Algorithms and techniques for bot detection in social networks

Maksim Kalameyets

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Cotutelle internationale: ITMO University

Présentée et soutenue par

MAKSIM KALAMEYETS

Le 20 octobre 2021

algorithmes et techniques de detection des bots dans les réseaux sociaux

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Sergei SOLOVIEV, Andrey CHECHULIN et Martin STRECKER

Jury

M. Dmitry GAVRA, Rapporteur
M. Costin BADICA, Rapporteur
M. Abdelmalek BENZEKRI, Examinateur
Mme Ileana OBER, Examinatrice
M. Fabio MARTINELLI, Examinateur
M. Dmitry KOMASHINSKY, Examinateur
M. Evgeny DANTSIN, Examinateur
M. Sergei SOLOVIEV, Directeur de thèse

ALGORITHMS AND TECHNIQUES FOR BOT DETECTION IN SOCIAL NETWORKS

Thesis submitted by

Maksim Kalameyets

under the guidance of

Prof. Sergei Soloviev, University Paul Sabatier, France
Martin Strecker, University Paul Sabatier, France
Andrey Chechulin, ITMO University, Russia

in partial fulfilment of the requirements
for the award of the degree of

Doctor of Philosophy in Computer Science

UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

ITMO UNIVERSITY

doctoral school

UNIVERSITY TOULOUSE 3 PAUL SABATIER

October 2021

THESIS CERTIFICATE

This is to certify that the thesis titled **Algorithms and techniques for bot detection in social networks**, submitted by **Maksim Kalameyets**, to the University Toulouse 3 – Paul Sabatier & ITMO University, for the award of the degree of **Doctor of Philosophy in Computer Science**, is a bona fide record of the research work done by him under the supervision of Prof. Sergei Soloviev, Martin Strecker and Andrey Chechulin. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Toulouse, October 2021

ACKNOWLEDGEMENTS

I would like to thank the following people who have helped me undertake this research: my supervisor from Russia Andrey C., for his enthusiasm for the project, for his support in information security, encouragement and patience; my supervisor from France Sergei S., for his support in the development of the theoretical foundations of the thesis and assistance with joint PhD organisation; my supervisor Martin M., for his support during my stay in France, the stimulating questions, and significant assistance in the theoretical part of the study; ITMO University and University Toulouse 3 Paul Sabatier for input throughout this joint PhD programme; St. Petersburg Federal Research Center of the Russian Academy of Sciences for the technical equipment provided for study.

For their contributions to data collection and sharing of knowledge: Anton P., who provided the platform for data collection; Dmitry L., who helped with data collection; Lidia V., who helped with the collection of bots and shared knowledge about them; Olga T., who shared knowledge about machine learning; Didier E. and Amira B., who helped with graph's processing.

And to my parents, who set me off on the road to this PhD.

ABSTRACT

KEYWORDS: Information security, bot detection, social networks analysis, social network attacks.

In this thesis, we propose machine learning techniques to detecting and characterizing malicious bots in social networks. The novelty of these techniques is that only interaction patterns of friends' of analysed accounts are used as the source data to detect bots. The proposed techniques have a number of novel advantages. There is no need to download a large amount of text and media data, which are highly language-dependent. Furthermore, it allows one to detect bots that are hidden by privacy settings or blocked, to detect camouflages bots that mimic real people, to detect a group of bots, and to estimate quality and price of a bot.

In the developed solution, we propose to extract the input data for the analysis in form of a social graphs, using a hierarchical social network model. After, to construct features from this graph we use statistical methods, graph algorithms, and methods that analyze graph embedding. And finally, we make a decision using a random forest model or a neural network. Based on this schema we propose 4 techniques, that allows one to perform the full cycle attack detection pipeline – 2 techniques for bot detection: individual bot detection, and group bot detection; and 2 techniques for characterization of bots: estimation of bot quality, and estimation of bot price.

The thesis also presents experiments that evaluate the proposed solutions on the example of bot detection in VKontakte social network. For this, we developed the software prototype that implements the entire chain of analysis - from data collection to decision making. And in order to train the models, we collected the data about bots with different quality, price and camouflage strategies directly from the bot sellers.

The study showed that using only information about the graphs of friends it is possible to recognize and characterize bots with high efficiency ($AUC - ROC \approx 0.9$). At the same time, the proposed solution is quite resistant to the emergence of new types of bots, and to bots of various types - from automatically generated and hacked accounts to users that perform malicious activity for money.

Contents

List of Tables

List of Figures

ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CIA	Confidentiality Integrity Availability
SMM	Social Media Marketing
Q	Quantile
AFD	Average Friend Distance
AZD	Average Zero Distance
RFC	Random Forest Classifier
NN	Neural Network
AUC-ROC	Area Under the Curve Receiver Operator Characteristics

Chapter 1

Introduction

Bot detection in social networks is one of the most requested security functions from commercial companies and law enforcement agencies. In many cases, bots are used to affect reputation, support unfair competition, spread disinformation, and fraud. In some cases, bot activity can pose a serious threat to the integrity of online communities. Modern electoral interference, the spread of conspiracy theories, and stock market manipulations are carried out using bots.

On the one hand, the demand for such services has led to a significant increase in bots detection studies. There have been many methods developed for bot detection: machine learning, graph-based, statistics-based, and analytical techniques. On the other hand, since security specialists pay much attention to the techniques for creating bots and for recognition of bots actions patterns, bots began to use better-camouflaging methods. Bots try to be like real users and use more and more sophisticated techniques. Bot selling companies can generate social network profiles and write texts with the use of Artificial Intelligence (AI), hack real users or control their accounts through malware, or even hire trained staff to manage bots.

As the result, bot detection specialists have to deal with several problems:

- Bot camouflaging problem. The need to detect a bot, which can be both a program with elements of AI and a hired person performing malicious activities. There is a ranking of bots by quality, for each bot detection methods applied can give different results.
- The problem of linguistic and behavioral universality. Different nations and different ethnic groups within the same nation have their distinctive language features, which makes high-quality universal text-based analysis almost impossible.
- The problem of limiting access to data. Large corporations that own social networks have almost unlimited access to data. However, governments, other private companies, and ordinary users are limited by API requests number on the part of social networks. Therefore, they cannot use a large amount of data that are necessary for bot detection.
- Bad quality training datasets problem. Very often, datasets for training and testing of bot detection techniques are formed by a subjective expert with the use of analytical methods. Thus, the developed bot detection approaches inherit the subjectivity of the bot / non-bot labeling from datasets.
- The problem of detecting a group of bots. It is unclear how to recognize the presence of bots in the community without analyzing each user separately.

These problems conjointly lead to challenges of bot detection quality and bot characterization. Also, low-quality detection for any reason of the above results in underestimation of the threat and false positives from the detection system, which can harm real users when implementing countermeasures.

The main goal of the thesis is to develop a bot detection and characterization approach that can work with a relatively small amount of data and number of API requests regardless of language features, can detect camouflaged bots, can characterize bots, can detect bots either one by one or in a group of accounts, and can provide a sufficient detection quality.

1.1 Overview of our approach

The object of our research is social network profiles of real users and artificial social network profiles that correspond to bots.

To detect bots we used supervised machine learning classification methods that can analyze many attributes of a social network profile and make a decision. We propose 4 techniques for bots detection, which can be carried out together under a unified bot detection pipeline, that is presented in Figure 1.1. The pipeline includes 2 techniques for bot detection: individual bot detection, and group bot detection; and 2 techniques for characterization of bots: estimation of bot quality, and estimation of bot price.

In these techniques, we generate data for analysis from an account using the semantic graph of a social network. In "group bot detection" technique, we extract profiles of group members using the semantic graph. In "individual bot detection" and "estimation of bot quality" techniques, we extract profiles from the friends-list. Based on these profiles we form the distribution of numerical metrics, friends-graph and list of identifiers. Thus, we analyze only numerical data and graph structures that are independent on the language features and other text or media information. These techniques use the same types of feature construction methods, that are the main subject of our research:

1. Feature construction methods that use statistics to construct features from numerical distributions include novel for bot detection Benford's law and Gini-index methods, and computational methods for basic distribution parameters.

2. Feature construction methods use graph algorithms (often referred as Network Science methods) to construct features from graphs. These methods include a wide range of graph algorithms that can be applied for undirected graphs like calculation of centrality measures, community detection methods, clusterization methods, etc.

3. Feature construction methods that use graph embeddings to construct features from the list of identifiers. This is the novel method that forms features based on the location of a profile on the embedding-structure of the whole social network friend graph.

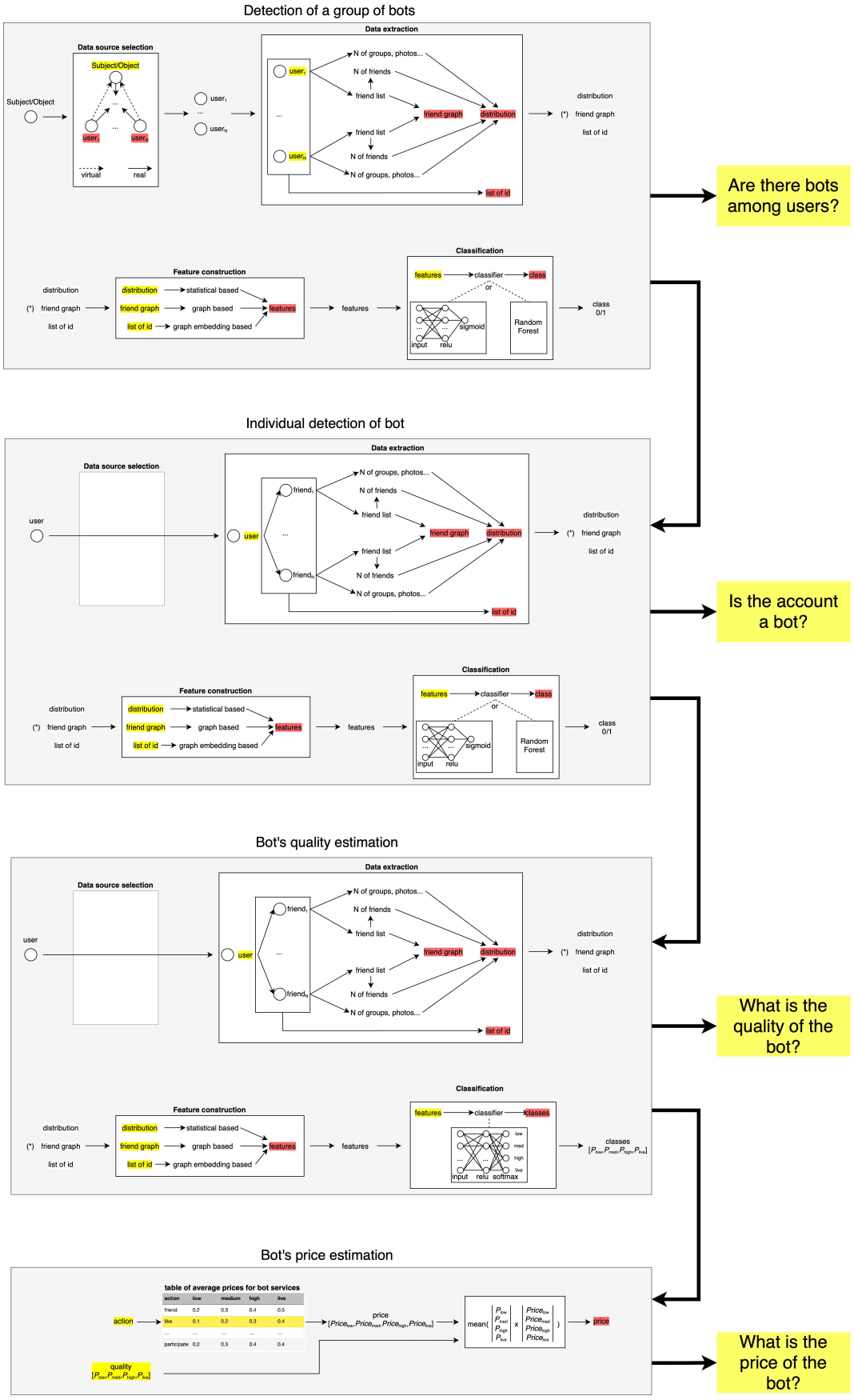

Figure 1.1: Proposed approaches for bot detection in bot detection pipeline.

We also select features based on the multicolinear and informativity analysis to reduce the amount of data needed, by filtering features that correlate with each other and do not correlate with bot labels.

To make a decision in the form "bots in group / no bots" or "a bot / not a bot" or estimate bot quality we use a Random Forest classifier and a Neural Network classifier.

To estimate a bot price, we conduct the market research where we estimate average bot prices. Based on the table of average prices from the market and bot quality we can estimate the bot price.

The proposed approaches were evaluated using datasets that we collected directly from bot trading companies. Therefore, there is no subjectivity in the datasets that we have collected for training. The quality of detecting bots using the proposed methods is at a level, which is sufficient for practical application in information security.

1.2 Contribution

The following results are presented to the defense:

1. technique for adjacent accounts extraction using the semantic graph of social network;
2. technique for feature construction based on the Benford's law, Gini index, graph algorithms and global graph embeddings methods;
3. technique for detection of a group of bots in a social network;
4. technique for individual bot detection in a social network;
5. technique for social network bot characterisation;

The results are combined in the full cycle attack detection pipeline: detecting the target of an attack using detection of a group of bots $\longrightarrow$ detecting bots individually among positive groups $\longrightarrow$ estimating bots quality among positive accounts$\longrightarrow$ estimating price of attack based on the bot quality.

The contribution of the thesis lies in the:

1. bot market analysis and pricing results;
2. classification of bots by bot quality and strategies for bot creation, management and camouflaging;
3. semantic model of a social network;
4. dataset with bots and real users that were collected directly from companies that are trading bots;
5. feature construction technique based on analysis of friend-graph of a social network;
6. feature construction technique based on analysis of graph embedding of a social network;
7. feature construction technique based on the agreement with Benford's law and Gini index;
8. software prototype for bot detection in VKontakte social network.

The scientific novelty of the thesis lies in the:

1. application of a unique combination of methods that together do not depend on text or media information, and also do not depend on the analyzed social network profile;
2. new bot detection technique that can detect bots even if the profile is hidden by privacy settings or even blocked;
3. new detection of a group of bots technique that can detect a group of bots without analysing each profile individually;
4. new feature construction techniques based on the statistical methods: agreement with Benford's law and Gini index;
5. new feature construction techniques based on analysis of graph embeddings of a social network.

