresholds highlight strict duplicates (including the most “primitive” copy-paste behavior), while higher thresholds reveal paraphrased clusters where more advanced semantic processing, including LLM-based inspection, becomes valuable.

When we relate these findings to the research objective, the most effective toolkit for anomaly detection in YouTube comments proved to be a combination of activity analysis (to identify suspicious time intervals), sentiment analysis (to detect atypical emotional volatility), and, most importantly, semantic clustering. This combination enabled the discovery of concrete groups of messages that are plausibly associated with bots or coordinated trolls. The outcome aligns with the broader literature, which recommends multi-signal approaches: each individual indicator may have a benign explanation, but the co-occurrence of multiple signals within the same time window substantially increases confidence in coordinated manipulation (Mihaylov et al., 2015; Keller et al., 2020). In our dataset, this convergence is observed in December 2022: an extreme activity burst, a marked shift toward polarized sentiment, and clusters of repeated propaganda-like replies together point to exogenous interference.

Subsection 3.3 introduced an additional identification mechanism: assessing reply relevance, i.e., the degree to which a reply is substantively connected to its parent comment (or to the meta-context, such as the video description for top-level comments). The underlying hypothesis is supported by prior observations that bots and trolls often publish messages weakly tied to the conversational context, such as generic slogans, insults, or spam (Marcondes et al., 2024). While low relevance is not a strong enough feature to classify individual comments in isolation, a systematic deviation of average reply relevance from baseline values can serve as a statistical indicator of coordinated activity.

Four models were used to quantify relevance, each assigning a probability that a given “comment-parent” pair constitutes a meaningful reply. The models differ in architecture and capacity, ranging from the lightweight RuBERT-tiny2 to the large LLM Gemma3:27b. The comparison was informative: different model families produced markedly different relevance distributions (Table 3.7). RuBERT-tiny2 and Gemma3:27b yielded statistically significant differences between comment levels, but with opposite directional conclusions. RuBERT-tiny2 suggested the low-

est relevance for top-level comments and the highest relevance for third-level replies, whereas Gemma3:27b produced an intuitively more plausible pattern: top-level comments mostly contain meaningful responses to the video, while second- and third-level threads contain a larger share of context-breaking or low-information inserts. GrigoryT22/cross-encoder-ru barely differentiated the groups (the null hypothesis of equal medians was not rejected, $p \approx 0.36$), and Gemma3:1b showed only a weak tendency toward higher relevance at the top level (significance around 0.03).

To explain the contradiction between RuBERT-tiny2 and Gemma3:27b, we inspected pairs with extreme relevance scores. This revealed a systematic failure mode of RuBERT-tiny2: many top-level comments that clearly respond to the video’s meaning – expressions of gratitude for filming, memories of Mariupol, praise for the footage – were classified as low-relevance compared to the video description. Gemma3:27b, in contrast, assigned high relevance to these cases. The reason is methodological: RuBERT-tiny2 produces text embeddings independently, and cosine similarity does not reliably capture pragmatic relatedness when lexical overlap is limited. A “thank you for the beautiful footage” reply is semantically and pragmatically connected to the video, yet it may share a few content words with the description. As a result, the embedding-similarity proxy underestimates relevance. The larger LLM, being more context-sensitive, captures this pragmatic relation. Therefore, we treat Gemma3:27b relevance assessments as the most adequate among the tested approaches for the specific task formulation used here. This also affects how the *Kruskal–Wallis* results should be interpreted: strong between-group differences in Gemma3:27b ($p \approx 2e^{-35}$) are consistent with meaningful discrimination, whereas similarly strong differences in RuBERT-tiny2 ($p \approx 4.6e^{-28}$) reflect systematic misclassification rather than a useful signal.

The violin plot in Fig. 4.1 supports this interpretation by showing that top-level comments are concentrated at higher `reply_prob` values, while second- and third-level replies have a substantial mass at low values. This pattern is consistent with the intuition that context-breaking content is more prevalent deeper in threads, where manipulative actors may attempt to blend into ongoing discussions with low-effort inserts. Related tactics, including “flooding” discussion spaces, are described in the literature on computational propaganda and coordinated inauthentic behavior (e.g., Shao et al., 2018).

Another important result emerged from examining the temporal dynamics of reply relevance. The Gemma3:27b scores were aggregated by month, and a plot of the average probability of a meaningful reply (`reply_prob`) was produced for each period (see Fig. 4.2). The plot shows a substantial decline in this indicator during the period of maximum activity. While in the first months after the video was published (January–March 2022), the average `reply_prob` remained at about 0.6 (i.e., most comments were on topic), in December 2022, it dropped sharply: first to about 0.48 at the beginning of the month and then to about 0.30 by the second decade. In other words, at the peak of the comment influx, the share of meaningful replies decreased by roughly half compared to the baseline level. This drop coincides precisely with the anomalies identified earlier: the December peak in message volume (Fig. 3.1) and the surge in negative polarization (Fig. 3.2) were accompanied by a mass emergence of content-poor comments. It is therefore reasonable to assume that a substantial share of these “*low-relevance*” messages resulted from the activity of trolls or automated bots. This technique – “cluttering” a discussion with irrelevant posts – is well documented in studies by Woolley (2023), Howard & Kollanyi (2017), and others. The goal of such information manipulation is to create an illusion of public opinion through a mass of similar comments or to distract the audience’s attention.

To verify that the observed relevance drop is not a random fluctuation, we conducted formal statistical tests (Subsection 4.2). The null hypothesis assumed an unchanged underlying share of meaningful replies, with deviations driven by random variability. Using one-sided binomial tests, z-tests, and Clopper–Pearson bounds, and checking both “directions” of the null hypothesis, we consistently rejected H_0 at $\alpha = 0.05$ and even at $\alpha = 0.01$. This means that the decrease in mean `reply_prob` during the high-activity window is systematic and unlikely to be explained by chance. In practical terms, the results support the conclusion that the December 2022 spike is associated with an exogenous factor that reduces the proportion of meaningful, human-like replies, consistent with elevated bot or coordinated-troll activity.

As a result, the study identified several specific indicators of low-level manipulations and coordinated activity. Table 4.2 summarizes these indicators, along with our subjective assessment of the strength of each signal (on a 0–3 scale, where 0 indicates the signal is absent and 3 indicates it is clearly expressed).

Table 4.2

Intensity assessment of low-level manipulation indicators identified in the study

Indicator of anomalous/coordinated activity	Intensity (0-3)
Temporal activity anomalies: almost the entire discussion volume fell within a single decade of December 2022, which is atypical for organic audience growth.	3 (very strong)
Emotional tone anomalies: unusually high sentiment polarization during the December activity peak, whereas earlier the tone was more stable.	2 (noticeable)
Relevance anomalies: a substantial share of comments during the spike were off-topic or content-poor.	3 (very strong)
Duplicating and repeating comments: cases of identical phrases posted repeatedly were observed (mostly due to user errors, but this technique can also be used by bots).	1 (weak)
Clusters of similar messages: groups of semantically similar comments addressed to different users and written by different authors were found, indicating coordinated actions.	2 (moderate)
Promotion of external content/spam: a cluster of messages promoting a religious movement unrelated to the video was identified, indicating an external dissemination campaign.	2 (moderate)
Unusual activity by individual authors: an example was found of a user posting dozens of similar replies. In this case, it appeared to be organic activity, but such a pattern can also indicate a "bot farm."	1 (ambiguous)

As Table 4.2 shows, the strongest indicators of low-level manipulations were abnormal timing of activity and a decline in message relevance. In this case, nearly all indicators converged within the same time window – December 2022. Each signal, considered separately, could have an alternative explanation (for example, the comment spike could be driven by recommendation algorithms, and negative tone could reflect external events). However, their simultaneous appearance, together with the detected clusters, strongly suggests a planned influence operation. Less pronounced signals (duplicates and isolated highly active authors) were also present but played a supporting role. Naturally, we do not claim that the intended target of the planned influence was this specific video; rather, it likely became one of many objects within a broader information campaign.

4.4. Research limitations

When interpreting the results, several limitations should be considered. First, the analysis covered comments from only one YouTube video focused on a specific topic and context. Although we identified patterns consistent with manipulation, the scale and form of these patterns may depend on the video's theme and its audience's characteristics. For that reason, the quantitative estimates obtained here, including relevance statistics, should not be directly generalized to other videos, other channels, or other platforms. A broader assessment of how common such phenomena are would require additional studies across multiple topics, communities, and time periods.