The results were presented at a number of conferences and published in scientific journals, the list of which is given at the end of the thesis.

The research results were also partially used in the following projects:

1. Grant of Russian Science Foundation № 18-71-10094 "Monitoring and counteraction to malicious influence in the information space of social networks", 2018-2021
2. Grant of Russian Science Foundation № 18-11-00302 "Intelligent digital network content processing for effective detection and counteraction of inappropriate, dubious and harmful information", 2018-2020

Also, the results of the thesis are used by the Russian state agency "Roskomnadzor" and by the "International Digital Forensics Center" to detect bots on social networks.

Chapter 2

Threats in social networks – scientific context

Social networks have changed the internet and our society. As platforms for information exchange, they have brought the whole world together so much that even a distant event is perceived as something personal. According to the latest polls [LC17, ES21], a significant part of society draws information from social networks, preferring them to television, newspapers, and other classical media. This trend is growing from year to year [LC17] and is not expected to reverse. Along with the popularity, the trust in social networks as a source of information is growing too. This phenomenon can be easily explained by the "all-permeability" and "speed of distribution" of information. Anyone with a smartphone can become a source of information at the click of a camera while disseminating information takes minutes. This is an opportunity to receive news first-hand, bypassing media companies, newspaper editors, and censors.

Social media has influenced our world and the Internet for the better by making it more transparent. But the same information dissemination mechanisms can be used to manipulate opinion, spread misinformation [SCV+17], spread rumors and conspiracy theories [Fer20], create a fake reputation, fraud, and even suppress political competitors [PAC20, FRE+17]. The world has not yet developed universal mechanisms for the dissemination and identification of malicious information on social networks. Damage can be done to anyone who acts on a social network platform: social media, a third-party company, civil society, or government.

At the same time, social networks have a simple, built-in, and self-organizing defense mechanism – institutional reputation, which is based on social network metrics (views, likes, etc.). Low-trust accounts cannot effectively disseminate information. Therefore, to effectively spread misinformation, the attacker either needs to enlist the support of someone with a huge following (influencer) or use bots to distort metrics of social networks.

That's why bot detection on social networks is one of the most requested security functions from commercial companies and law enforcement agencies.

Bot detection is one of the key mechanisms in protecting social networks. In this chapter, we bring the problem of malicious social network bots into context. We describe the types of user interactions in social networks, the different types of bots that one can buy in the market, and the main opposing parties to bots.

2.1 Social networks as information dissemination platforms

Today, the types of social networks are experiencing explosive growth. If earlier, only Facebook could be called a social network, today many services add to themselves the functions of social networks. Some non-obvious examples – online stores that have product reviews and social ratings of sellers, banks that have messages function and friendslists for fast money transferring, news platforms that allow one to subscribe to authors and comment on their work. All of this creates a social relationship that can be attacked: an attacker can understate the rating of a competitor in an online store, can pretend to be someone to fraud, or simply spread rumors and misinformation in comments on a news site. It can be expected that social functions will continue to be implemented in various types of services. Therefore, threats from bots will become more widespread.

Technologically, a social network is a program that provides access to a database through the API. This database stores information about users, the content they generate, and interactions. Both a human and a bot can make requests to the program. Since bots can execute many more requests compared to users, social networks impose limits on the number of API requests. Thus, social network platforms restrict access to information for massive automated requests, but users do not notice this restriction at all. It also helps to reduce the load on the social network infrastructure. On the other hand, this means that protection tools also fall under these restrictions – like bots, they need information from social networks for analysis. Therefore, one of the key features of social networks is API limits, which impose restrictions on access to data during automated uploads.

Also, a key feature of social networks is the presence of social connections between people. Behind all diversity of social networks, there are 3 functions, each of which forms a different structure of social connections:

1. Friendship. By adding friends, a person strives to reproduce their real circle of acquaintances. As a result, many overlapping communities are formed (small world structure). Examples of social networks with such function are Facebook (add friend), Twitter (subscribe) or LinkedIn (connect).

2. Participation. The participation function is needed to indicate a connection of user with some information channels. As a result, a bipartite graph is formed: on the one hand, users, and on the other hand, information channels. An obvious examples are YouTube (users subscribe to video channels), Instagram (users subscribe to photo feed), VKontakte (users participate events).

3. Discussion. In the discussion, users can respond to each other, forming a thread of messages. For example, comments on Facebook, threads of Twits on Twitter, or messages on public chats on Telegram.

These functions are forming graphs as shown on Figure 2.1. Graphs of participation and discussion need to be transformed – because social connection are relationships between people (not between people and content). These social connections are the target of the attacker who is trying to distort them. In addition, such connections carry a lot of information about attacks, including information about bots. There are two key points to keep in mind:

1. Social connections that are formed by these functions are not necessarily public. For example, in messengers (ex. WhatsApp), the *friendship* function is not public – one cannot see another person's list of friends. But one can indirectly establish a user's *subscription* to a public chat by finding this person in the list of chat's participants. And one can establish this person's *discussion's* messages only within one public chat.

2. If friendship displays social connection between users directly, then subscription and discussion displays social connections indirectly (fig. 2.1). For example, we can connect all *subscribers* of one YouTube channel with a fully connected graph of social connections, as shown in the center of Figure 2.1 (the idea belongs to Mark Newman [NWS02]). One can also link the participants in the same *discussion* who are responding to each other, as shown in the right side of Figure 2.1.

Indirect social connections (subscription and discussion) are more difficult to analyze because they provide distorted information. However, for some social networks, the analysis of indirect social connections is the only possible solution (ex. messengers). In this thesis we will focus only on those social networks for which direct social connections (friendship) can be obtained. The bot's key task is to influence users through social connections. To do this, they use various strategies, which we discuss in the next section.

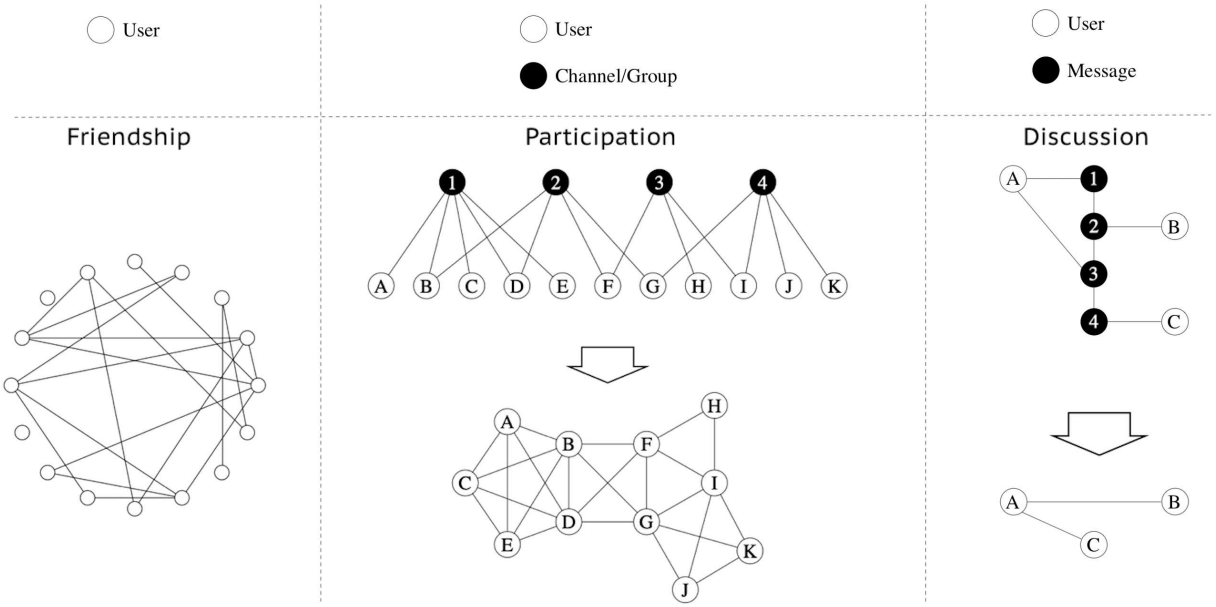

Figure 2.1: Social connections that can be created by friendship, participation and discussion

2.2 Types of social network bots

As part of our research, we consider bots that pose security threats. Of course, not all bots are malicious. For example, bots that provide weather forecasts, generate memes, provide store services and so on are harmless.

In our study, we only consider malicious bots. To do this, we identify a malicious bot via the types of threats. We also analyse bots market – based on the data from the bot market we define different types of bots and common strategies of bot-trading companies.

2.2.1 Bot threats

There are many threats on social media, including password leaks, use of private data by third-party companies, compromise of personal correspondence, etc. Here we focus only on those threats that can be implemented using bots. We distinguish 3 classes of threats and relate them with CIA-triad:

1. Fraud (*threat to confidentiality*) – deceiving social media users to get money or private information. Fraud occurs through correspondence with the user. For example, bots can collect private photos on dating services for blackmail, or the data needed to bypass the bank and mail security systems based on security questions. If such bots have AI and can automatically conduct conversations, this will cause serious damage to tens of thousands of users. Even if the success rate is low, they can fraud a large number of people simply by horizontally scaling the botnet.

2. Promotion of harmful or censored information – propaganda and spread of information that was prohibited by social network platform (hate speech, trolling, etc.) or was prohibited by the government (terrorism, incitement to violence, etc.) Depending on the goal of the attacker:
   (a) *Threat to integrity* – if the goal is to exacerbate the conflict and violate the integrity of the community;
   (b) *Threat to availability* – if the goal is to drown alternative viewpoints with spam.

3. Rating manipulation *threat to integrity* – overestimating the rating to increase user confidence. For this, social network metrics are faked – the number of likes, friends, reviews, etc.

The implementation of these threats by bots requires one key quality from them – *malicious bots pretend to be real people* because it is a key component of user trust.

A person will not believe a fraudster if the account does not look like a real person. Real users will be skeptical about rumors spread by bots. And, of course, the customer will be suspicious of a store that has a lot of positive reviews from bots.

As we will show below, the similarity of a bot with a real person is the main property that the bots management is aimed at. Besides, the similarity of a bot to a real person is one of the main criteria for their pricing.

The complexity of creating bots that look like real people, as well as the difference in strategies for creating and managing them, creates many types of bots. We propose an analysis of the bot market, the understanding of which is necessary for people who are developing tools for detecting them.

2.2.2 Characteristics of malicious bots

Attempts to attribute types to bots have been made earlier. Bot typing has already been done in several papers for bot-detection [OMAAK20]. For example, the paper [GPAT17] described strategies for controlling bots by software, human and hybrid approach. Bots classification by types of threats was presented in [OES19]. The analysis of bot prices is well presented in the report [B+18].

We propose the classification of bots through the market.

To classify bots by type, we analyzed a variety of sites that trade bots. Bot traders provide 2 options – buy an account and manage the bot yourself, or rent an account (buy bot activity) and the bot trader will perform the necessary actions.

We analyzed offers from 7 bot trader companies that provide a bot rental service and 1 forum where bots are sold. The companies were selected from the Russian-speaking segment of the Internet, because Russian bot service providers dominate the social media manipulation market, as highlighted in [B+18]. We analyzed rental offers on 5 platforms: VKontakte, Instagram, Telegram, YouTube, and TikTok. We analyze account sales only for one platform: Vkontakte (because there was relatively little data on other platforms). A total of 1,657 rental offers and 45 sales offers were studied. We also bought 9 offers to analyze bot metrics.

We propose several classification systems that will describe the whole variety of commercial offers.

Each bot trader defines the bot classes differently over multiple parameters.

Bot stores introduced their parameter systems, many of which differed terminologically. For example, one store had a quality scale with terms such as: start, premium+, ultima, etc; and another store: low, high, medium; and the third store: standard, good, best. Some stores determined the speed of bots' actions in quantitative form, other stores – in qualitative, and third stores did not write about speed.

But the bot traders published detailed instructions on their sites on what these or those

characteristics of bots mean. We have aggregated all of these instructions together to develop a unified terminology system.

For buying and renting account parameters are:

1. Activity – is the owner of this account still using it. If yes, then the real owner can quickly notice the suspicious activity that he/she didn't perform. Thus, accounts with positive activity were compromised, but the attacker was unable to change the password.

2. Account age – older accounts are more valuable and are less likely to be blocked than newer ones. Also, people's trust in new accounts is lower as they are relatively easy to register.

3. Binding to a phone number – if the account is not tied to the phone, then the bot is often forced to enter a captcha (what can be a problem for an attacker if the bot is controlled by a program).

4. Location – to which country or region the account is tied and the person of which country it imitates. Bots imitating people from one region may seem suspicious to real users from another region.

5. Owner – is access to the account was obtained by hacking or fraud, or account has no owner. Some bot sellers explicitly point this out in order to demonstrate that their bots are of high quality, as such bots are hacked accounts of real people. Hacking can be carried out through:
   - mail service (if the attacker gained access to mail and requests to restore access to the social network);
   - malware on PC or mobile phone;
   - spoofing through unsecured public networks (for example, public wi-fi in metro, hotels, etc.);
   - hacked databases with passwords (when users use the same passwords across multiple sites);
   - brute force password attack on social networks or mail.

   Fraud involves the usage of social engineering through correspondence with the user.

   No owner – means that the attacker registered this account. Usually, virtual phone numbers are used for this.

6. Number of owners – how many people can buy and use this account at the same time.

For renting account bot traders also provide the next parameters:

1. Quality – an integral ranking of bot that expresses how much a bot looks like a human:
   (a) "Low" – the user can easily recognize the bot. Usually, the profile is empty, with no photos, few friends. It also includes active users who may notice that they have been hacked and the account is being used.
   (b) "Middle" – it is difficult for the user to recognize the bot. Usually the profile not empty, some photos, the average number of friends.
   (c) "High" – bot cannot be distinguished by profile. Usually, these are accounts of real people who have lost or shared access to them (due to hacking or sale). But the bot can be easily recognized by its characteristics that change due to unnatural activity (illogical messages, unusual distribution of friends, etc.).
   (d) "Live" – accounts of real people (hacked and those who act for money). Unnatural activity is the only way to recognize a bot of this type.
2. Type of action – what action the bot needs to perform. We divided the types of activity according to the degree of attracting attention:
   - leaving no digital footprints – ex. views;
   - leaving digital footprints that cannot be seen by visiting the bot page – ex. likes;
   - leaving digital footprints that can be seen by visiting the bot page – ex. friends;
   - leaving a digital footprint of direct user interaction, difficult to implement in automatic mode – ex. comment;
3. Speed of action – how quickly bots can take the required action. For example, how quickly can 100 bots write a comment. Usually measured in the number of activities per day. A spike in activity can trigger social media protection algorithms.

2.2.3 Strategies for management and creation

Different bot stores provide different management strategies. Bots can be controller by:

1. Software – the bot actions are performed automatically by some algorithms. For low, middle, and high quality bots. Usually for actions that can be implemented in automatic mode.
2. Operator – the bot is manually controlled by the operator. For high and live quality bots. Usually for actions that cannot be implemented in automatic mode.
3. Exchange platform – the bot is controlled by a real account owner who agrees to perform some actions for money. For high quality bots. Usually for actions that cannot be implemented in automatic mode.
4. Troll Fabric – agencies employing professionals. The services of such companies are not public – therefore we did not find them. But we believe that they should be included in the list, as they are responsible for many attacks according to a lot of evidence.

Of course, to know exactly the characteristics of the account is possible only after its purchase. Nothing is stopping the bot trader from deceiving you, since this kind of business is not entirely legal (but the purchase of bots is not prohibited, especially for the development of security tools – see the section "Legal and Ethical issues"). Besides, different bot traders understand quality differently. But most of the parameters describe various aspects of the bot's similarity to a real person – quality.

By investigating bots and comparing them to real people on social media, we've identified the main ways bots try to disguise themselves as real people:

1. Use privacy settings. An attacker can fill in just a few fields (profile photo, name, and surname) and hide the rest with privacy settings. Since many users prefer to hide pages on social networks, this will not raise suspicion. This technique is used by a wide variety of bots – from low to high quality. According to our observations, live bots use privacy settings much less often.

2. Use account of a real person. An attacker can use a real person's account by hacking it or buying/renting it. Bot have live quality if a person is an inactive and low quality if active.

3. Generate profile. A difficult task that can include 3 techniques:
   - An attacker can try to generate profile fields. To do this, an attacker can fill an account with photos from another user who has this data open and randomly generate numeric and string parameters.
   - Attacker can generate photos and text content using neural networks. This approach allows the bot to pass the duplicate check (when we are looking for another account with the same photos). A neural network for writing text allows attackers to automate bots that work in chats. Depending on the filling of the account can vary from the middle to live quality.
   - Attacker can generate friendslist graph structure. Real users add people they already know to their friends, forming a small world (see Fig 2.2). "Friends" relation has to be approved by the two friends, thus, likelihood that a real person will approve friendship with an unfamiliar account is small. Therefore, it is difficult for a bot to form a list of friends from real users where everyone is connected. He can add random users – then they will be less likely to form at least some connected graph, or try to form the structure of the friendslist from other bots.

2.2.4 Pricing analysis

We carried out a pricing analysis and checked some correlations of parameters. The purpose of this analysis is to see how price, quality, and type of bot action correlate for various social networks. If the action of the bot correlates with the quality, then the action can be used to assess the quality of the bot. Similarly, if the quality of the bot correlates with the price, then the quality of the bot can be used to estimate the coast of attack. This can be

taken into account for the development of bot detection tools and characterization of bots for various social networks.

To do this, we have parsed HTML-pages with bots rental offers from 7 companies, but each company provides several types of bots for several social networks. In each rental offer, we determined:

1. the lot size – bot activity is sold in batches, for example, 1, 100, 1000, etc.;
2. cost per unit – in cases where the price was indicated for a lot, we calculated the price for one bot;
3. social network for which bots are needed – VKontakte, Instagram, Telegram, YouTube, or TikTok.
4. action – view (viewing post, photo, video, etc.), like, poll (cheat votes in a poll), repost, participate (subscribe to channel, group, etc.), friend, alerts (massive alerts for blocking content by a social network), comment.
5. quality — Low, Middle, High, and Live.

To mark up the dataset by action and quality, we looked for keywords in the description of the offer. For example, to include an offer in the low quality class, we looked for the words: "low", "no avatars", "no guarantee", "slow", "possible activity", etc. We did the same for the rest of the quality and action classes.

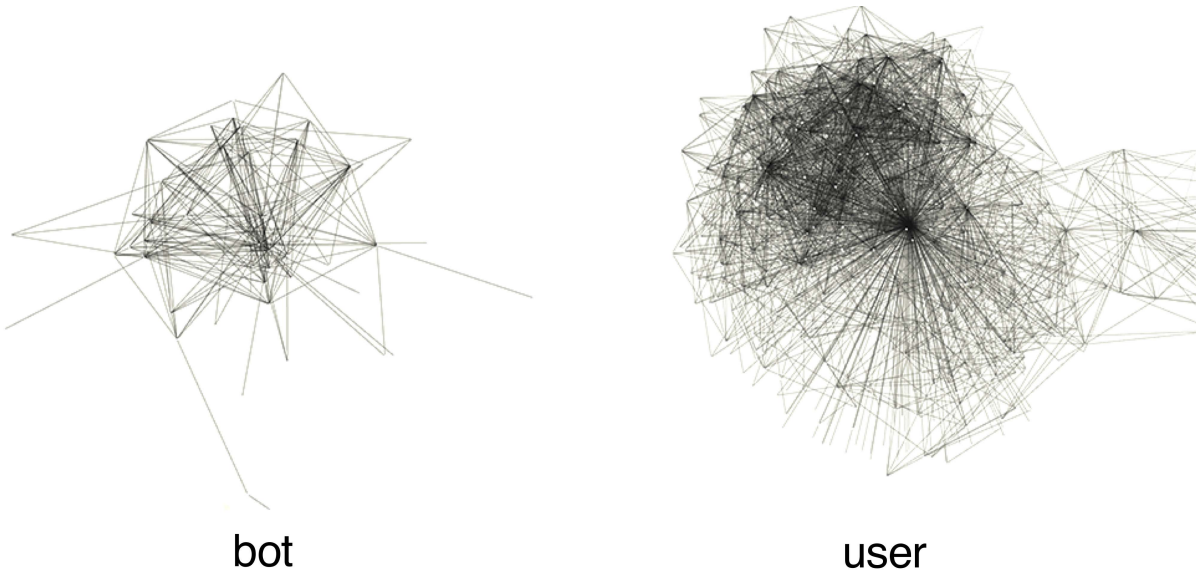

Figure 2.2: The structure of friends is a small world for a real user, and the bot's attempt to simulate a small world

As the result we built 3 charts:

1. Figure 2.3 shows the bubble chart of bot pricing over quality and actions for different social networks. The size of the bubble indicates the price.
2. Figure 2.4 shows the scatter plot of bot pricing over actions for different social networks.
3. Figure 2.5 shows the scatter plot of bot pricing over quality for different social networks.

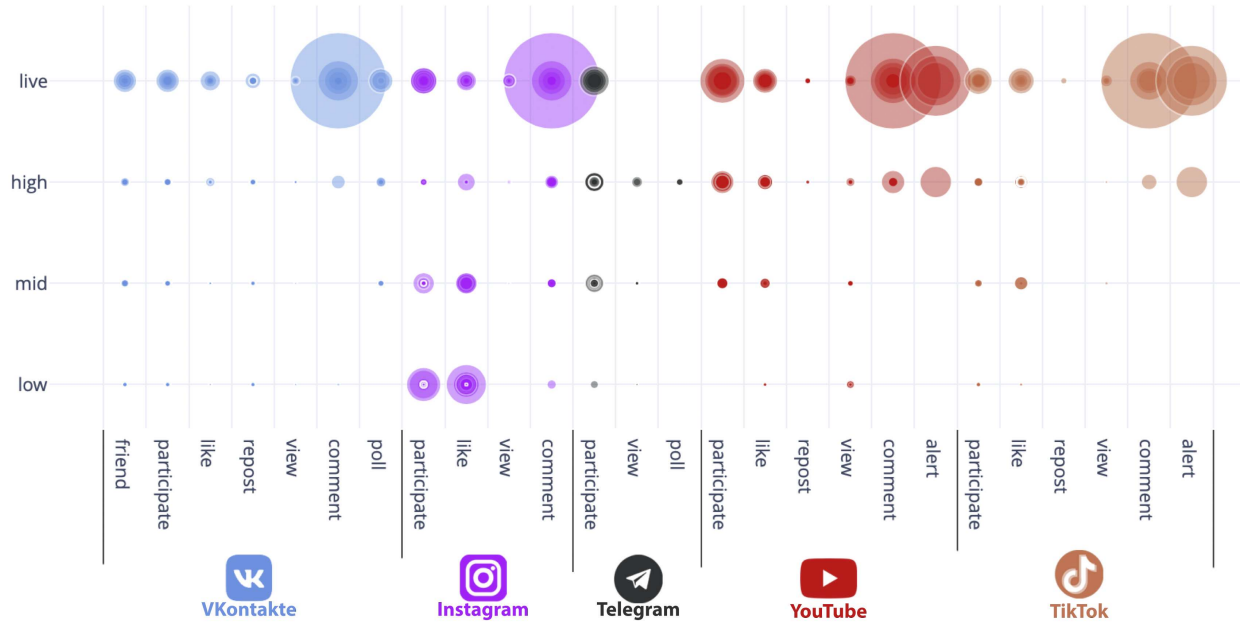

Figure 2.3: Offers for renting bots with a certain quality and a certain action. Price expressed as bubble size.

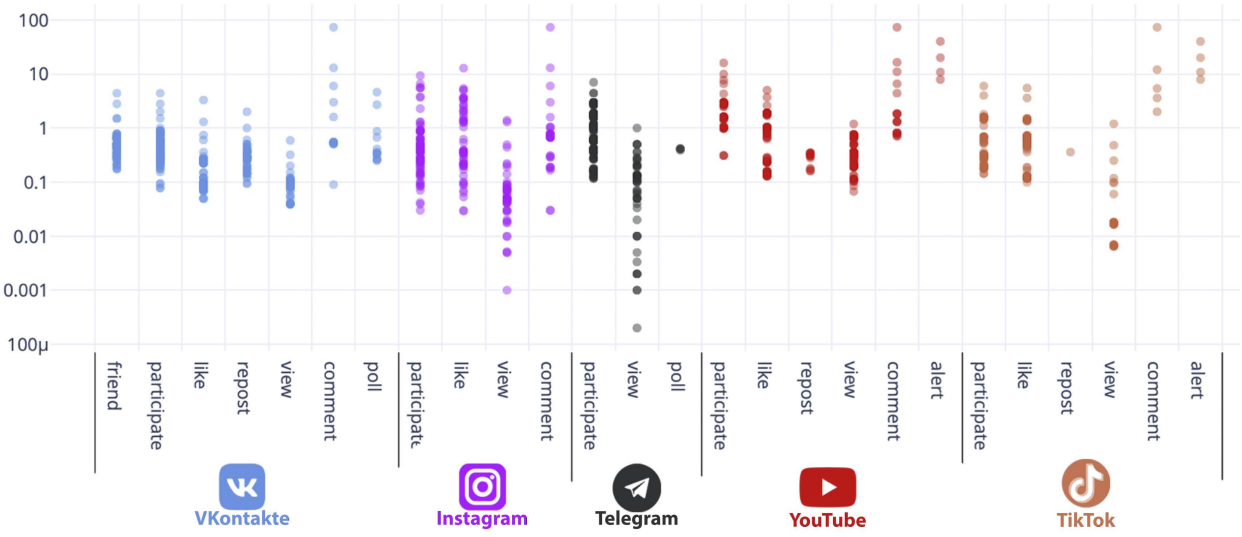

Figure 2.4: Dependence of the action of bots on the price in rubles

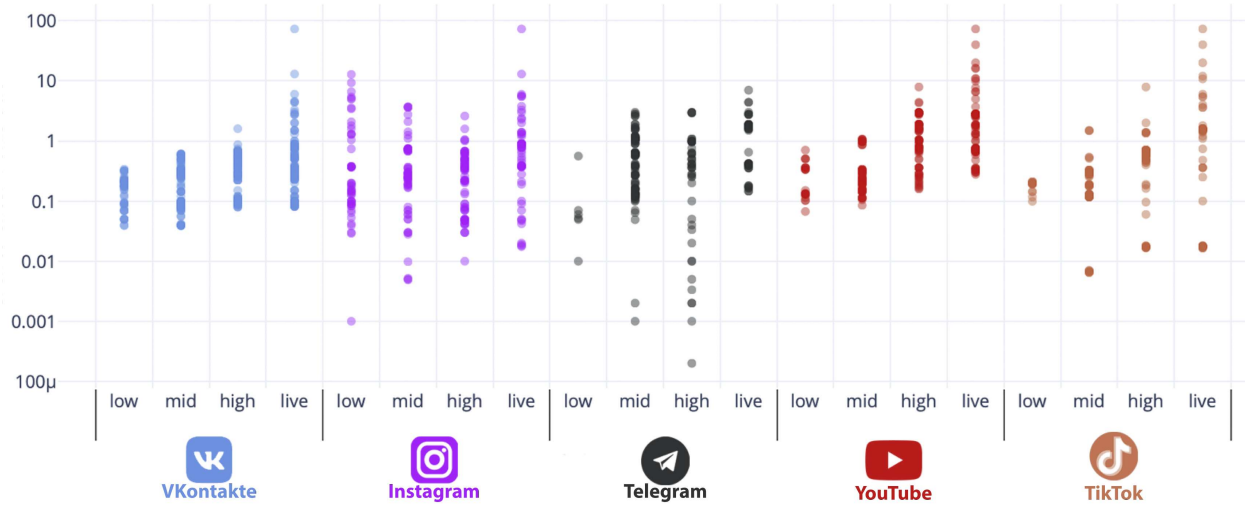

Figure 2.5: Dependence of the quality of bots on the price in rubles

We also perform some correlation analyses.

To do this, we converted qualitative types to quantitative for actions (see Table 2.2) and quality (see Table 2.1). The conversion logic is based on the classifications that were presented before: actions that leave a more visible digital footprint have bigger values, and the quality increases linearly from low to live.