The ability to detect manipulative activity is also constrained by the structure of the data provided by YouTube. The platform offers a relatively "flat" discussion structure: practical nesting is limited to a small number of levels, and direct ties between commenters cannot be reconstructed to support robust network-level analysis. As a result, high-accuracy bot-network-detection approaches described in the literature are either not applicable to YouTube in this setting or provide only limited insight. In this study, we therefore concentrated on content-based and behavioral markers within the comments themselves, using time-based distributions and semantic analysis, but this focus inevitably leaves some coordination signals outside the scope of observation.

A related limitation follows from the fact that the API data does not fully specify the parent content for deeper replies. In our dataset, establishing the parent text for third-level comments required a heuristic procedure described in Subsection 2.3. In most cases, the algorithm works well, but it can fail in certain scenarios, for example, when there is a large time gap between a reply and the likely parent comment, or when the potential parent author posted many other comments nearby. Such errors introduce noise into the constructed "parent text – reply" pairs and, consequently, into relevance assessment.

Another important constraint is the unobserved effect of moderation. The dataset contains only the comments that remained visible under the video at the time of collection. We have no information about comments removed by YouTube's moderation systems or deleted by the video author, and we do not even know whether such removals occurred or how often they occurred. If a substantial number of problematic

comments were removed, our dataset would underestimate the true intensity of manipulative activity. In this sense, the limitation primarily affects completeness rather than the direction of inference: the presence of “invisible” removed comments would more likely strengthen than weaken the manipulation signals.

The next limitation concerns the volume and temporal distribution of the data. While the total number of comments is not small (around 3,700), the overwhelming majority are concentrated in a single month. Strictly speaking, this is insufficient for training supervised classifiers or building complex models that require balanced and diverse temporal coverage. The same issue affected hypothesis testing: beyond the December 2022 burst, the only other period with a meaningful number of comments was the early post-publication stage, which restricted the range of comparisons. In statistical testing, we used a binomial model, which is appropriate under large sample sizes and independence assumptions, but in reality, comments are not fully independent because one provocation can trigger cascades of replies. We treated independence as an approximation. Even so, the extremely small p -values obtained suggest that the main conclusion is robust to moderate violations of these assumptions.

Further limitations arise from the text-processing toolchain. We relied on existing pretrained models (SBERT, RuBERT, Gemma, and others) and used a zero-shot approach for the GPT-family models. Their performance depends on how well they handle Russian-language YouTube comments in practice, including misspellings, emojis, slang, and code-switching. Larger models demonstrated better pragmatic understanding, which is reflected in the more plausible behavior of Gemma3:27b for relevance estimation, but it cannot be guaranteed that any model is consistently correct in all individual cases. The GrigoryT22 cross-encoder, for example, showed sensitivity to the order of input texts; we mitigated this by bidirectional checking, yet the issue illustrates that each model introduces its own failure modes. Importantly, we did not calibrate the models on task-specific labeled data and relied on pretrained capabilities with light post-processing (e.g., scaling). This makes the methodology more accessible and less resource-intensive, but it may reduce accuracy compared to approaches that fine-tune models specifically for reply relevance. In future work, for more reliable conclusions, it is worth having experts manually mark part of the sample to validate the model’s estimates.

Computational constraints also shaped the experimental design. The largest model, Gemma3:27b, could not be run locally and had to be accessed via a cloud API with rate and quota limits. Even the smaller Gemma3:1b required more than an hour to process the full dataset. These realities imply that scaling the approach to much larger corpora, such as thousands of videos or millions of comments, would require substantial computing resources or a redesigned pipeline with lighter proxy metrics. In practice, faster signals such as sentiment trends, posting rates, and simple repetition indicators can be used as a first-stage filter to isolate suspicious segments, which can then be examined in depth by heavier models. Similarly, in the graph-based clustering stage, it is reasonable to scan with a very strict threshold first to identify literal duplicates, and then apply a more permissive threshold to expand clusters and uncover semantically close messages, treating these stages as complementary rather than interchangeable.

Finally, there is a fundamental methodological limitation related to the verifiability of conclusions. The study infers bot activity from indirect indicators, such as temporal concentration, sentiment volatility, textual similarity, and low reply relevance. We do not possess direct evidence of automation, such as platform-confirmed labels of inauthentic accounts or independent proof that specific commenters are bots. We also did not attempt to identify individual bots, because that would require either cross-video activity traces or manual auditing of user profiles, neither of which is available in our design. Consequently, the conclusions should be interpreted probabilistically: we identify patterns that *jointly* point to a high likelihood of coordinated inauthentic activity, but we cannot attribute the origin of each low-quality or low-relevance comment with certainty. It also remains possible that a portion of the “irrelevant” comments was produced independently by real users, for example, users in a heightened emotional state or users already influenced by external narratives. This uncertainty does not negate the detected anomalies, but it defines the appropriate scope of claims: the study provides evidence of coordinated campaign-like patterns rather than definitive identification of automated actors.

Conclusion

The analysis reveals that the combined set of methods used in this study enabled the identification of signs of low-level manipulation in the examined content with a high degree of confidence. The detected configuration of coordinated activity in the comments to the selected YouTube video, which served as the test case for the proposed methodology, is consistent with a planned external influence. It is unlikely that the campaign was targeted solely at this specific video. More plausibly, the video became one of many targets within a broader information operation. In this sense, the proposed combination of methods can be transferred to the study of other similar objects in the information environment.

To analyze the discussions and identify indicators of external information influence, we used methods such as statistical analysis of how key features are distributed over time, sentiment analysis, message clustering approaches (t-SNE, hierarchical clustering, and graph-based clustering), BERT- and GPT-based large language models, and statistical hypothesis testing. This combination of analytical methods enabled the observation of systematic deviations from baseline patterns across multiple signals simultaneously, thereby substantially increasing confidence in the conclusion that the discussion contains signs consistent with a coordinated information attack. Although the dataset was unlabeled and the study lacked direct platform-level confirmation of bot participation, the identified patterns align with scenarios of astroturfing and trolling described in prior research. The limited data do not allow firm attribution of the organizers or a reliable estimate of the campaign's overall scale, but the combination of intensity, temporal concentration, repetition, and reduced relevance corresponds to the standard profile of artificial interference in online communities. Similar characteristics are used by platforms such as Facebook and Twitter when reporting cases of "coordinated inauthentic behavior". The results of this study show that analogous practices can also manifest in YouTube comment sections and can be detected even when the available information is restricted.

Overall, the study demonstrates practical ways to detect signals of artificially generated low-level media activity. The validation on

a YouTube comment dataset suggests that the approach is sufficiently effective for exploratory detection tasks. Despite the limitations discussed earlier, statistically grounded evidence of a decline in the share of meaningful replies, together with the identification of clusters of repeating narratives, supports the conclusion that the analyzed case involved information influence. These findings provide a basis for considering the research objectives achieved. At the same time, the exploratory nature of the work and the constraints of data availability imply that the results should be interpreted as evidence of coordinated campaign-like patterns rather than definitive identification of specific automated actors.

Future work in this direction should expand both the empirical base and the operational relevance of the methodology. One priority is to collect and analyze additional datasets and test the proposed approach on other online media platforms. Another direction is adapting the methodology for use in real media-monitoring settings. As the results suggest, this will require either substantial computational resources or a redesign toward a two-stage pipeline supplemented with proxy metrics. Such a design would enable early filtering of suspicious segments and minimize reliance on computationally expensive methods, while preserving the ability to perform deeper analysis when required.

References

1. Abu-El-Rub, N., & Mueen, A. (2019, May). Botcamp: Bot-driven interactions in social campaigns. In *The world wide web conference* (pp. 2529–2535).
2. Agresti, A. (2018). *An introduction to categorical data analysis* (3rd ed.). Hoboken, NJ: John Wiley & Sons.
3. Akhtar, M. M., Masood, R., Ikram, M., & Kanhere, S. S. (2024, July). Sok: False information, bots and malicious campaigns: Demystifying elements of social media manipulations. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security* (pp. 1784–1800).
4. Al Tawil, A., Almazaydeh, L., Qawasmeh, D., Qawasmeh, B., Alshinwan, M., & Elleithy, K. (2024). Comparative analysis of machine learning algorithms for email phishing detection using TF-IDF, Word2Vec, and BERT. *Comput. Mater. Contin.*, 81, 3395.
5. Alieva, I., Moffitt, J. D., & Carley, K. M. (2022). How disinformation operations against Russian opposition leader Alexei Navalny influence the international audience on Twitter. *Social Network Analysis and Mining*, 12(1), 80.
6. Allgaier, J. (2019). Science and environmental communication on YouTube: Strategically distorted communications in online videos on climate change and climate engineering. *Frontiers in communication*, 4, 36.
7. Alpert, C., & Adams, D. (2025). We can't tell if we're being persuaded by a person or a program. *Pursuit*. <https://pursuit.unimelb.edu.au/articles/we-cant-tell-if-were-being-persuaded-by-a-person-or-a-program>
8. Althubiti, S. A., Alenezi, F., & Mansour, R. F. (2022). Natural Language Processing with Optimal Deep Learning Based Fake News Classification. *Computers, Materials & Continua*, 73(2).
9. Bahrainian, S. A., & Dengel, A. (2013, November). Sentiment analysis using sentiment features. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* (Vol. 3, pp. 26–29). IEEE.
10. Barbero, F., op den Camp, S., van Kuijk, K., Soto García-Delgado, C., Spanakis, G., & Iamnitchi, A. (2023, November). Multi-modal embeddings for isolating cross-platform coordinated information campaigns on social media. In *Multidisciplinary International Symposium on Disinformation in Open Online Media* (pp. 14–28). Cham: Springer Nature Switzerland.
11. Bioinformatics Laboratory, University of Ljubljana. (2025). *Orange Data Mining* (Version 3.39.0) [Computer software]. <https://download.biolab.si/download/files/Orange3-3.39.0.zip>

12. Bradshaw, S., & Howard, P. N. (2018). Challenging truth and trust: A global inventory of organized social media manipulation. *The computational propaganda project*, 1, 1–26.
13. Bronstein, M. V., Pennycook, G., Bear, A., Rand, D. G., & Cannon, T. D. (2019). Belief in fake news is associated with delusionality, dogmatism, religious fundamentalism, and reduced analytic thinking. *Journal of applied research in memory and cognition*, 8(1), 108–117.
14. Calvo, D., Cano-Orón, L., & Llorca-Abad, G. (2022). COVID-19 vaccine disinformation on YouTube: analysis of a viewing network. *Communication & society*, 223–238.
15. Casella, G., & Berger, R. L. (2002). *Statistical inference* (2nd ed.). Pacific Grove, CA: Duxbury.
16. Chen, Y., & Skiena, S. (2014, June). Building sentiment lexicons for all major languages. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: short papers)* (pp. 383–389).
17. Cointegrated. (n.d.). *rubert-tiny2* [Model]. Hugging Face. Retrieved August 27, 2025, from <https://huggingface.co/cointegrated/rubert-tiny2>
18. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2015). Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems*, 80, 56–71.
19. Cunha, A. A. L., Costa, M. C., & Pacheco, M. A. C. (2019, May). Sentiment analysis of youtube video comments using deep neural networks. In *International Conference on Artificial Intelligence and Soft Computing* (pp. 561–570). Cham: Springer International Publishing.
20. DAIR.AI. (2025, October 30). *Zero-shot prompting*. Prompt Engineering Guide. <https://www.promptingguide.ai/techniques/zeroshot>
21. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186).
22. Dimitrieska, S. (2025). Guerrilla Marketing for Small Businesses. *Предпримачество*, 13(2), 91–104.
23. Discord Trust & Safety Team. (2026, February 17). *Digital Services Act – Information on average monthly active recipients in the European Union*. Discord Support. Retrieved from <https://support.discord.com/hc/en-us/articles/12477677109143-Digital-Services-Act-Information-on-Average-Monthly-Active-Recipients-in-the-European-Union>
24. Döring, N. (2020). Gendered hate speech in YouTube and YouNow comments: Results of two content analyses.
25. Ellaky, Z., & Benabbou, F. (2024). Political social media bot detection: Unveiling cutting-edge feature selection and engineering strategies in machine learning model development. *Scientific African*, 25, e02269.

26. Ellaky, Z., Benabbou, F., & Ouahabi, S. (2023). Systematic literature review of social media bots detection systems. *Journal of King Saud University-Computer and Information Sciences*, 35(5), 101551.
27. Feng, S., Wan, H., Wang, N., Li, J., & Luo, M. (2021, October). Twibot-20: A comprehensive twitter bot detection benchmark. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 4485–4494).
28. Ferrara, E., Chang, H., Chen, E., Muric, G., & Patel, J. (2020). Characterizing social media manipulation in the 2020 U.S. presidential election. *First Monday*, 25(11). <https://doi.org/10.5210/fm.v25i11.11431>
29. García-Orosa, B. (2021). Disinformation, social media, bots, and astroturfing: the fourth wave of digital democracy. *El profesional de la información*, 30(6).
30. Gilani, Z., Kochmar, E., & Crowcroft, J. (2017, July). Classification of twitter accounts into automated agents and human users. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017* (pp. 489–496).
31. Godec, P., Đukić, N., Pretnar, A., Tanko, V., Žagar, L., & Zupan, B. (2021). Explainable Point-Based Document Visualizations. *arXiv preprint arXiv:2110.00462*.
32. Google. (2025-a). *gemma3:1b* [Large language model]. Ollama Library. Retrieved August 27, 2025, from <https://ollama.com/library/gemma3%3A1b>
33. Google. (2025-b). *gemma3:27b* [Large language model]. Ollama Library. Retrieved August 27, 2025, from <https://ollama.com/library/gemma3%3A27b>
34. Google. (n.d.). *Rate limits | Gemini API* [Documentation]. Google AI for Developers. Retrieved December 27, 2025, from <https://ai.google.dev/gemini-api/docs/rate-limits?hl=en>
35. Google Ireland Limited. (2025, August 18). *Information about monthly active recipients under the Digital Services Act (EU)* [Report]. Google Transparency Report. Retrieved from https://storage.googleapis.com/transparencyreport/report-downloads/pdf-report-24_2025-1-1_2025-6-30_en_v1.pdf
36. GrigoryT22. (n.d.). *cross-encoder-ru* [Model]. Hugging Face. Retrieved August 27, 2025, from <https://huggingface.co/GrigoryT22/cross-encoder-ru>
37. Guarino, S., Trino, N., Celestini, A., Chessa, A., & Riotta, G. (2020). Characterizing networks of propaganda on twitter: a case study. *Applied Network Science*, 5(1), 1–22.
38. Hagen, L., Neely, S., Keller, T. E., Scharf, R., & Vasquez, F. E. (2022). Rise of the machines? Examining the influence of social bots on a political discussion network. *Social Science Computer Review*, 40(2), 264–287.

39. Hargittai, E., & Sandvig, C. (Eds.). (2015). *Digital research confidential: The secrets of studying behavior online*. MIT Press.
40. Howard, P. N. & Kollanyi, B. (2017). Bots, #StrongerIn and #Brexit: Computational Propaganda during the UK-EU Referendum. ArXiv:1606.06356 [Physics]. <http://arxiv.org/abs/1606.06356>
41. Hughes, H. C., & Waismel-Manor, I. (2021). The Macedonian fake news industry and the 2016 US election. *PS: Political Science & Politics*, 54(1), 19–23.
42. Júnior, M., Melo, P., da Silva, A. P. C., Benevenuto, F., & Almeida, J. (2021, November). Towards understanding the use of telegram by political groups in brazil. In *Proceedings of the Brazilian Symposium on Multimedia and the Web* (pp. 237–244).
43. Kausar, S., Tahir, B., & Mehmood, M. A. (2022). Push-to-trend: A novel framework to detect trend promoters in trending hashtags. *IEEE Access*, 10, 113005–113017.
44. Keller, F. B., Schoch, D., Stier, S., & Yang, J. (2020). Political astroturfing on twitter: How to coordinate a disinformation campaign. *Political communication*, 37(2), 256–280.
45. Kireev, K., Mykhno, Y., Troncoso, C., & Overdorf, R. (2025). Characterizing and Detecting Propaganda-Spreading Accounts on Telegram. In *34th USENIX Security Symposium (USENIX Security 25)* (pp. 161–180).
46. Kollanyi, B., Howard, P. N., & Woolley, S.C. (2016). Bots and Automation over Twitter during the U.S. Election. Data Memo Oxford, UK: Project on Computational Propaganda
47. Lazer, D. M., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., ... & Zittrain, J. L. (2018). The science of fake news. *Science*, 359(6380), 1094–1096. <https://doi.org/10.1126/science.aao2998>
48. Lee, H., Shin, M., Yang, J., & Chock, T. M. (2025). Virtual influencers vs. human influencers in the context of influencer marketing: The moderating role of machine heuristic on perceived authenticity of influencers. *International Journal of Human-Computer Interaction*, 41(10), 6029–6046.
49. Lefebvre, V. (2015). *Conflicting structures*. Lulu. com.
50. Linvill, D. L., & Warren, P. L. (2020). Troll factories: Manufacturing specialized disinformation on Twitter. *Political Communication*, 37(4), 447–467.
51. Lundberg, E., & Mozelius, P. (2025). The potential effects of deepfakes on news media and entertainment. *AI & SOCIETY*, 40(4), 2159–2170.
52. Marcondes, F. S., Almeida, J. J., & Novais, P. (2024). An exploratory design science research on troll factories. *Integrated Computer-Aided Engineering*, 31(1), 95–115.
53. Mints, A. (2017). Classification of tasks of data mining and data processing in the economy. *Baltic Journal of Economic Studies*, 3(3), 47–52.