For each group of offers (the same social network, bot trader, quality, and action), we calculated the median price value. This is necessary because the same bot store usually sells services in lots. Thus, these are the same set of bots, and the price difference is due to the discount for buying a large lot. The results of the correlation between these groups of offers are shown in Figure 2.6. We scaled the color scale from 0 to 0.7 (maximum correlation value excluding diagonal) for better perception of results. Correlation shows how social networks' features affect bot trading and bot diversity.

Table 2.1: Bots quality scale.

quality (qualitative score)	**quality (quantitive score)**	**description**
Live	4	accounts of real people
High	3	the bot cannot be distinguished by profile
Middle	2	it is difficult for the user to recognize the bot
Low	1	the user can easily recognize the bot

As expected, for all social networks the price of bots depends on the quality and actions.

Table 2.2: Bots actions scale.

action	**quantitative score**	**action's type (qualitative score)**
comment alert	4	leaving public footprints which is difficult to implement in automatic mode
friend participate repost	3	leaving public footprints that cannot be seen by visiting the bot page
poll like	2	leaving public footprints that can be seen by visiting the bot page
view	1	leaving no public footprints

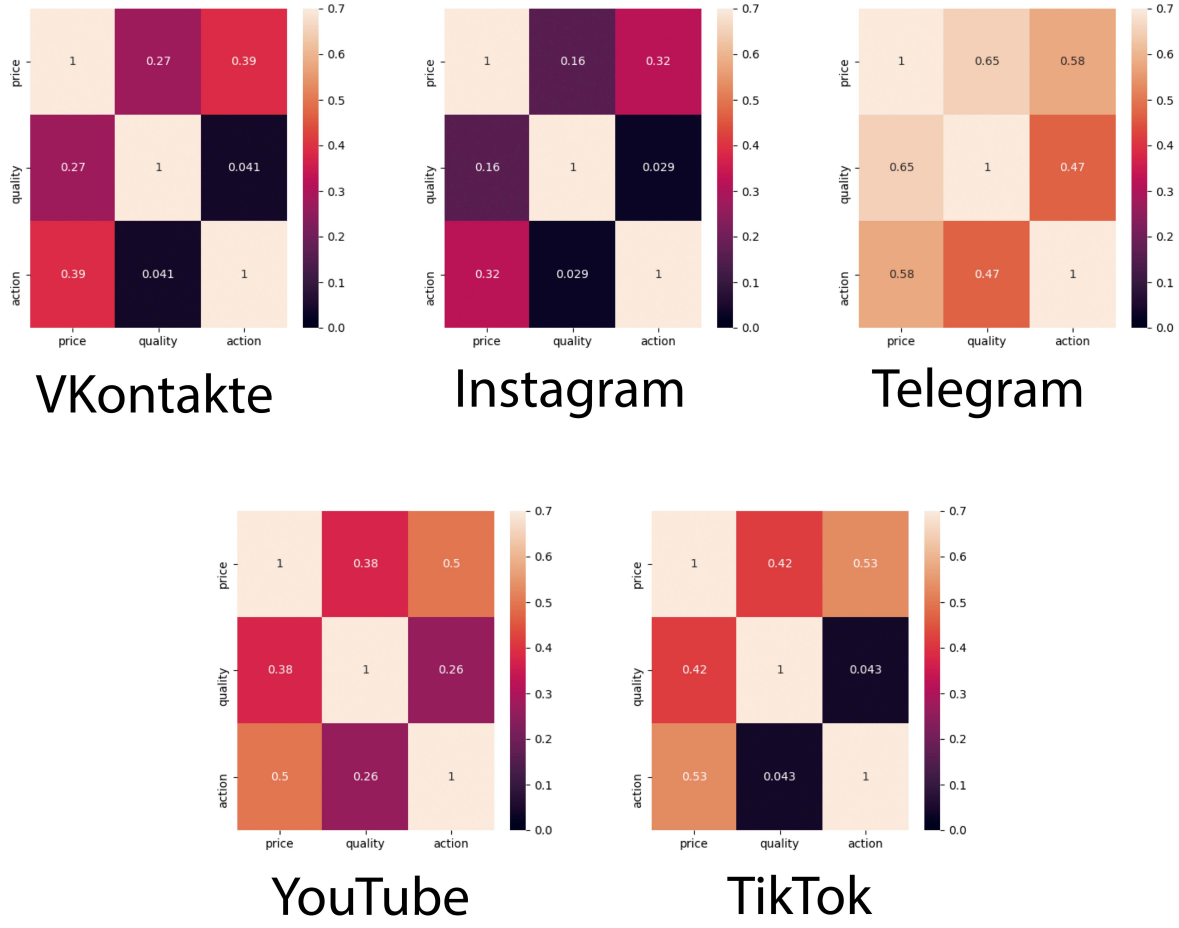

Figure 2.6: Correlation between bot's quantitative quality (Table 2.1), quantitative action (Table 2.2) and price for VKontakte, Instagram, Telegram, YouTube and TikTok social networks

Comments and alerts, as the most complex and attention-grabbing to the bot's profile, have the highest price tag (Fig. 2.3 and Fig. 2.4). Views are the least expensive because they do not leave digital traces and are easily automated.

For all social networks (except for Instagram), there is a dynamics of price growth depending on quality (Fig. 2.5). These dynamics are also noticeable in the correlation results. It can be seen that the dependence of quality on price is different for social networks. We attribute this to the effectiveness of algorithms and measures to combat bots. The more efficient the algorithms on a social network, the more valuable the difference between low and high quality bots becomes.

There is also a noticeable correlation for Telegram and YouTube that better bots are used for more complex actions. For all other social networks, it is almost zero.

For Telegram, this is explained by the fact that Telegram is a messenger, where the main function is participating in chats, discussion, and comments. Therefore, Telegram has a clear demand for human-controlled bots that can write complex text.

For YouTube, this can be explained by changes in the promotion algorithms. YouTube has significantly increased the role of comment activity to promote videos with its recommendation systems. Thus, the demand for human-controlled bots that can write complex text has also increased.

TikTok and Instagram also have a function for writing comments. But in TikTok and Instagram, comments are meant to express emotions, not discussion. Thus, bots can leave emojis or some phrases from the dictionary of sentiments, which low and mid quality bots can also do.

For example, to detect bots on YouTube, it must be taken into account that comments are likely to be written by human-controlled bots. At the same time, on Instagram or TikTok it will be a more mixed group of bots.

Also, the proposed classification of bots allows one to hypothesize which algorithms for feature extraction can provide the most informative features, which features can be resistant to certain bot strategies.

2.3 Opposing parties

Various actors are interested in countering bots. Each of them has its own threats, motivations, resources and limitations. We define 4 classes of main actors: (1) social network companies, (2) civil society, (3) commercial and non-profit companies, (4) and governments.

Social network companies. The main threat to social media is loss of reputation. Risks of this are associated with civil society or government discontent when social networks do not put in enough effort to combat malicious bot activity.

The accusations leveled against social media are quite natural, since social media is an actor who:

1. Has the maximum possible amount of information necessary to detect bots. If a third party needs to download data for analysis, social networks already have the necessary well-structured data about all users. In addition, depending on the privacy policy, social networks have access to a number of unique information that is not available to other actors: data that is hidden by privacy settings (private messages, hidden profile fields, etc.) and data about Internet connections (IP addresses, used browsers, clicks, etc.).
2. Has the flexibility to introduce countermeasures. Social networks are the only actor who can ban, easily hide potentially malicious content from the search results, demand captchas, etc.
3. Has the necessary computing and human resources to counteract harmful influences.

On the other hand, social media have interest in ethical compliance, as their reputation depends on it. For example, unreasonable use of countermeasures can lead to accusations of bias against a certain group of people.

Civil society. Civil society is the main target of attackers. After all, the ultimate goal of spreading misinformation is either fraud or changing people's minds to a brand or to a group of people. Civil society can be characterized as:

1. Has access to public information only. Administrators of online communities and ordinary users can only receive data that has a public status within the framework of the privacy policy.
2. Can only introduce countermeasures within their communities. At the same time, the tools for implementing countermeasures are provided by the social network. For example, community administrators can ban users, but this ban only applies to that community.
3. They do not have the resources to effectively counter malicious activity. Effective counteraction is only possible within small or closed communities.

Unlike social networks, civil society has a greater ethical freedom to use countermeasures – a separate community seeks to ensure the safety of only itself.

Commercial and non-profit companies. As with social media, the main threat to commercial and non-profit companies is loss of reputation. Companies invest a lot of money to build their own brand. Malicious activity can be aimed at discrediting this brand by competitors. Commercial and non-profit companies are actors who:

1. Like users, only have access to public information.
2. Like users, can only take countermeasures within their own community.
3. Like social networks, have enough resources to counter malicious information.

Commercial and non-profit companies strive to build a popular brand that will appeal to many people. Therefore, like social media, they have interest in ethical compliance.

Government. The main risks for states are associated with violation of the law and political risks. The state can demand censoring of certain topics (calls for terrorism, aggression, drug trafficking, etc.). In addition, states can accuse other countries of interfering in internal affairs through social media (usually related to politics). At the same time, the states:

- States have access to public information. But unlike companies and civil society, the state can require social networks to release private information.
- In theory, the only countermeasure the states can impose is to shut down the social network. In practice, the state does not implement countermeasures, but requires the implementation of countermeasures from social networks, civil society and companies.
- Like social networks, states have enough resources to counter malicious information.
- Almost unlimited resources are available to the state.

Unlike other actors, the state is engaged in monitoring compliance with the law and preventing foreign interference. All countermeasures implemented by the state lie outside of social media platforms, but states may require countermeasures to be implemented by other actors.

The identification of the actors is necessary in order to understand who and how will be able to use our solutions.

2.4 Chapter Summary

In this chapter we brought the problem of malicious social network bots into context.

We described the variety of social networks by their differences in types of social connections - friendship, participation and discussion.

We described bot threats and provide bot's classification through the analysis of bots market, which is necessary to develop tools that will taking into account different types of bots.

We described main parties, which is necessary for future discussion – who can use the proposed solutions, and how they can be used.

In the next section, we explore the problems of bot detection and what are the best practices in this area.

Chapter 3

Bot detection challenges

Due to its great relevance, the direction of research on detecting bots is well developed. The main contribution to the development of this area is made by social networks themselves, law enforcement agencies of various governments, research institutes, and private organizations (analytical agencies and associations of journalists). In this chapter, we review the current state of bot detection research. First, it should be noted that bot detection techniques can be divided into two large (and often overlapping) areas: analytical and data-driven.

The analytical approach. Distinctive features of the analytical approach are:

- the large role of the human analyst in decision-making (for example, manual data markup and interpretation);
- non-systematic and non-automated (for example, when data is processed manually and semi-automatically);
- the use of data sources outside the framework of social networks (for example, investigative journalism about companies that provide bot services)
- the subsequent use of countermeasures outside the framework of social networks (for example, the adoption of new laws by states, the reaction of the national security agencies, etc.)

The object of analytical research is not so much a bot, as attacks (in a broad sense) on social networks. An analytical approach results in an investigation and a report on it. Thus, the analytical approach does not pursue the goal of identifying the account of a specific bot. Instead, the goal is to determine the purpose of bots and associate it with any social phenomenon.

The data-driven approach differs from the analytical approach in that it is mostly based on data only. It is the opposite of the analytical approach and its distinguishing features:

- almost complete exclusion of a human from the decision-making process as a factor of subjectivity;
- using automatic methods of working with data;
- using only data that is obtained through social networks
- focuses on countermeasures that can be implemented within the social network.

The object of analysis is the accounts of social network bots. The result can be a label or a probability (bot or not a bot) or some characteristics of bots (word clouds of the content generated by bots, characteristics of the target audience, etc.).

As part of the thesis, we consider the data-driven approach as more objective, automated, and measurable. To do this, we consider its main components: data sources, bots features, and decision-making approaches.

3.1 Available data sources

To detect bots one needs to get the account's features from a social network. The list of features differs from one social network to another.

For example, Facebook, Vkontakte, and Linkedin have a user profile in which the user indicates a lot of information - date of birth, hobbies, place of work, etc. There is no such detailed information on Twitter, Instagram, TikTok, or Telegram. Human connections are not visible in messengers like Telegram or WhatsApp. And on Instagram and TikTok, users do not have a microblog, instead, a set of photos or videos.

To review and classify data sources, we relied on several works devoted to bot detection [DVF+16, OMAAK20, GPAT17, OES19, AAK+17], and especially on the papers [DL18] devoted to bot features on Twitter. We found that it most fully describes the classes of data sources and it is not difficult to interpolate these classes from Twitter to other social networks.

There are 5 big groups of data sources:

1. Account-based – data from account's home page. Home pages vary depending on the social network. They usually include the name, city, hobbies, and other general information that helps describe the user. As numerical values can be used the length of fields, the number of words in a field, numbers of friends / posts / photos / followers / subscriptions / etc., account age, and other values depending on the social network.
2. Adjacent account-based – distribution parameters of account-based, that one can extract from adjacent accounts (accounts that are somehow connected with the analyzed account). This data source is usually referred to simply as friends. But we specifically emphasize – the list of related accounts can be obtained not only through friends. There are many alternative ways to obtain the list of adjacent accounts. The most obvious is to get the lists of followers (for example, on Vkontakte, friends and subscribers are different), accounts that replied to this user's post, family members, etc. More lists of adjacent accounts can be obtained by complex requests, for example: accounts with which users start the discussion; accounts that regularly "liked"; and others.
3. Text and Media – data that one can extract from generated content. The content can include text in posts or comments, videos, photos, polls, emojis, live streams, wishlists, gifts, and so on.

4. Graphs – data that can be obtained by analysis of graphs of accounts (the vertex is account) or graphs of content (the vertex is text/media/etc.). The graphs of accounts can be formed the same ways as adjacent accounts lists (e.g. graph of friends, a graph of accounts that regularly "liked"). Usually, the edge in such graphs is friendship, but it is also possible to use other complex relations of users.

5. Temporal - data on the dynamics of actions performed. These can include time-series of publications, frequency of writing comments, etc.

Table 3.1 provides a comparison of several social networks with a description of the data sources and their availability. This table is not complete but reflects the variety of data for various social networks.

Vkontakte is the most classic type of social network. It is analogous to Facebook and Linkedin. Due to a large number of functions, this type of social network has the largest number of data sources.

Another type of social network is Instagram. Twitter is an analog. A distinctive feature is a limited description of the profile, from which it is almost impossible to extract data. Nevertheless, a clear element of social relations persists - subscriptions, which can be interpreted as friendship.