54. Mints, O. (2023). Information manipulation and manufactured consensus in social media: detection methods and strategies. *Science and Technology Today*, 13(27), 897–908. [https://doi.org/10.52058/2786-6025-2023-13\(27\)-897-908](https://doi.org/10.52058/2786-6025-2023-13(27)-897-908)
55. Mints, A., & Sidelov, P. (2022). Digital payment card fraud: new vectors and detection. *Digital Technologies in the Contemporary Economy: Collective Monograph/editor Žaneta Simanavičienė*. ISBN 9786094880506.
56. Mints, O. (2025-a) Content labeling based on unstable emoji reactions. In *CSIT Conference 2025, Yerevan, Armenia, September 22-26*, pp. 35–37.
57. Mints, O. (2025-b). Context aware evolution of emoji sentiment reactions in a large Telegram community In *Information Systems Development (ISD2025) Proceedings*. Belgrade, Serbia: University of Gdańsk & University of Belgrade. ISBN: 978-83-972632-1-5. <https://doi.org/10.62036/ISD.2025.93>
58. Mostefai, B., Boutefara, T., Chahinez, A., & Aberkane, M. (2025). Machine Learning Techniques for Identifying Textual Propaganda on Social Media: Development of a Detecting Digital Manipulation System. *ITEGAM-JETIA*, 11(53), 43–49. (TF-IDF+SVM/NN/MNB (Artificial Neural Network with Multi-Layer Perceptron, Support Vector Machine (SVM), and Multinomial Naive Bayes (MNB))
59. Mundra, S., Reddy, J., Mundra, A., Mittal, N., Vidyarthi, A., & Gupta, D. (2023). An automated data-driven machine intelligence framework for mining knowledge to classify fake news using NLP. *ACM Transactions on Asian and Low-Resource Language Information Processing*.
60. Obadimu, A., Mead, E., Hussain, M. N., & Agarwal, N. (2019, June). Identifying toxicity within youtube video comment. In *International conference on social computing, Behavioral-cultural modeling and prediction and behavior representation in modeling and simulation* (pp. 214–223). Cham: Springer International Publishing.
61. Orange Data Mining. (n.d.). *Hierarchical Clustering*. Retrieved August 28, 2025, from <https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/>
62. Policar, P. G., & Zupan, B. (2024). Visualizing highdimensional temporal data using direction-aware t-SNE. CoRR, abs/2403.19040, 5. <http://file.biolab.si/tmp/pavlin/2024-arxiv-temporal-tsne.pdf>
63. Polupanova, K., Mints, O., Simanaviciene, Z., & Miliaieva, Y. (2025). Research into the phenomenon of self-organization of online communities to counter disinformation and manipulation. *Publishing House "Baltija Publishing"*, pp. 40–46.
64. Pote, M. (2024). Computational Propaganda Theory and Bot Detection System: Critical Literature Review. *arXiv preprint arXiv:2404.05240*.
65. Pradhan, R. (2021, November). Extracting sentiments from youtube comments. In *2021 Sixth International Conference on Image Information Processing (ICIIP)* (Vol. 6, pp. 1–4). IEEE.

66. Pratama, P. G., & Rakhmawati, N. A. (2019). Social bot detection on 2019 Indonesia president candidate's supporter's tweets. *Procedia Computer Science*, 161, 813–820.
67. Pushshift (n.a.) *NCRI Reddit Access* Retrieved November 11, 2025, from <https://pushshift.io>
68. Rossi, A., Arenas, M. P., Kocyigit, E., & Hani, M. (2022, June). Challenges of protecting confidentiality in social media data and their ethical import. In *2022 IEEE European symposium on security and privacy workshops (EuroS&PW)* (pp. 554–561). IEEE.
69. Sager, M. A., Kashyap, A. M., Tamminga, M., Ravoori, S., Callison-Burch, C., & Lipoff, J. B. (2021). Identifying and responding to health misinformation on reddit dermatology forums with artificially intelligent bots using natural language processing: design and evaluation study. *JMIR dermatology*, 4(2), e20975.
70. Saif, H., He, Y., & Alani, H. (2012, November). Semantic sentiment analysis of twitter. In *International semantic web conference* (pp. 508–524). Berlin, Heidelberg: Springer Berlin Heidelberg.
71. Sayyadiharikandeh, M., Varol, O., Yang, K. C., Flammini, A., & Menczer, F. (2020, October). Detection of novel social bots by ensembles of specialized classifiers. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 2725–2732).
72. Schler, J., & Bonchek-Dokow, E. (2024). Profiling Astroturfers on Facebook: A Complete Framework for Labeling, Feature Extraction, and Classification. *Machine Learning and Knowledge Extraction*, 6(4), 2183–2200.
73. Schler, J., & Bonchek-Dokow, E. (2024). Profiling Astroturfers on Facebook: A Complete Framework for Labeling, Feature Extraction, and Classification. *Machine Learning and Knowledge Extraction*, 6(4), 2183–2200.
74. Schoch, D., Keller, F. B., Stier, S., & Yang, J. (2022). Coordination patterns reveal online political astroturfing across the world. *Scientific reports*, 12(1), 4572.
75. SEO Sandwich (2025) *Key Reddit Statistics for 2025* <https://seosandwich.com/reddit-stats/>
76. Serrano, R.M. (2024, February) Disinformation on Youtube: research and content moderation policies. *EU DisinfoLab*. https://www.disinfo.eu/wp-content/uploads/2024/02/20240228_YouTube_factsheet.pdf
77. Social Samosa. (2025, September 3). *Instagram surpasses Facebook in EU with 10x faster growth*. Retrieved from <https://www.socialsamosa.com/news-2/instagram-surpasses-facebook-eu-faster-growth-9780765>
78. Sunvideo. (2022). *New Year's Mariupol 2022! Everything is covered in snow! All the Christmas trees and lights of the city! Drama Theater. Aerial photography* [Video]. YouTube. https://www.youtube.com/watch?v=Dz_ACaSlSqE

79. Synowiec, A. (2025). Cognitive processes and emotions in social media: a case study of user-generated responses to political disinformation. *Scientific Papers of Silesian University of Technology. Organization & Management/ Zeszyty Naukowe Politechniki Slaskiej. Seria Organizacji i Zarzadzanie*, (pp. 489–502).
80. Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2), 146–160.
81. Tong, J., Marx, J., Turel, O., & Cui, T. (2024). Combating Deepfake Misinformation on Social Media: A Scoping Review and Research Agenda.
82. Twitter IRA (2018) Twitter Internet Research Agency Dataset. *Twitter*. <https://archive.org/details/twitter-ira>
83. Vaiqoh, I., & Astuti Nurhaeni, I. D. (2024). Optimizing AI Potential in Political Communication: A Study of the 2024 Indonesian Presidential Election. *e-BANGI Journal*, 21(3).
84. Vargas, L., Emami, P., & Traynor, P. (2020, November). On the detection of disinformation campaign activity with network analysis. In *Proceedings of the 2020 ACM SIGSAC conference on cloud computing security workshop* (pp. 133–146).
85. Varol, O., Ferrara, E., Davis, C., Menczer, F., & Flammini, A. (2017, May). Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the international AAAI conference on web and social media* (Vol. 11, No. 1, pp. 280–289).
86. Vasara, A. (2020). Theory of reflexive control: origins, evolution and application in the framework of contemporary Russian military strategy. National Defence University.
87. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
88. Wardle, C., & Derakhshan, H. (2017). *Information disorder: Toward an interdisciplinary framework for research and policymaking* (Vol. 27, pp. 1–107). Strasbourg: Council of Europe.
89. Wasserman, L. (2004). All of statistics: A concise course in statistical inference. New York, NY: Springer.
90. Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., ... & Le, Q. V. (2021). Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652. <https://arxiv.org/pdf/2109.01652>
91. Wilson, T. D., & Maceviciute, E. (2022). Information misbehaviour: modelling the motivations for the creation, acceptance and dissemination of misinformation. *Journal of Documentation*, 78(7), 485–505.
92. Wilson, T., & Starbird, K. (2020). Cross-platform disinformation campaigns: lessons learned and next steps. *Harvard Kennedy School Misinformation Review*, 1(1).

93. Woolley, S. (2023). *Manufacturing consensus: Understanding propaganda in the era of automation and anonymity*. Yale University Press.
94. World Population Review. (n.d.). *Telegram users by country 2026*. Retrieved February 13, 2026, from <https://worldpopulationreview.com/country-rankings/telegram-users-by-country>
95. X Internet Unlimited Company. (2025, April). *DSA transparency report – April 2025* [Report]. X Transparency Center. Retrieved from <https://transparency.x.com/assets/dsa/transparency-report/dsa-transparency-report-april-2025.pdf>
96. Xiao, Z., Zhu, J., Wang, Y., Zhou, P., Lam, W. H., Porter, M. A., & Sun, Y. (2023). Detecting political biases of named entities and hashtags on Twitter. *EPJ Data Science*, 12(1), 20.
97. Zannettou, S., Caulfield, T., De Cristofaro, E., Sirivianos, M., Stringhini, G., & Blackburn, J. (2019, May). Disinformation warfare: Understanding state-sponsored trolls on Twitter and their influence on the web. In *Companion proceedings of the 2019 world wide web conference* (pp. 218–226).

Abstract

The main aim of this study is to develop and test an applied methodology for detecting low-level information manipulation in comment-based communication when direct attribution to coordinated bot networks is difficult. The empirical setting is a YouTube discussion thread attached to a neutral video of Mariupol filmed in December 2021, which allows observation of a comparatively “quiet” baseline and later periods of higher political salience. Using a Python pipeline and the YouTube API, the study compiled a pseudonymized dataset of 3,700 comments posted between 31 December 2021 and 20 August 2025, preserving timestamps, comment depth, and reply-parent links.

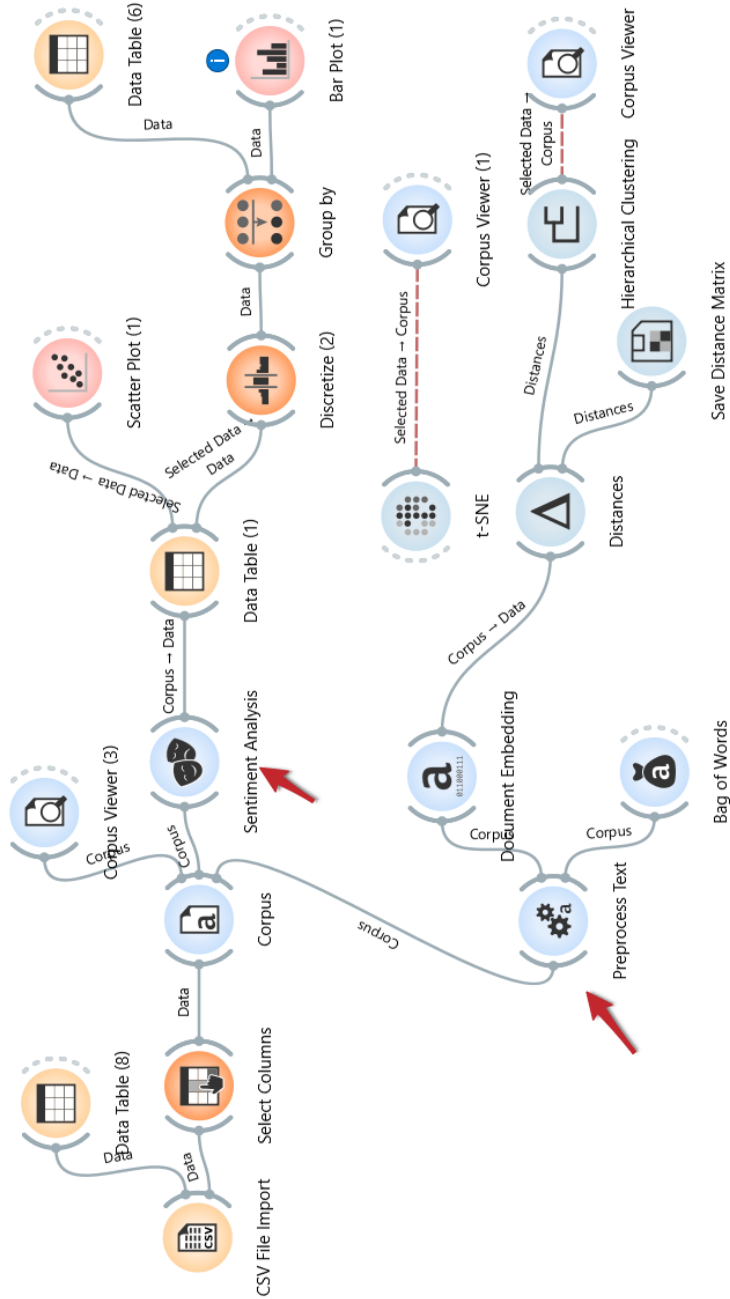
The methodology combines several complementary indicators. First, temporal aggregation is used to detect abnormal concentration of activity and to compare burst dynamics with the long-run baseline. Second, sentiment analysis tracks changes in the emotional background of the discussion over time. Third, semantic similarity and clustering are applied in two representations, Bag-of-Words and Sentence-BERT embeddings, to identify near-duplicate or paraphrased message groups that may reflect coordinated narrative injection. Finally, the study introduces reply relevance degradation as an interaction-aware signal and estimates the likelihood that a reply is meaningfully connected to its parent text. Reply relevance is computed using multiple model families, including compact embedding-based methods and GPT-like LLMs (Gemma3:1b and Gemma3:27b), with monthly aggregation and statistical testing of proportion shifts.

The results reveal a strongly anomalous period in December 2022, when commenting volume sharply concentrates and the mean reply relevance drops markedly relative to early 2022 levels. Statistical tests confirm that the relevance decline is unlikely to be explained by random fluctuation, supporting the interpretation of exogenous influence on conversational coherence. The study concludes that combining activity bursts with relevance degradation and semantic clustering offers a practical, interpretable toolkit for detecting low-level manipulation in YouTube discussions, while also noting limitations related to single-case design, platform data constraints, moderation opacity, and model dependence.

Keywords: *information manipulation, astroturfing, coordinated behavior, YouTube comments, contextual relevance, zero-shot classification, conversational coherence.*

Appendix

1. Data analysis scenario



2. Parsing Youtube comments

```
import os
import googleapiclient.discovery
import csv

DEVELOPER_KEY = «__Key here__»
VIDEO_ID = «__id here__»

def youtube(nextPageToken=None):
    os.environ[«OAUTHLIB_INSECURE_TRANSPORT»] = «1»
    api_service_name = «youtube»
    api_version = «v3»

    youtube = googleapiclient.discovery.build(
        api_service_name, api_version, developerKey = DEVELOPER_KEY)

    request = youtube.commentThreads().list(
        part=»id,snippet«,
        maxResults=100,
        pageToken=nextPageToken,
        videoId=VIDEO_ID
    )
    response = request.execute()
    return response

def youtubechild(NextParentId, nextPageToken=None):
    os.environ[«OAUTHLIB_INSECURE_TRANSPORT»] = «1»

    api_service_name = «youtube»
    api_version = «v3»

    youtube = googleapiclient.discovery.build(
        api_service_name, api_version, developerKey = DEVELOPER_KEY)

    request = youtube.comments().list(
        part=»id,snippet«,
        maxResults=100,
        pageToken=nextPageToken,
        parentId=NextParentId
    )
    response = request.execute()
    return response

def main():
```

```

print('download comments')
response = youtube()
items = response.get('items')
nextPageToken = response.get('nextPageToken')
i=1
while nextPageToken is not None:
    print(str(i*100))
    response = youtube(nextPageToken)
    nextPageToken = response.get('nextPageToken')
    items = items + response.get('items')
    i+=1

print(len(items))

print('download replies')
replies = []
for line in items:
    if line.get('snippet').get('totalReplyCount') > 0:
        print(line.get('snippet').get('totalReplyCount'))
        response = youtubechild(line.get('snippet').
get('topLevelComment').get('id'))
        replies = replies + response.get('items')
        nextPageToken = response.get('nextPageToken')
        i=1
        while nextPageToken is not None:
            response = youtubechild(line.get('snippet').
get('topLevelComment').get('id'), nextPageToken)
            nextPageToken = response.get('nextPageToken')
            replies = replies + response.get('items')
            i+=1

print(len(replies))
print('Open csv file')
with open('youtuberresults.csv', 'w', encoding='utf-8') as csv_
file:
    writer = csv.writer(csv_file, quoting=csv.QUOTE_ALL,
lineterminator='\n')

    row = [
        'etag'
        , 'parentid'
        , 'id'
        , 'textDisplay'
        , 'textOriginal'
    ]