Telegram is a typical messenger (the same as WhatsApp). It has almost no profile and direct social relationships. But one can restore relations through chats and comments, but such relationships will not be as strong as on Vkontakte or Instagram.

YouTube and TikTok are typical video services. They don't have profiles (or they do, but users almost never fill them out). Social relationships can only be established through a channel subscription relationship or a comment.

At the same time, for all social networks, the variety of graphs and temporal data depend on the amount of data from adjacent accounts and text and media, respectively.

Social network	**Account-based**	**Adjacent account based**	**Text and Media**	**Graphs**	**Temporal**
VKontakte	birthdate, city, occupation, education, social media, politics, religion, etc. overall more than 20 fields	friends, likes, reposts, comments, groups, events, family, classmates, etc. communities	posts, comments, photos, videos, music, gifts	formed from adjacent account	formed from text and media
Instagram	name, website, short bio	subscriptions, likes, reposts, comments	photos, videos, comments	formed from adjacent account	formed from text and media
Telegram	name, phone, short bio	participants in chats, discussions in chats	photos, videos, comments	formed from adjacent account	formed from text and media
YouTube	-	participants in channels, comments	videos, comments	formed from adjacent account	formed from text and media
TikTok	name, social media, short bio	participants in channels, comments, likes	videos, comments	formed from adjacent account	formed from text and media

Table 3.1: Example of available data sources for social networks.

Another aspect that needs to be noted is the availability of the data from the actor's point of view. We have already mentioned that different types of data are available to different actors on social networks. This is correlated with the bot detection approaches classification presented in [DL18]. The book [DL18] discusses 3 concepts to detecting bots:

1. Holistic – when one can get almost complete access to the data. In fact, only social networks and (in some cases) national security agencies can get such access.
2. Pairwise account comparison – when comparing groups of accounts and looking for anomalies in their behavior that are not typical of natural processes. This approach requires extensive use of continuous temporal data, which implies long-term or medium-term account monitoring. Collecting such data can only be done by commercial companies that have sufficient computing power for this.
3. Egocentric – when one takes a snapshot of the account at one point in time and use this snapshot for analysis. This approach is the least demanding and is available even to ordinary users. If we talk about temporal data, then the difference from *pairwise account comparison* is that the temporal data is represented by several snapshots.

This thesis and the data sources discussed in this section refer to the egocentric approach of analysis – we are looking at detection methods that analyze bots at a specific point in time.

3.2 Feature construction approaches

Another important aspect is feature construction from data. As we said, we have 5 data sources from which we can construct features. Bots are identified based on these features, so researchers are constantly looking for new methods to construct them. It is difficult to get features that will effectively indicate bots due to the masking of bots.

It is possible to use 4 "families" of methods for feature construction. We have compiled this classification mainly based on papers [AAK+17, DL18]. We will also briefly review some examples of features that can be obtained using these methods.

1. Statistical - methods [CWZ+16, KMY+20, GKL08, KA13, AC18, JCZ+16, DL18] that are based on searching for features in distributions. They are most common for feature construction since a very large amount of data is represented by distributions (distribution of parameters of friends, text, etc.).

   The simplest example of how statistical methods can help in identifying bots is shown in the Figure 3.1. These statistics were obtained from one of the low-quality bot datasets (which we will analyze later). It shows that after reaching a certain number of groups, bots stop joining groups. Such obvious anomalies are very easily detected by basic statistical methods.

   Statistical techniques can be applied to all types of data sources that can be presented in numerical form.

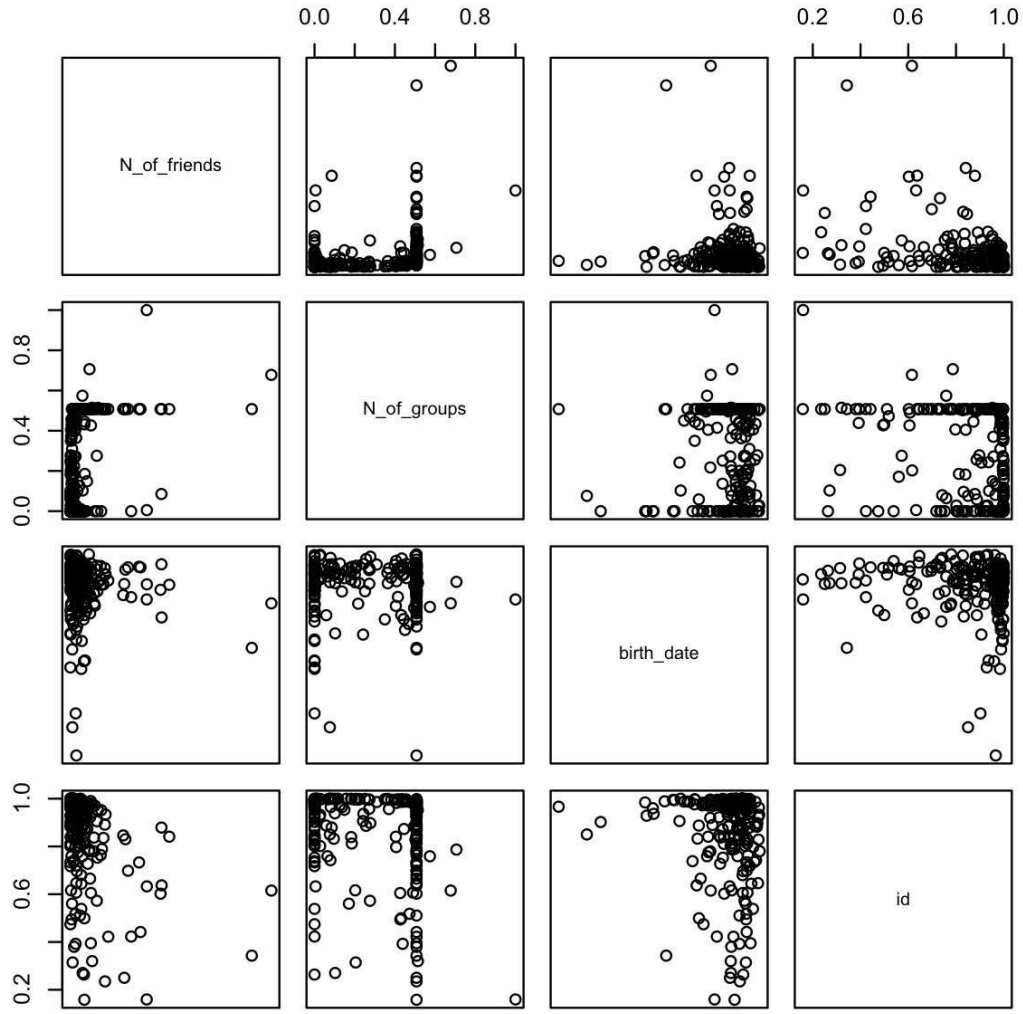

Figure 3.1: An example of a statistical anomaly – when many accounts have the same number of groups.

2. Network science - methods [New18, GKL08, UKBM11, AAK+17] that are based on the analysis of graph structures that can be formed by user or content of the social network. Network science algorithms can be applied for various types of graphs to calculate centrality measures, graph coefficients and other graph metrics.

   Graphs can be a very valuable source of bot features. For example, bots usually trying to imitate friend graph structure (see. Figure 2.2), and for this, some bot-sellers use the strategy of adding other bots as friends. Therefore, if one analyze their graph of friends, it will seem natural. But when a coordinated bot attack occurs, this strategy, on the contrary, exposes the bots. For example, the left side of Figure 3.2 shows the graph of real users' friends in the form of the density of vertices. The closer the vertices, the more likely these people are friends, so the groups of friends will form dense clusters. These users gave likes to posts on the public page and for the most part these users where random, and the likelihood of a friendly connection between them is small. Whereas bots that add other bots as friends formed a very coherent graph (right side of Figure 3.2). In this case, graph metrics (for example, clustering coefficient) will be a valuable feature for bot detection.

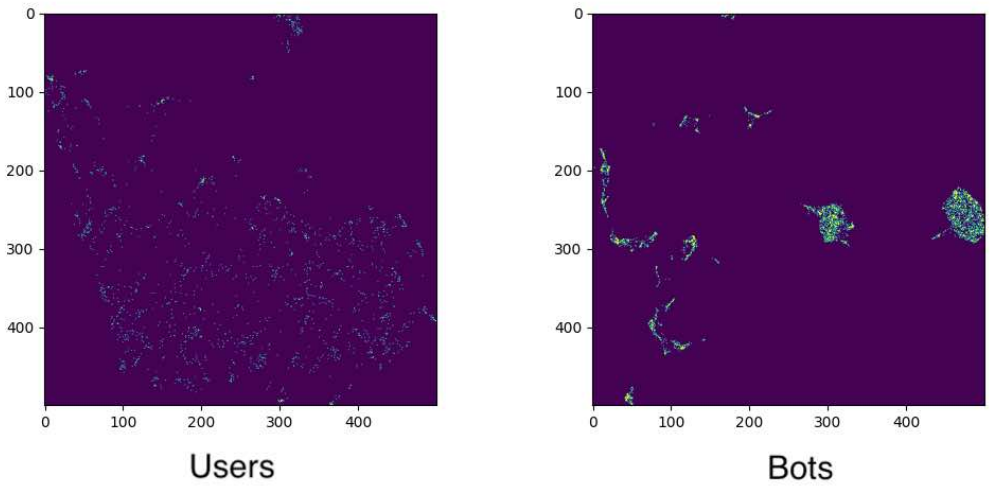

Figure 3.2: An example of coordinated attacks of bots that added other bots as friends.

3. Analytical - various methods of manual mapping of bots. It can be a mapping of users and bots by viewing of user's profile page, visual analysis of graph structures that are formed by sets of users, manual text markup, and similar manual techniques. These methods cannot provide high accuracy and are very expensive. Also, it is difficult to justify decisions based on analytical techniques: often, the criteria that are used in such methods are very subjective.

   However, social media companies have a larger arsenal of data than others. So they can reasonably detect bots by manual analysis of user HTTP requests, used IP addresses, and other information inaccessible to a wide audience.

   In this thesis, we will not use this method, but we consider it important to emphasize its presence.

4. Machine learning methods are used very often, because on social networks a lot of data is presented in non-numerical form - texts, images, videos, audios, etc. The machine learning approach to feature extraction can be broken down into several more approaches:
   - Video, image, or audio recognition. Analyzing media information can be very valuable because it is extremely difficult for bots to generate.

     For example, if a bot account is created from scratch, the bot needs to take profile photos somewhere. If a bot steals a photo from another user [OES19], one can find the original profile using a facial recognition service [SZ21, aut21]. We will give a simple example of detecting a bot using the search4faces [aut21]. In the left of Figure 3.3 is a profile of a real person, and in the right is the search result that showed that someone else was impersonating this person.

     In addition, methods [MS20, NMM+19, HLZ18] are being actively developed for detecting photographs that were generated [Wan21] by neural networks.

     One can also use ready-made solutions [Goo21] to determine what is shown in the photo and video. Object types can also be used as features.

     Audio messages can be converted into text [KB17] and analyzed using methods for text analysis.

- Natural language processing is used for text analysis. There are many articles devoted to text-based detection, largely because the most popular social network for bot researchers is Twitter (and there is almost nothing on Twitter except text).

  Feature extraction based on the text analysis can be various. For example, one can try to recreate numerical features of a user's profile (such as gender, age, and so on) based on the written text [HJU20].

  Another popular approach [WN19, AAEA18, PdPLS19] is to use word embeddings - vector representations of text (typical solutions are word2vec algorithm [MCCD13] and pretrained GloVe word vectors [PSM14]).

  Another well developed approach is to use sentiments [DKS14, HJ20, DVF+16] (different characteristics of emotions and opinion) as features.

- Learning on graphs. As with text, graph structures can also be converted to vector space. For this, algorithms such as Node2Vec [GL16] and DeepWalk [PARS14] are used. By transforming the structure of the graph into a vector, it can be used as a set of features [KG20, STZ18].

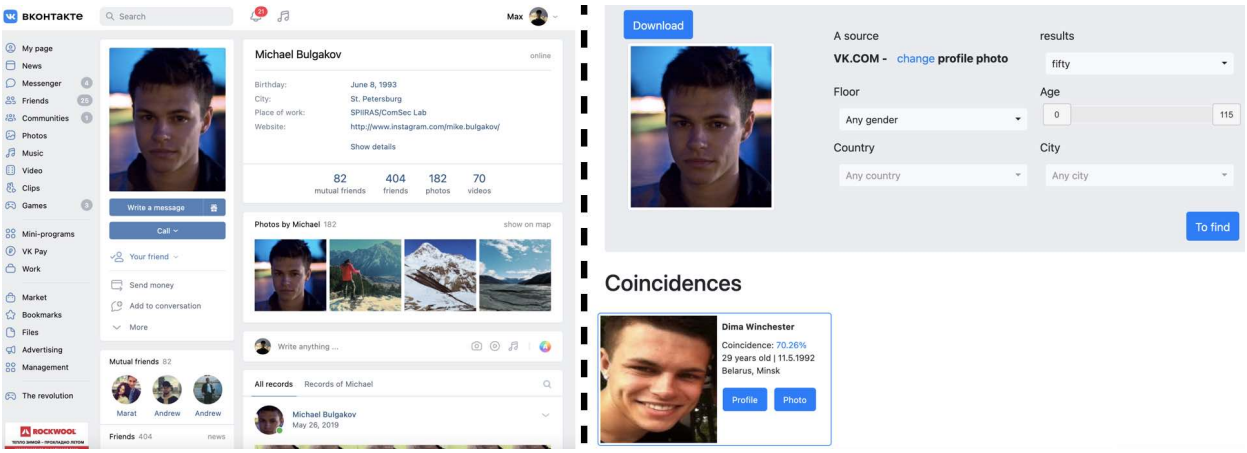

Figure 3.3: An example of a bot "Dima Winchester" (on the right side) that stole a real user's photo (on the left side) and that was exposed using search4faces face recognition service.

3.3 Decision making

A very good taxonomy of decision-making methods has been proposed in [AAK+17, OMAAK20]. But in fact, the conclusion that can be drawn from it is that the most developed approach is machine learning (if we discard the analytical approach and consider only the data-driven approach to identifying bots.) This can be easily explained since machine learning approaches give the best results. This is also confirmed in [AAK+17], which states that 68 percent of journal publications on bot detection use machine learning. Therefore, we will only consider the part of the taxonomy that is devoted to machine learning.