```

```

        , 'authorDisplayName'
        , 'authorProfileImageUrl'
        , 'authorChannelUrl'
        , 'authorChannelId'
        , 'likeCount'
        , 'publishedAt'
        , 'updatedAt'
    ]
    print(«Start write in csv»)
    writer.writerow(row)

    print(«Write comments in csv»)
    for line in items:
        topLevelComment = line.get(«snippet»).get(«topLevelComment»)
        if topLevelComment.get('snippet').get('authorChannelId')
is not None:
            authorChannelId = topLevelComment.get('snippet').
get('authorChannelId').get('value')
            else:
                authorChannelId = ''
            row = [
                topLevelComment.get('etag')
                , topLevelComment.get('id')
                , topLevelComment.get('id')
                , topLevelComment.get('snippet').get('textDisplay')
                , topLevelComment.get('snippet').get('textOriginal')
                , topLevelComment.get('snippet').get('authorDisplayName')
                , topLevelComment.get('snippet').get('authorProfileImageUrl')
                , topLevelComment.get('snippet').get('authorChannelUrl')
                , authorChannelId
                , topLevelComment.get('snippet').get('likeCount')
                , topLevelComment.get('snippet').get('publishedAt')
                , topLevelComment.get('snippet').get('updatedAt')
            ]
            writer.writerow(row)

    print(«Write replies in csv»)
    for line in replies:
        if line.get('snippet').get('authorChannelId') is not None:
            authorChannelId = line.get('snippet').get('authorChannelId').
get('value')
        else:
            authorChannelId = ''
        row = [

```

```
        line.get('etag')
    , line.get('snippet').get('parentId')
    , line.get('id')
    , line.get('snippet').get('textDisplay')
    , line.get('snippet').get('textOriginal')
    , line.get('snippet').get('authorDisplayName')
    , line.get('snippet').get('authorProfileImageUrl')
    , line.get('snippet').get('authorChannelUrl')
    , authorChannelId
    , line.get('snippet').get('likeCount')
    , line.get('snippet').get('publishedAt')
    , line.get('snippet').get('updatedAt')
    ]
    writer.writerow(row)
print(«done»)

if __name__ == «__main__»:
    main()
```

3. Data pseudonymization

```

import csv
import os

def anonymize_youtube_comments(input_filename='name1.csv', output_
filename='name1(anonym).csv'):
    # Dictionary to store the mapping: {'original_name': 'user-1'}
    author_map = {}
    # Counter for generating unique anonymous IDs
    user_counter = 1

    # List to store the processed rows
    processed_rows = []

    try:
        with open(input_filename, mode='r', encoding='utf-8') as infile:
            reader = csv.reader(infile)

            # Read the header and find the index of the required column
            header = next(reader)
            try:
                author_display_name_index = header.index('authorDisplayName')
            except ValueError:
                print(f»Error: Column 'authorDisplayName' not found
in file {input_filename}»)
                return

            processed_rows.append(header)

            # Process the remaining rows
            for row in reader:
                # Check if the row has enough columns
                if len(row) > author_display_name_index:
                    original_name = row[author_display_name_index]

                    # If the name has not been seen before, add it
                    if original_name not in author_map:
                        author_map[original_name] = f'user-{user_
counter}'
                        user_counter += 1

            # Replace the original name with the anonymized one

```

```
        anonymized_name = author_map[original_name]
        row[author_display_name_index] = anonymized_name

    processed_rows.append(row)

    # Write the result to a new file
    with open(output_filename, mode='w', encoding='utf-8',
newline='') as outfile:
        writer = csv.writer(outfile)
        writer.writerows(processed_rows)

    print(f»Anonymization completed successfully. Result saved
to file: {output_filename}»)
    print(f»Total of {len(author_map)} unique users were
anonymized.»)

except FileNotFoundError:
    print(f»Error: File '{input_filename}' not found.»)
except Exception as e:
    print(f»An unexpected error occurred: {e}»)

# --- Script execution ---
anonymize_youtube_comments()
```

4. Selecting similar comments from distance matrix

Cell1 Import data & libraries

```
import pandas as pd
import numpy as np
import time

# --- SETTINGS ---
# Specify the path to your file here
FILE_PATH = «D:/Exchange/!Data/YT-comment2/distance matrix(bag).
xlsx»

# The threshold value for finding distances. You can change this and
re-run Cell 4.
THRESHOLD = 1

# The maximum difference between the indices of the rows and columns
to be checked.
MAX_INDEX_DIFFERENCE = 50

# This cell should be run only once to load the data into memory.
try:
    print(«Loading data from Excel file... This may take a while.»)
    start_time = time.time()

    # Load data from the Excel file into a pandas DataFrame.
    # header=None indicates that there is no header row in the file.
    df = pd.read_excel(FILE_PATH, header=None)

    # Convert the DataFrame to a NumPy array for faster computations.
    distance_matrix = df.to_numpy()

    end_time = time.time()

    rows, cols = distance_matrix.shape
    print(f»Matrix loaded successfully in {end_time - start_time:.2f}
seconds.»)
    print(f»Matrix dimensions: {rows}x{cols}»)

except FileNotFoundError:
    print(f»Error: File '{FILE_PATH}' not found. Please check the
FILE_PATH variable.»)
    distance_matrix = None # Set to None if file is not found
except Exception as e:
```

```
print(f»An error occurred during file loading: {e}»)
distance_matrix = None
```

Cell 2: Analysis Function

```
def analyze_distances(matrix, threshold, max_diff):
    «»»
    Analyzes a pre-loaded distance matrix and returns a list of pairs
    with distances less than the threshold.

    Args:
        matrix (np.array): The NumPy array containing the distance
        matrix.
        threshold (float): The threshold value for filtering distances.
        max_diff (int): The maximum difference between the indices
        (i - j).

    Returns:
        list: A list of dictionaries, where each dictionary represents
        a found pair.
    «»»
    if matrix is None:
        print(«Matrix is not loaded. Cannot perform analysis.»)
        return []

    found_pairs = []
    rows, _ = matrix.shape

    # Iterate through the elements below the main diagonal.
    for i in range(rows):
        # The inner loop starts from a point limited by max_diff.
        start_col = max(0, i - max_diff)

        for j in range(start_col, i):
            distance = matrix[i, j]
            # Check if the value exists (is not NaN) and is less than
            the threshold.
            if pd.notna(distance) and distance < threshold:
                # Store the result as a dictionary in the list.
                # Using 0-based indexing for rows/columns in the result.
                found_pairs.append({
                    «row»: i+1,
                    «column»: j+1,
                    «distance»: distance
```

```
    })
```

```
    return found_pairs
```

Cell 3: Run Analysis and Display Results

```
# --- EXECUTION ---
# Ensure the matrix was loaded before running analysis
if 'distance_matrix' in locals() and distance_matrix is not None:
    print(f»Analyzing matrix with threshold = {THRESHOLD} and max
index difference = {MAX_INDEX_DIFFERENCE}...»)
    start_time = time.time()

    # Call the analysis function with the loaded matrix and current
settings.
    results = analyze_distances(distance_matrix, THRESHOLD, MAX_
INDEX_DIFFERENCE)

    end_time = time.time()
    print(f»Analysis completed in {end_time - start_time:.4f}
seconds.»)

# --- OUTPUT ---
# The results are now stored in the 'results' variable for further
use.
# Now, we print them from memory.

    print(f»\nFound {len(results)} pairs with distance less than
{THRESHOLD}:»)
    if results:
        # Print first 15 results as an example
        for pair in results[:15]:
            print(f»Row: {pair['row']}, Column: {pair['column']},
Distance: {pair['distance']:.2f}»)
            if len(results) > 15:
                print(f»... and {len(results) - 15} more.»)
    else:
        print(«No pairs found matching the criteria.»)
else:
    print(«Please run Cell 2 to load the data first.»)
```

Cell 4 Finding Clusters

```
import networkx as nx
```

```
# 1. Create an empty undirected graph
G = nx.Graph()

# 2. Add edges to the graph from the 'results' list
# Each pair (row, column) is an edge.
for pair in results:
    G.add_edge(pair['row'], pair['column'])

# 3. Find all connected components (the clusters)
# The function returns a generator of sets, where each set is a
cluster.
clusters_generator = nx.connected_components(G)

# 4. Convert the generator to a list of lists for easier viewing
clusters = [list(c) for c in clusters_generator]

# 5. Print the results
print(f»Found {len(clusters)} clusters:\n»)
long_clust_count = 0
for i, cluster in enumerate(clusters):
    # Sort for consistent output
    cluster.sort()
    # printing only clusters with more than 2 elements
    if len(cluster)>2:
        print(f»Cluster {i+1}: {cluster}»)
        long_clust_count+=1
print(«\n», long_clust_count, « clusters contains more than 2
elements»)
```

5. Selected clusters (through finding graph structures in the comments pairs)

1. Matrix bagw (Threshold=5, idx_diff=100)

Cluster 2: [19, 21, 34, 41]+
 Cluster 3: [10, 29, 43, 56, 79, 88, 89, 92, 132, 149, 156, 162, 166, 177, 188, 218, 220, 230, 233, 238, 240, 259, 347, 361, 408, 432, 440, 471, 495, 501, 564, 566, 573, 594, 609, 611, 645, 653, 655, 707, 717, 733, 742, 745, 768, 799, 805, 857, 915, 917, 957, 977, 978, 1012, 1051, 1052, 1063, 1090, 1094, 1100, 1132, 1146, 1151, 1167, 1242, 1280, 1306, 1307]short
 Cluster 12: [1437, 1492, 1538, 1553, 1556, 1573, 1610, 1636, 1640, 1645, 1673, 1681, 1722, 1742]em
 Cluster 16: [1695, 1697, 1698, 1701, 1702, 1703, 1704, 1705, 1706, 1707]+
 Cluster 19: [1876, 1877, 1899, 1928]em
 Cluster 25: [2029, 2100, 2153, 2185]em
 Cluster 31: [2293, 2369, 2406, 2461, 2466, 2481, 2507, 2523, 2528, 2576, 2599, 2645, 2673, 2705, 2726, 2738, 2800, 2804, 2805, 2852, 2951, 2996, 3023, 3026, 3118, 3133, 3150, 3246]em
 Cluster 62: [3526, 3543, 3551, 3573, 3584, 3585, 3594, 3616, 3635, 3654, 3668, 3674, 3682, 3687, 3693, 3700]-

2. Matrix embed (Threshold=10, idx_diff=50)

Cluster 2: [8, 9, 19, 21, 23, 28, 30, 34, 41]+
 Cluster 4: [5, 10, 27, 53, 62, 64]
 Cluster 7: [156, 166, 177, 188, 218, 230, 242, 259, 266, 307, 311, 347, 361, 401, 408]
 Cluster 9: [183, 227, 247, 248, 252]
 Cluster 10: [271, 274, 279, 294, 295]
 Cluster 13: [358, 368, 386, 400, 442, 469, 485]
 Cluster 15: [407, 448, 461]
 Cluster 16: [466, 468, 475]
 Cluster 19: [495, 526, 544, 556, 576, 583, 587, 589, 599, 613]
 Cluster 22: [564, 566, 568, 569, 573]
 Cluster 27: [646, 653, 681, 707, 709, 715, 716, 717, 733, 745, 750, 754, 755, 756, 757, 781, 783]
 Cluster 31: [771, 799, 820]
 Cluster 33: [941, 951, 956, 958, 977, 982, 983, 985, 986, 995]
 Cluster 35: [999, 1046, 1072, 1094]
 Cluster 38: [1090, 1100, 1132, 1146, 1160, 1173]
 Cluster 40: [1087, 1102, 1111, 1128, 1133]
 Cluster 45: [1258, 1261, 1279, 1283, 1284, 1307, 1313, 1328]

Cluster 52: [1380, 1422, 1437, 1460, 1468, 1492, 1507, 1511, 1523, 1540, 1541, 1542, 1551, 1552, 1563, 1576, 1584, 1585, 1599, 1607, 1610, 1615, 1618]
 Cluster 59: [1645, 1681, 1722, 1725, 1742, 1746, 1751, 1754, 1760, 1761, 1762, 1767, 1769, 1770, 1772, 1776, 1781, 1784, 1787, 1805, 1808, 1809, 1815, 1822]
 Cluster 60: [1692, 1695, 1697, 1698, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707]
 Cluster 61: [1777, 1778, 1780]
 Cluster 66: [1859, 1865, 1876, 1877, 1899, 1915, 1928]
 Cluster 71: [1990, 1991, 2007, 2008]
 Cluster 73: [1981, 2023, 2029, 2073]
 Cluster 84: [2225, 2227, 2262, 2263]
 Cluster 88: [2270, 2273, 2292, 2294, 2297, 2299, 2302, 2317, 2321, 2323, 2327, 2328, 2329, 2339, 2341, 2349, 2352, 2360, 2374]
 Cluster 95: [2466, 2482, 2496]
 Cluster 102: [2631, 2650, 2688]
 Cluster 107: [2663, 2697, 2699, 2700, 2713]
 Cluster 111: [2741, 2742, 2744]
 Cluster 112: [2772, 2773, 2816, 2817, 2821, 2822]
 Cluster 116: [2800, 2804, 2805, 2826]
 Cluster 122: [2847, 2869, 2882, 2889, 2893, 2925]
 Cluster 123: [2861, 2879, 2880, 2881, 2900, 2914]
 Cluster 128: [**3056**, 3059, 3063, 3065, 3074, 3077, 3078, 3080]+
 Cluster 129: [3057, 3068, 3096, 3097, 3099, 3102, 3103]-
 Cluster 130: [3066, 3069, 3071, 3072, 3086]+
 Cluster 131: [3060, 3076, 3084]
 Cluster 132: [3090, 3091, 3093, 3094]
 Cluster 134: [3134, 3136, 3139, 3142]
 Cluster 147: [3443, 3458, 3471]
 Cluster 148: [3467, 3498, 3522, 3542, 3544]
 Cluster 150: [3539, 3556, 3587]
 Cluster 152: [3580, 3585, 3626]
 Cluster 153: [3645, 3659, 3674, 3680, 3682, 3687]

3. Matrix embed (Threshold=7.5, idx_diff=100)

Found 87 clusters:

Cluster 2: [8, 9, 19, 21, 28, 30, 34, 41]
 Cluster 3: [156, 177, 188]em
 Cluster 4: [162, 230, 266, 347, 361]em
 Cluster 10: [564, 566, 573]+
 Cluster 12: [653, 707, 717, 733, 745, 799]
 Cluster 16: [1063, 1090, 1100, 1132, 1146]
 Cluster 28: [1507, 1563, 1610]

Cluster 30: [1692, 1695, 1697, 1698, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707]+

Cluster 31: [1777, 1778, 1780]+

Cluster 34: [1876, 1877, 1899, 1915, 1928]

Cluster 38: [1981, 2029, 2100]

Cluster 44: [2225, 2227, 2262, 2263]+

Cluster 54: [2741, 2742, 2744]+

Cluster 55: [2772, 2773, 2821, 2822]+?

Cluster 59: [2800, 2804, 2805]+

Cluster 69: [3059, 3065, 3077, 3078, 3080]+

Cluster 70: [3066, 3069, 3071, 3072, 3086]+

Cluster 71: [3060, 3076, 3084] +

Cluster 87: [3585, 3674, 3682, 3687]

19 clusters contains more than 2 elements

6. Python script for finding parent_text for comments in the dataset

```
# === CONFIGURATION PARAMETERS ===
input_csv = «input.csv»           # input CSV file
output_csv = «output.csv»         # output file path
video_description = « Этот видеоролик показывает мирный предновогодний
Мариуполь в декабре 2021 года. Вскоре в город придет война, будет убито
много людей и разрушено много зданий. Но пока всё красивое и целое, все
живы. Показаны новогодние ёлки, Драмтеатр и другие красивые локации. « #
used as parent_text for top-level comments

# Column names in your dataset:
id_col      = «id»                 # comment ID (may include suffix after «.»)
parent_col  = «parentid»          # root comment ID of the thread
text_col    = «textOriginal»      # comment text
author_col  = «authorDisplayName» # author display name (nickname)

# Behavior flags:
fill_missing_with_description = False # if no parent found, replace with
video description
create_clean_fields           = True  # create   textOriginal_clean   and
parent_text_clean

# =====

import re
import pandas as pd
from IPython.display import display

# --- load data ---
try:
    df = pd.read_csv(input_csv)
except UnicodeDecodeError:
    df = pd.read_csv(input_csv, encoding=»utf-8-sig»)

# --- validation ---
need = {id_col, parent_col, text_col, author_col}
miss = need - set(df.columns)
if miss:
    raise ValueError(f»Missing required columns: {miss}. Available:
{list(df.columns)}»)

# --- prepare helper structures ---
df[id_col]      = df[id_col].astype(str)
```

```

df[parent_col] = df[parent_col].astype(str)

ids_present = set(df[id_col])
id_to_text = df.set_index(id_col)[text_col].to_dict()

# Normalize names for matching
def norm_name(s: str) -> str:
    if not isinstance(s, str):
        return «»
    s = s.strip().lstrip(«@»).casefold()
    s = s.strip(«.,:;!()?[]{}<>\»'“”„`~»)
    s = re.sub(r»\s+», « «, s)
    return s

def norm_compact(s: str) -> str:
    s = norm_name(s)
    s = re.sub(r»[.\-_,:;!()?]\»'`]+», «», s)
    s = s.replace(« «, «»)
    return s

# Detect mention at the beginning of a comment: one or two '@',
# optional spaces, nickname until first delimiter
mention_re = re.compile(r»^s*«{1,2}s*([^\s,:;!()?)\[\]\{\}<>#@]+)»)

def extract_mention_head(text: str):
    if not isinstance(text, str):
        return None
    m = mention_re.match(text)
    return m.group(1) if m else None

# Clean leading mention from text
clean_mention_re = re.compile(
    r»»»»^
    \s*                # optional leading spaces
    @ {1,2}            # one or two '@'
    \s*                # optional spaces
    [^\s,:;!()?)\[\]\{\}<>#@]+ # nickname until first delimiter
    \s*                # optional spaces
    [,:\-- - ]*        # optional punctuation separators
    \s*                # optional spaces after separators
    «»»,
    re.VERBOSE
)