The three main classes of detection algorithms are:

- Supervised learning. These algorithms allow training classifiers on pre-labeled data. Usually, researchers try to solve a binary classification problem when it is necessary to distinguish a bot from a real user (this is the most common task [DL18, OMAAK20, AAK+17]). But it can also be a multi-class classification task, for example, when one goes to detect users and several types of bots [SAD+16].
- Semi-supervised learning. Algorithms that train machine learning models using datasets that have only a small labeled part. This approach is very promising because the size and quality of data labeling is almost the main problem for researchers. For example, in [DAD18, SZC19] is used semi-supervised classification techniques for bot detection.
- Unsupervised learning. Algorithms that try to cluster accounts so they don't need labeled data. We find such an approach interesting because it allows one to work with a group of accounts at once and not separately. Moreover, such methods are not related to the quality of the datasets. A good example of unsupervised learning based bot detection is DeBot [CHM16] and [MCKM17]. However, it is unclear whether such approaches can detect more sophisticated bots and recognize their quality.

Which particular algorithm is the best to use is difficult to predict. But as follows from the reviews [OMAAK20], the most popular is Random Forest. Other methods include AdaBoost, Supported Vector Machine, K-Nearest Neighbor Search, and various types of neural networks (Long Short-Term Memory, Convolutional Neural Networks, Back Propagation Neural Networks).

Nevertheless, to a greater extent, the efficiency of algorithms depends more on the features used, the quality of the dataset, and the problem being solved. Therefore, researchers usually test several algorithms at once, choosing the one that gives the best result.

3.4 Bot detection problems

The three main components of bot detection are a selection of data sources (feature collection), features construction and selection, and selection of a decision-making method.

Nevertheless, there are many obstacles in the way of using bot detection methods that will prevent the approach from being applied, even if it has very high accuracy. These are the problems that researchers face when trying to develop methods for detecting bots. Some of them are hardly mentioned by researchers, but we consider them critically important: (1) API limitation problem, (2) Bot camouflaging problem, (3) Detection efficiency problem, (4) Bad training datasets problem, and (5) Group detection problem.

3.4.1 API limitation problem

We have already mentioned that several groups of interests (actors) operate at once on social networks: government, companies, civil society, and social networks. The key problem for them is not so much accuracy of bot detection as the availability of resources - the ability or inability to use one or another detection method.

In this case, the bottleneck is that social media limit the number of API requests.

Some methods require a lot of information to construct features. For example, if one needs to use posts from a microblog for analysis, downloading these posts for only one account may require thousands of API requests. Vkontakte provides data only for a few API requests, which vary from 1000 to 5000. And for example, TikTok does not have an open API at all.

Social networks have full access to data. Government security agencies and large companies can negotiate with social media exceptional rights to API limits. At this time, small companies have to breed their own bots that collect data, just to get around the API limits. And ordinary users do not have the ability to download such large amounts of data at all. Therefore, methods based on holistic and pairwise account comparison approaches are completely inaccessible for them.

On the other hand, one can understand social networks that seek to reduce the load on their services. And for them, bulk loading of data about bots is most likely no different from a DDoS attack. Therefore, social media can offer various options, such as including access to unlimited API for money or increasing the API limits for scientific research.

Scientific papers do not pay attention to the problem of the amount of data that is required to detect bots. However, this is the main problem that prevents common users from using most solutions.

3.4.2 Bot camouflaging problem

We mentioned that bots are of different quality, which affects both the cost of the attack and the ability to detect bots. More sophisticated bots use more sophisticated camouflage techniques, while the simplest ones can be detected manually by looking closely at the profile. We have already given an example of camouflage in the picture 2.2 when a bot tries to imitate the structure of friends.

Researchers usually don't study how their tools work for bots of varying quality. Besides, a bot is defined as a binary trait, although there can be a huge difference between different types of bots due to different strategies for forming a profile.

In most datasets for training, there is also no information about the quality of bots, which does not allow assessing the balancing of the dataset. It is also impossible to test the robustness of detection tools for recognizing bots with new strategies on such datasets.

We believe that the key problem is not only the recognition of bots but also the recognition of their quality, as well as the ability to describe strategies for their creation / management / profile filling.

Based on quality information, many conclusions can be drawn by security specialists - from countermeasures selection to calculating the cost of attack.

3.4.3 Detection efficiency problem

This is the main problem facing researchers. This is what researchers usually try to improve. The efficiency of bot recognition depends on many things: the dataset, selected data sources, selected feature extraction methods, selected model, optimization of machine learning model parameters, and interpretation of results. Typically, a validation sample is drawn to test the effectiveness.

Bot detection efficiency is very important because the set of countermeasures will depend on the efficiency indicators. For example, for detection tolls with a high false-positive rate, only "soft" countermeasures can be applied (Turing tests ex. captcha, the requirement to bind a phone number, reduction of API limits, etc.). For instruments with a low false-positive rate - more drastic countermeasures (account blocking and freezing).

Also, detection tools should have high efficiency indicators, since there are much more real users on the social network than bots. At the same time, their ratio is unknown and, as a consequence, the effectiveness can be established only indirectly (since we do not know the ratio of bots to users: neither in the validation set nor in the real conditions of using the bot detection tool).

3.4.4 Bad training datasets problem

Usually, for training models, datasets are used that are provided by a social network or that were formed by researchers. At the same time, we have come across three types of datasets.

Datasets that were formed by manual analysis. There are a lot of questions about the quality of such datasets. First, it is generally unclear whether a person can effectively recognize a bot that has used camouflage methods (despite the fact that camouflage methods can have different effectiveness). Secondly, such datasets have a clear element of subjectivity. The simplest example is that an operator may incorrectly place a label due to bias or fatigue. Thirdly, when learning on such methods, machine learning methods tend not to recognize

bots, but to reproduce the logic of the operator.

Datasets that were formed by automatic analysis. Such datasets are formed by ready-made protection tools. For example, a social network can provide a dataset with bots that have been detected by their security tools. Or researchers themselves can use a software solution for detecting bots (for example, BotOrNot) to form a dataset. The key problem with such datasets is that they are generated by tools that also have performance indicators. Thus, learning on such datasets, the model can never exceed the efficiency of the tool on the basis of which the dataset was formed.

We see the problem in the fact that such datasets do not allow anyone to say that the bots in these datasets are really bots, and they also do not allow to reliably characterize bots. For example, to estimate their price, quality, or other characteristics. We believe that best option is to use datasets that were formed by buying bots from bot-sellers. In our opinion, the highest quality datasets can be purchased from companies that sell bots. In such datasets, the quality of a bot is known in advance. Depending on the company, one can find out the specific strategies of bots, since most likely, one strategy is used within one company and can be obtained from bot-seller. And there are no real users in such datasets. The disadvantages is that one need to make sure that bots will not have an impact on user communities when he/she will buy a bots to collect datasets. Also, it is difficult to generate large datasets in this way.

3.4.5 Group detection problem

Most of the research focuses on the individual classification of profiles. At the same time, if we consider a hypothetical tool for a mass search for bots, this approach is not very effective, since one need to scan each account separately.

It looks more logical to identify bots in two stages. For example, first, determine whether there are bots among users of a certain group and only then make a decision on individual detection.

Therefore, the need for individual account scanning makes bot detection tools inconvenient when a large number of profiles need to be scanned.

3.5 The goal of thesis

Based on the identified problems in existing solutions, as well as the need for bot detection, we can form the goal of the thesis.

The main goal of the thesis is to increase the security of users of social networks by developing new solution for recognizing and characterizing malicious bots. The proposed solution must satisfy the following requirements that follow from the problems:

1. Low API complexity. The proposed solution should require less information to detect bots and fewer requests to the social network in comparison with existing approaches.
2. Resistant to camouflage. The proposed solution is required to detect bots that use camouflage techniques.
3. High efficiency of detection. The proposed solution must have bot detection efficiency not worse than those of analogs.
4. Determining the price and quality of bots. The proposed solution should be able to determine the price and quality of bots, for which the training of the model should use datasets in which there are corresponding labels of quality and price.
5. Detection of a group of bots. The proposed solution should be able to detect a group of bots without analyzing each account separately.

In the following sections, we present the proposed solutions that allow us to achieve this goal, as well as their experimental and theoretical evaluation.

3.6 Chapter Summary

Tn this chapter we discuss the possible approaches for bot's detection that includes: available social network's data sources to perform analysis, existing feature construction approaches and decision making approaches based on machine learning techniques.

We also present 5 bot detection problems that researchers face when trying to develop methods for detecting bots: API limitation problem, bot camouflaging problem, detection efficiency problem, bad training datasets problem and group detection problem.

Based on the identified problems in existing solutions, as well as the need for bot detection, we formed the goal of the thesis and present requirements for our solutions.

In next chapter, taking into account the stated requirements, we present the proposed techniques for bot detection.

Chapter 4

Proposed features and techniques for bot detection

To detect bots, we have developed several approaches and methods to obtain the data sources, bots features and bot detection models. It includes: data extraction, feature construction and model training.

The main idea of these methods is that they **use only the list of adjacent accounts and graphs of friends** for an analysis. Thus, we use only two types of data sources: the distribution of numerical parameters of adjacent accounts and the graph of adjacent accounts. This is a very simplified approach to the problem of bot camouflage – after all, we do not analyze an account that is protected by privacy settings, but only the accessible metrics of users adjacent to it.

To do this, we have built a model of a social network that allows us to obtain graphs and lists of such adjacent accounts. We have proposed 3 categories of methods for feature extraction from the data. And we have proposed techniques for detecting bots based on the proposed features. The proposed approaches can be applied to any social networks in which it is possible to represent the relationships between people in the form of a social graph.

4.1 Social network model

The data of a social network can be represented as a graph in which the vertices will be objects and subjects of the social network, and the edges will be the relations between them. We propose to use semantic model for graphs representation. A semantic graph is a social network model that is defined by a set of vertices and edges that have various classes and properties. Using the vertex and edge classes, one can build various graphs. For example, one can build the graph, where the edges will be likes between users. At the same time, it is possible to construct the graph, where the edges will be comments between users. In this case, the same data – users and parameters of their posts (such as likes, comments, and links) – can give different types of graphs. Such graphs can differ topologically and in semantic content. This is important because bot detection depends not only on the bot's features but also on the input data. The semantic model allows one to extract different types of graphs, thereby significantly increasing the variability of possible types of input data for further feature extraction.

In this section, we present the approach to the practical representation of social networks as semantic models.

4.1.1 Graph representation of social network

The basis of the proposed representation of a social network is the semantic model with an hierarchy of classes of graph elements – vertices and edges. Vertices and edges are represented as inheritable tree of classes. This allows one to operate with classes of different levels of tree hierarchy.

Let us consider a hierarchical representation based on the analysis of the VKontakte social network. VKontakte is a Russian social network. VKontakte users can microblog on their behalf or on the behalf of the community, participate in several communities, post messages and media on friends' microblogs, upload and attach photo, video and audio content to the posts, give likes and repost posts.

The graph G of the social network can be represented as a collection of interconnected vertices V of subjects $S = \{s_i\}, i = 1, ..., n$ and objects $O = \{o_j\}, j = 1, ..., m$. These categories are characteristic of any typical social network. The subjects s_i in VKontakte are users $u_k \in U$ or communities $c_l \in C$. E are the action relationships Act (to be "friends", to participate in the community, etc.) and the relationships of inclusion $Incl$ (the post contains an image, the user profile includes videos, and so on.). At the highest level, the social network contains users U, communities C, objects O, action relationships A and relationships of inclusion I:

$$G = (V, E), V = S \cup O, S = U \cup C, E = Act \cup Incl$$

By a user $u_k \in U$, we mean the subject $s_i \in S$, who has a personal page assigned to a specific account (person). By community $c_l \in C$, we mean the social subnetwork with its pages. A user u_k can create and participate in same c_l. We will also imply that the content of the pages of u_k and the pages of c_l occurs according to similar principles.

By objects O, we mean various types of content that are used by S: media, media collections, micro- blogs, discussions, posts, etc. S and O can be organized into a hierarchical model of the graph's vertex classes. At low level, model can be organized as the following hierarchy of V (Figure 4.1):

1. Subjects S are accounts that represent real people; these can be both user accounts and community accounts (online shops, communities of interest, meetings and so on):

   (a) groups (or communities) C – meetings, events and other public pages, including their microblogs;

   (b) users U – user accounts, including their microblogs.

2. Objects O – the content that was created by the subjects:

(a) Collections – collections of various kinds and having different functions:

i. Discussion – a public space in the form of a message feed that was created by a subject for a specific topic or a discussion in which users can post the messages;
ii. Libs – collections of media files, such as video albums, photo albums, and audio playlists:

A. Photo-libs – collections of photos organized into photo albums;
B. Audio-libs – collections of audio recordings organized into playlists;
C. Video-libs – collections of video records organized into video albums;

(b) Posts – a container cards with the text which the subject publishes:

i. Origins – posts created by the subject personally;
ii. Reposts – posts posted by a subject, and referred to another post;
iii. Comments – posts posted as a comment to another post or discussion;

(c) Media – media content files and the like:

i. Poll – a multi-choice survey;
ii. Document – a downloaded file of any format;
iii. Video – a loaded video;
iv. Audio – a loaded audio;
v. Photo – a loaded image.