```

```

def clean_leading_mention(s: str) -> str:
    if not isinstance(s, str):
        return s
    return clean_mention_re.sub(«», s, count=1)

# Resolve dotted parent ID: strip suffixes after «.» until a valid id
is found
def resolve_dotted_parent_id(row_id: str) -> str | None:
    candidate = row_id
    while «.» in candidate:
        candidate = candidate.rsplit(«.» , 1)[0]
        if candidate in ids_present:
            return candidate
    return None

parent_text_out = []
resolved_parent_id_out = []
method_used_out = []

# Main loop: FIRST use mention heuristic, THEN official id/parentid
hierarchy
for i, row in df.reset_index(drop=True).iterrows():
    rid = row[id_col]
    pid = row[parent_col]
    text = row[text_col]

    txt = None
    used = None
    resolved_pid = None

    # 0) Top-level → use video description
    if pid == rid:
        txt = video_description
        used = «top_level(description)»
        resolved_pid = None

    # 1) Heuristic: if comment STARTS with @/@@ → look for nearest
    previous comment by that author
    if txt is None:
        mention = extract_mention_head(text)
        if mention:
            target_norm = norm_name(mention)
            target_norm_comp = norm_compact(mention)
            for j in range(i - 1, -1, -1):

```

```

        author_j = df.iloc[j][author_col]
        if (norm_name(author_j) == target_norm) or (norm_
compact(author_j) == target_norm_comp):
            txt = df.iloc[j][text_col]
            used = «mention_backtrack»
            resolved_pid = df.iloc[j][id_col]
            break

# 2) Official hierarchy: dotted parent
if txt is None:
    dotted = resolve_dotted_parent_id(rid)
    if dotted is not None:
        candidate_txt = id_to_text.get(dotted)
        if candidate_txt is not None:
            txt = candidate_txt
            used = «dotted_parent»
            resolved_pid = dotted

# 3) Official hierarchy: root parentid
if txt is None and pid in ids_present:
    candidate_txt = id_to_text.get(pid)
    if candidate_txt is not None:
        txt = candidate_txt
        used = «direct_parentid(root)»
        resolved_pid = pid

# 4) Fallback
if txt is None and fill_missing_with_description:
    txt = video_description
    used = «fallback_description»
    resolved_pid = None

parent_text_out.append(txt)
resolved_parent_id_out.append(resolved_pid)
method_used_out.append(used)

df[«parent_text»] = parent_text_out
df[«_resolved_parent_id»] = resolved_parent_id_out # debug column
for analysis
df[«_parent_method»] = method_used_out # debug column
for analysis

# Additional clean fields without leading mentions
if create_clean_fields:

```

```

    df[«textOriginal_clean»] = df[text_col].apply(clean_leading_
mention)
    df[«parent_text_clean»] = df[«parent_text»].apply(lambda s:
clean_leading_mention(s) if isinstance(s, str) else s)

# Summary
num_rows    = len(df)
num_top     = int((df[parent_col] == df[id_col]).sum())
num_resolve = int(df[«parent_text»].notna().sum())
num_missing = int(df[«parent_text»].isna().sum())
by_method   = df[«_parent_method»].value_counts(dropna=False).to_
dict()

print(f»Total rows: {num_rows}. Top-level comments: {num_top}.»)
print(f»parent_text    resolved:    {num_resolve},    missing:    {num_
missing}.»)
print(«Resolution methods used:», by_method)

# Preview
cols_to_show = [id_col, parent_col, author_col, text_col, «parent_
text», «_parent_method»]
if create_clean_fields:
    cols_to_show += [«textOriginal_clean», «parent_text_clean»]
display(df[cols_to_show].head(12))

# Save result
df.to_csv(output_csv, index=False, encoding=»utf-8-sig»)
print(f»Saved to: {output_csv}»)

```

7. Statistics for the comments' relevancy assessments by Gemma3:27b*

publishedAt	likeCount - Mean	reply_prob - Mean	reply_prob - Count
1	2	3	4
< 22 Jan 18	44,359	0,670513	39
22 Jan 18 - Feb 01	4	0,65	2
22 Mar 29 - Apr 12	69	0,85	3
22 Apr 12 - Apr 26	7,75	0,5625	8
22 Apr 26 - May 10	11,875	0,575	8
22 Jun 07 - Jun 21	13	0,55625	8
22 Jul 19 - Aug 02	430,75	0,525	4
22 Sep 13 - Sep 27	135,667	0,75	3
22 Oct 11 - Oct 25	58	0,63	5
22 Oct 25 - Nov 08	51,0714	0,642857	14
22 Nov 08 - Nov 22	79,1667	0,545	12
22 Nov 22 - Dec 06	184,81	0,669048	21
22 Dec 06 - Dec 20	8,1376	0,476643	2907
22 Dec 20 - 23 Jan 03	1,92338	0,463536	509
23 Jan 03 - Jan 17	3,81818	0,370909	33
23 Jan 17 - Jan 31	1,7	0,3	10
23 Jan 31 - Feb 14	1,44444	0,327778	9
23 Feb 28 - Mar 14	6	0,3	2
23 Mar 28 - Apr 11	15	0,425	2
23 Apr 25 - May 09	6,42857	0,614286	7
23 May 09 - May 23	12,3333	0,6	3
23 May 23 - Jun 06	6,4	0,7	5
23 Jun 20 - Jul 04	4,6	0,5	5
23 Jul 04 - Jul 18	0,333333	0,533333	3
23 Aug 15 - Aug 29	3,5	0,5	2
23 Nov 21 - Dec 05	2,875	0,675	8
23 Dec 05 - Dec 19	1,5	0,4	2

Continuation

1	2	3	4
23 Dec 19 - 24 Jan 02	6,4	0,46	5
24 Jan 30 - Feb 13	3	0,8	3
24 Mar 26 - Apr 09	0	0,35	2
24 Jul 02 - Jul 16	0	0,6	2
24 Dec 03 - Dec 17	7	0,666667	3
24 Dec 17 - Dec 31	2,2	0,52	5
24 Dec 31 - 25 Jan 14	0,888889	0,388889	9
25 Jan 14 - Jan 28	0	0,15	2
25 Feb 25 - Mar 11	0	0,53	5
25 Mar 25 - Apr 08	2,5	0,875	2
25 Jul 15 - Jul 29	0	0,7	3
> 25 Aug 12	0	0,3	2

* Only rows with reply_prob-Count > 1 are included.

Notes

Наукове видання

МІНЦ ОЛЕКСІЙ ЮРІЙОВИЧ

**МЕТОДИ ВІЯВЛЕННЯ МАНІПУЛЯЦІЙ З ІНФОРМАЦІЄЮ
В ЦИФРОВИХ МЕДІА: ВИПАДОК YOUTUBE**

Наукове дослідження

Друкується в авторській редакції

Підписано до друку 10.03.2026

Формат 60x84 1/16. Папір офсетний

Друк цифровий. Ум. друк. арк. 6,98. Обл.-вид. арк. 5,38

Тираж 300 прим. Замовлення № 06-03/7

Відділ реалізації. Тел.: (067) 542-08-01.

E-mail: info@book.sumy.ua

ПФ «Видавництво “Університетська книга”»

40000, м. Суми, площа Покровська, 6

Тел.: (0542) 65-75-85. E-mail: publish@book.sumy.ua

www.book.sumy.ua

Свідоцтво суб'єкта видавничої справи ДК № 7461 від 05.10.2021

Віддруковано на обладнанні

ПФ «Видавництво “Університетська книга”»