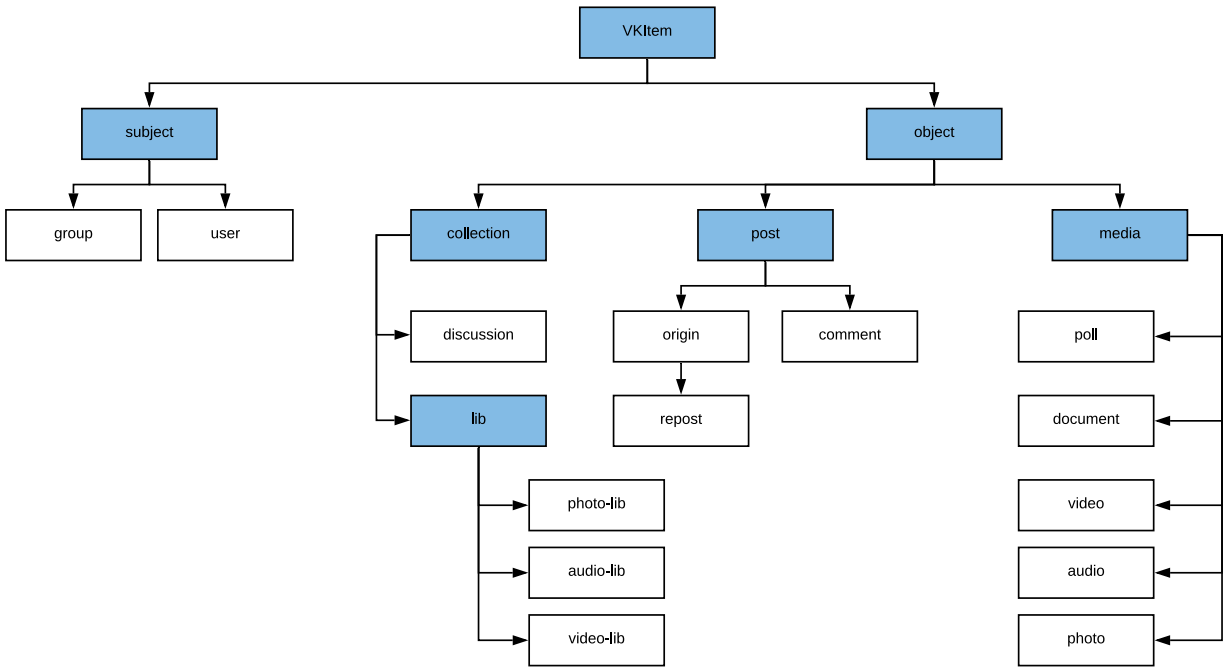

Figure 4.1: The hierarchy of vertex classes of the social network semantic graph of VKontakte (abstract classes are highlighted with blue).

As already mentioned, S can perform actions Act on O and S, O and S can be in the relation inclusion $Incl$ with O. So,

$$Act \subseteq S \times (O \cup S), Incl \subseteq (O \cup S) \times O$$

At low level, the relationships E can also be organized into a model. For VKontakte, the hierarchy of the edge classes is as follows (Figure 4.2):

1. The subject acts Act – the actions of the subject in relation to another subject or object:
   (a) Creates – actions by the subject on the object;
      i. Like – creation of like by subject;
      ii. Post – creation of post by subject;
         A. Repost – creation of repost (is referred to another post) by subject;
         B. Commented by – posting as a comment to another post or in a discussion;
   (b) Subscribe – creating relationships between subjects;
      i. Participating in a group / meeting – the relationship between users and groups;
      ii. Follow – one-way relationship between users;
         A. Friend – two-way relationship between users;
   (c) Administers – a relationship indicating that the user is the administrator of the group.
2. The object contains $Incl$:
   (a) Linked – an object contains a link to another subject or object;
   (b) Include – an object or a subject contains another object.

The final model of the class hierarchy of the VKontakte social network is represented in Figure 4.3 as a matrix. Rows and columns depict the hierarchy of the vertices V . Cells of the matrix reflect which classes of E can exist between one or another class of V . Based on this model, one can build G. The vertex classes are in matrix titles of rows and columns, the edges classes are in matrix cells.

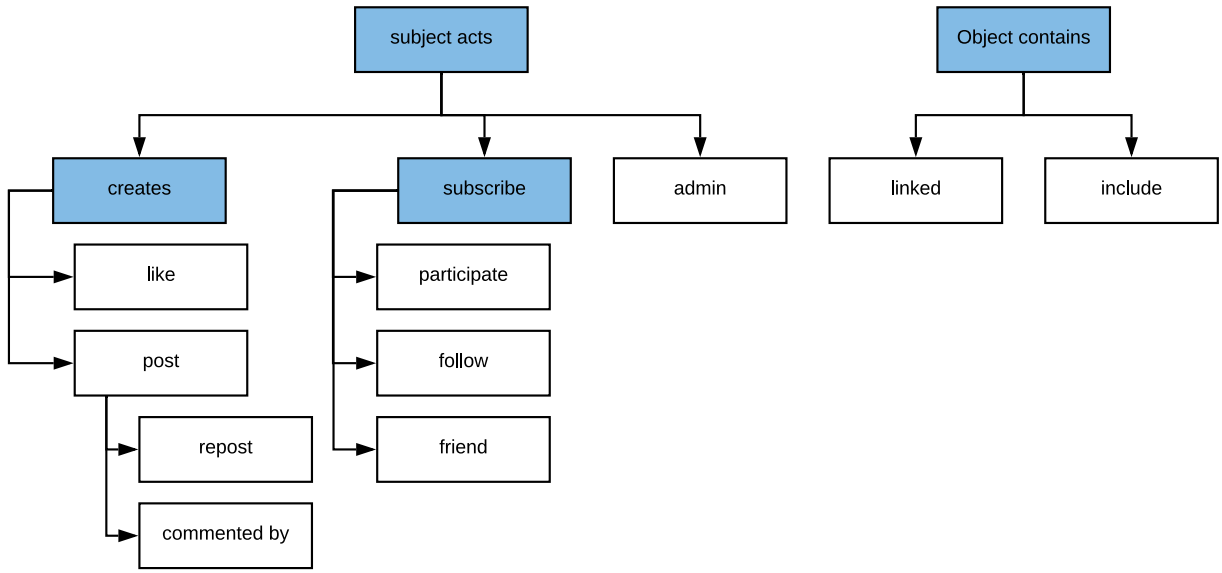

Figure 4.2: The hierarchy of edge classes of the social network semantic graph of VKontakte (abstract classes are highlighted with blue).

4.1.2 Interaction graphs in hierarchical model of social network

The schematic model of the graph allows to generate two types of graph structures: real graphs and virtual graphs.

By *real graphs* we mean structures that are derived from data directly. Thus, a real graph is a graph that can be obtained by scanning a social network and which will be stored in a database according to a specific data model. An example is shown in Figure 4.4, where the real graph is formed by continuous edges: a post with ”like” forms the real structure, which consists of the post, the user who created the post and the user who liked the post.

A *virtual graph* is the graph derived by processing a real one, for analyzing a specific process. For example, in Figure 4.4, the virtual graph is represented by dashed edges and shows ”likes” between users. Although the user cannot directly ”like” another user (in case of the VKontakte), we can get them indirectly on the basis of likes to the content and form a graph of users likes to each other.

Algorithms, that determine the similarity of instances of a class, allow one to aggregate the vertices of the graph, thereby creating a new structure. A simple example of aggregation is shown at Figure 6: a user liked a post with an image that one group loaded, another user liked a post of another group that contains a similar image (Figure 4.5 a). If these two photos are identical, their vertices may be combined into one (Figure 4.5 b). Later, when creating virtual graphs, the post vertices can be deleted, forming the virtual graph of images the users like (Figure 4.5).

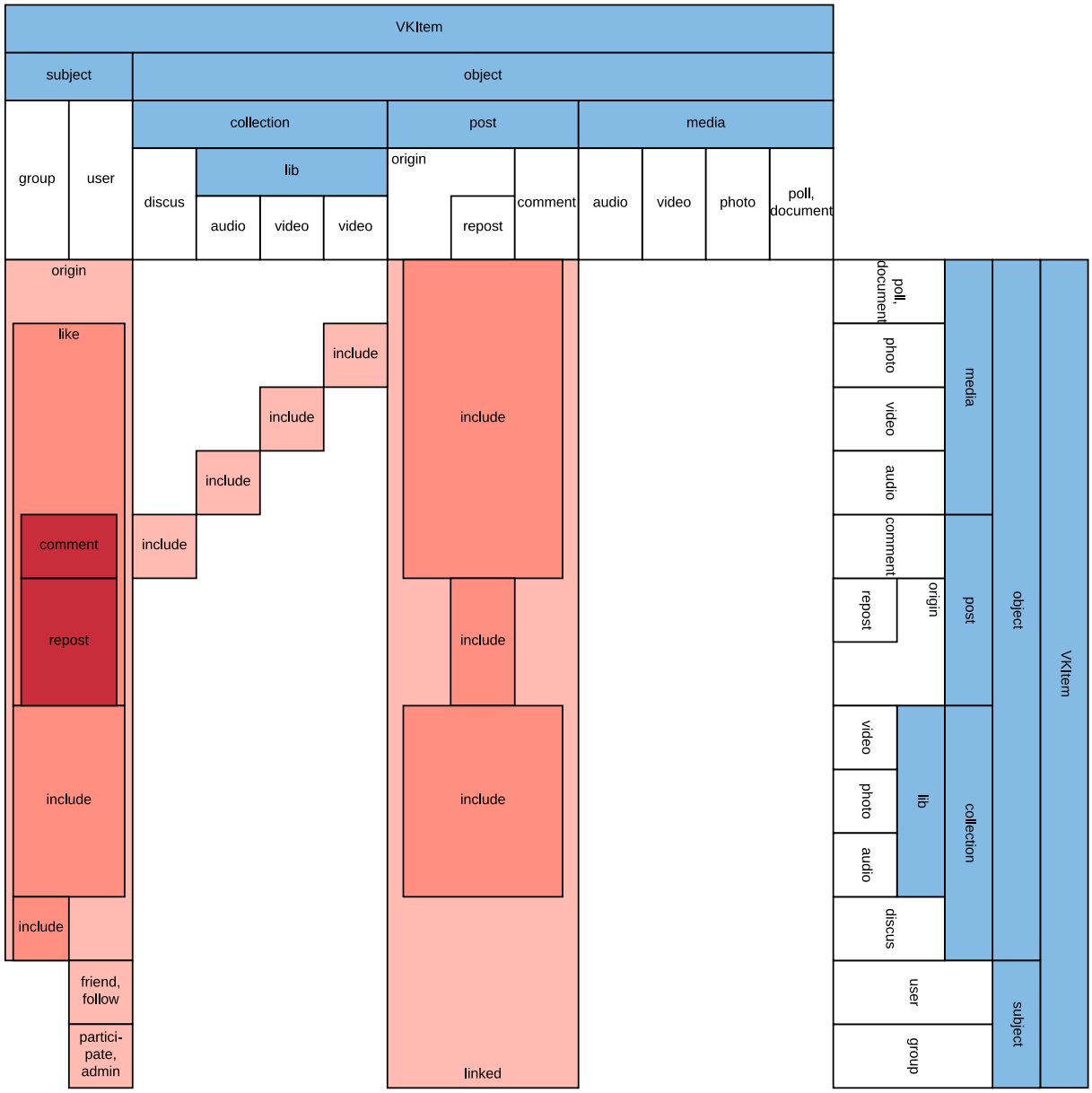

Figure 4.3: A hierarchical semantic model of the VKontakte social network graph.

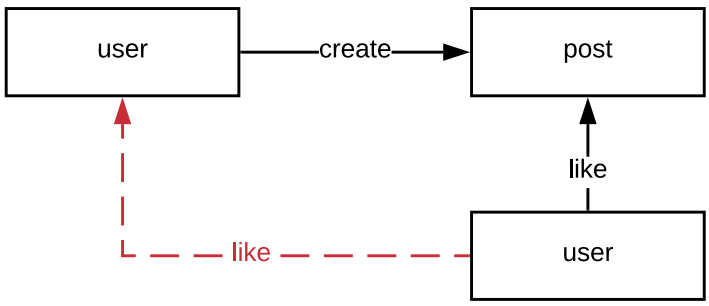

Figure 4.4: The real graph is represented by continuous edges, the virtual graph – by dashed edges.

Creation of new graphs is necessary to extract features that will be used in bot detection, as more specific graphs are capable of carrying more information for creating features. Based on the hierarchical data model of the social network, it is possible to create required types of graphs (real and virtual) by combination of hierarchies or by creation of virtual edges or using instance aggregation. The results of the algorithms will depend on the context of the analyzed graph. In the next section, we give an example of graph analysis using the presented data model and graph representations.

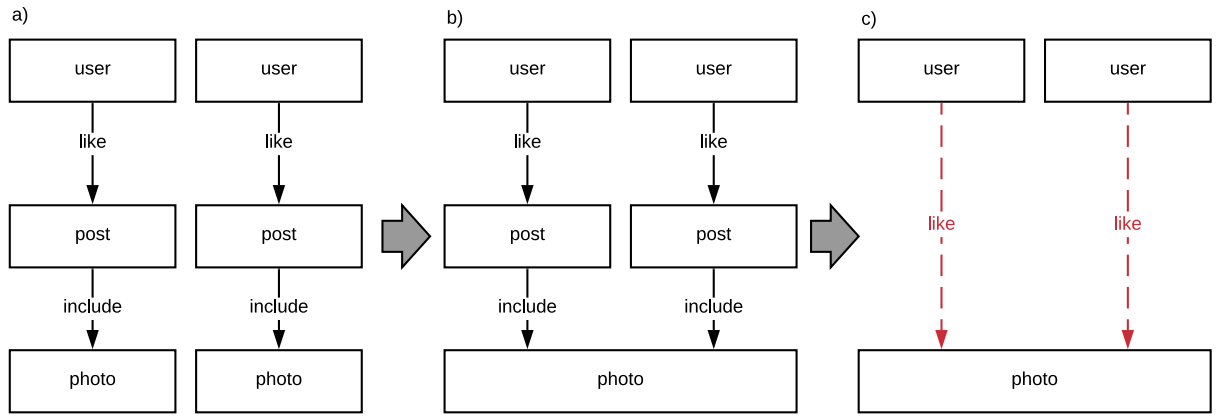

Figure 4.5: Aggregation of the graph vertices by the class “photo”.

4.1.3 Possible data sources

We propose to use the graph of users to detect the bot. Thus, the vertices of the desired graph are *users*, the interactions are various classes of edges. According to the semantic model (Figure 4.3) between users can be the following classes of edges (types of interactions): *follow* and/or *friend*. In the class hierarchy, the *follow* and *friend* classes are extended from the *subscribe* class (Figure 4.2). The *participate* class is also extended from the *subscribe* class. At the same time, the *participate* edge can only be between the *user* and the *group*, so querying from semantic model the request ”*users subscribe users*” will only give the *follow* and *friend* edges.

Another type of user interaction is that users leave posts and comments on each other’s microblogs, posts and reposts. Within the framework of the hierarchical data model, it looks like in Figure 4.6 and 4.7, where: the continuous edge *post* and *repost* show who posted the record; the continuous edge *include* indicates that the entry contains another entry or that the user’s microblog contains this entry; the dashed edge shows the virtual edge it is needed to get.

The specifications of the options are as follows:

1. Option a – *user* A can $\xrightarrow{\text{post}}$ an *origin* on $\xleftarrow{\text{include}}$ *user* B (Figure 4.6a).
2. Option b – *user* A can $\xrightarrow{\text{repost}}$ a *repost* that $\xrightarrow{\text{include}}$ *origin* that was $\xleftarrow{\text{post}}$ by *user* B (Figure 4.6b). Note that *user* A can $\xrightarrow{\text{repost}}$ a *repost* only on his own page (*user* A $\xrightarrow{\text{include}}$ *repost*). Also, *user* B may not necessarily $\xrightarrow{\text{post}}$ an origin on his own page — he can do it on the page of another *user* or *group*.
3. Option c – *user* A can $\xrightarrow{\text{repost}}$ a *repost* that $\xrightarrow{\text{include}}$ *repost* that was $\xleftarrow{\text{repost}}$ by *user* B (Figure 4.6c). For our purposes, from where *user* B $\xrightarrow{\text{post}}$ a *repost* does not matter, since we analyze only the interaction between *users* A and B.
4. Option d - *user* A may $\xrightarrow{\text{like}}$ the *origin* that was $\xleftarrow{\text{post}}$ by *user* B (Figure 4.6d).
5. Option e - *user* A may $\xrightarrow{\text{like}}$ the *repost* that was $\xleftarrow{\text{post}}$ by *user* B (Figure 4.7e).
6. Option f – *user* A can $\xrightarrow{\text{post}}$ a *comment* to $\xleftarrow{\text{include}}$ *origin* that was $\xleftarrow{\text{post}}$ by *user* B (Figure 4.7f).
7. Option g – *user* A can $\xrightarrow{\text{post}}$ a comment to $\xleftarrow{\text{include}}$ *repost* that was $\xleftarrow{\text{repost}}$ by *user* B (Figure 4.7g).
8. Option h - *user* A can $\xrightarrow{\text{like}}$ the comment that was $\xleftarrow{\text{post}}$ by *user* B (Figure 4.7h).

The hierarchical structure allows to simplify all cases. On the interaction schemes, the *user* A performs the edge-actions: *post*, *repost* and *like*. All these actions are extended from the *creates* edge class. *User* A performs the *creates* action on objects of the classes *origin*, *repost* and *comment* which are extended from the vertex class *post*. These objects can have edges: *include* to the vertex *user* (Figure 4.6a), include to the vertex *post* (Figures 4.6b, c and 4.7f,g) or *creates* to the vertex *user* (Figures 4.7d and 8e, h). Thus, the presented cases are reduced to three schemes:

1. *User* A $\xrightarrow{\text{create}}$ *post* that $\xleftarrow{\text{include}}$ *user* B.
2. *User* A $\xrightarrow{\text{create}}$ *post* that $\xleftarrow{\text{create}}$ *user* B.
3. *User* A $\xrightarrow{\text{create}}$ *post* that $\xleftarrow{\text{include}}$ *post* that $\xleftarrow{\text{create}}$ *user* B.

Based on these queries to semantic model, one can generate several graphs of users, which will be associated with analysed account and can be used as source of data for feature construction.

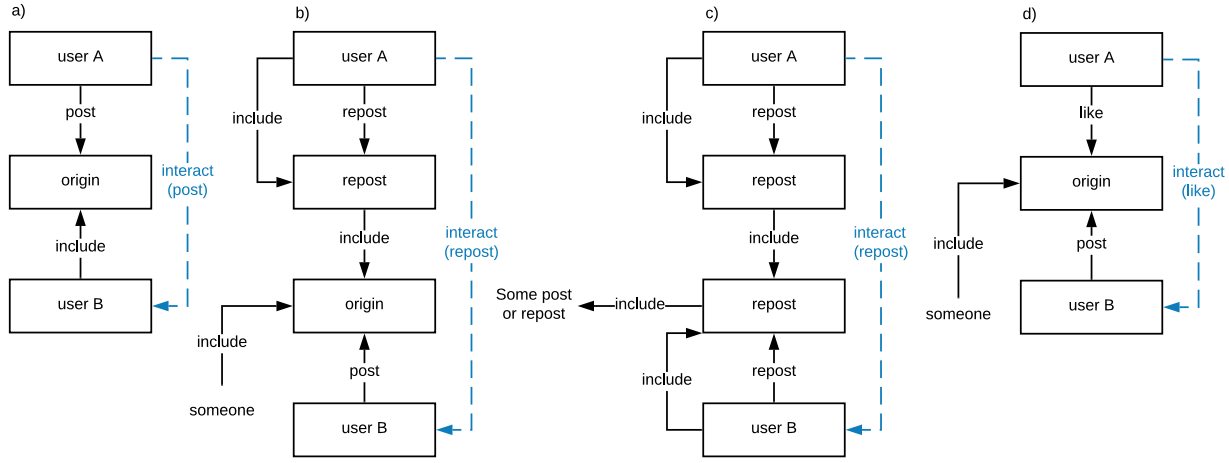

Figure 4.6: User to user interaction options *a-d*.

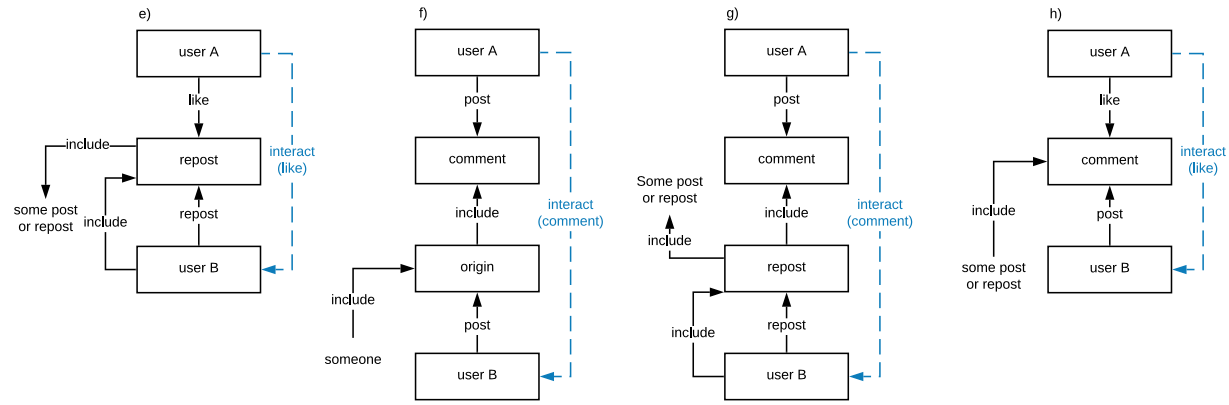

Figure 4.7: User to user interaction options *e-h*.

4.1.4 Friend list as main data source

Using the proposed social network model, it is possible to create various user graphs and even content graphs. This approach allows one to significantly expand the number of data sources from which one can construct features. At the same time, this significantly expands the scope of the proposed approach. Using the proposed semantic model, one can generate graphs not only for traditional social networks but also for messengers, forums, websites with social functions, etc.

For bot detection, we will concentrate on the graph that is present in almost all social networks – the friend graph.

We assume that the input data is a *id* of a user account. For it, one will need to determine the graph of friends, a set of features of this graph, and then determine whether the account is a bot. In this section, we describe how we can get a graph of friends. We propose to use a two parts schema:

1st step – graph traversal. The first step is to find friends of analyzed accounts, connections between them, and generate a graph. Data collection can be described using a traverse function with a depth of 2, where the vertices will be *user* and the edge will be *friend* (according to model on Figure 4.3). At depth 1, one can get a *list of friends* that is associated with the analyzed account. At depth 2 one gets *friends of friends*, and one can associate them with the *list of friends* we got at depth 1. Visually, the vertices and edges of the graph that we get are shown at the top of Figure 4.8: the vertices and edges that one obtains in the result of 1st depth traversal are highlighted with blue color, and for 2nd depth, traversal are highlighted with red color.

Since we have determined the type of graph , we can calculate the API complexity of obtaining such a graph. The API complexity (the number of API requests) of getting such a friend graph from the VKontakte social network depends on the size of the set of vertices for which the traverse is performed. For 1st depth traverse API complexity is 1 (as there is only one vertex) and for the 2nd depth traverse API complexity equal to the number of vertices obtained at depth 1 traversal (number of friends). Thus, overall API complexity is:

$$API\ complexity = 1 + number\ of\ friends$$

2nd step – graph generation. Based on the obtained graph, one can generate several significant subgraphs of friends, which will have different topologies. The resulting graph has 4 vertex layers (middle of Figure 4.8) – (1 – blue) the analyzed account, (2 – red) the list of friends, (3 – orange) mutual friends of friends, and (4 – cyan) friends of friends.

Layer 2 (friendslist) must be present in all types of graphs, while other layers can be included or filtered:

- 1st layer – the analyzed account. It is connected with all vertices from layer 2, so its inclusion/filtering may affect the results of algorithms using path calculation.
- 3rd layer – the mutual friends of friends. They allow one to establish indirect friendships for vertices from layer 2 through mutual friends. Inclusion/filtering may affect the results of algorithms using path calculation and the number of neighboring vertices.
- 4th layer – friends of friends. It allows one to determine the number of friends for vertices from layer 2. Inclusion/filtering may affect the results of algorithms using the number of neighboring vertices.

These graphs can be used to extract features. Depending on the selected type of graph, features can take on different values.

For bot detection we will use graphs of friends, excluding the following types of vertices from the graph: analyzed account (layer 1), mutual friends of friends (layer 3), and friends of friends (layer 4) – as shown in the bottom of Figure 4.8, where vertices and edges that are included in the analysis are highlighted with red.

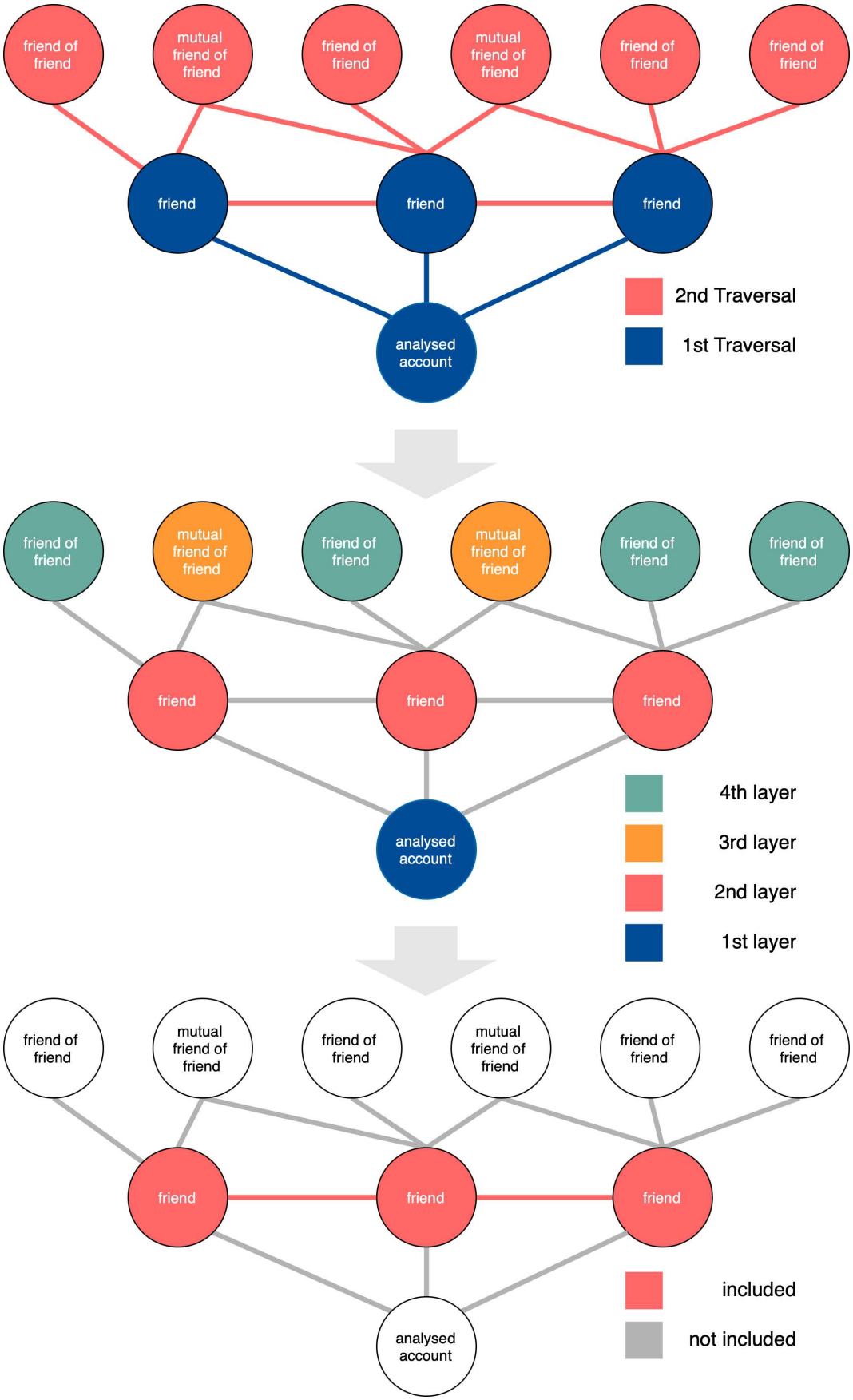

Figure 4.8: Graph traversal and graph generation scheme.

4.2 Proposals of feature construction techniques

We propose to detect bots based on features that will be constructed from social network accounts' data. Features is a characteristics of analysed account. On the basis of these features, machine learning methods will make decisions (bot/ not a bot). To build an informative set of features, we need to analyze the social network data related to the analyzed account.

For this, we propose 2 types of data for analysis: a list of adjacent accounts (accounts who performed an action on the object/subject) and a graph of friends (bottom of Figure 4.8). We propose 3 categories of methods for constructing features from this data:

1. Statistical – methods that we will apply to extract features from the numerical parameters of the friend list.
2. Graph based – methods that we will apply to extract features from the topology of friends graph.
3. Graph embeddings based – methods that we will apply to extract features from the locations of friends or adjacent accounts on the graph embedding of social network (a graph that represents the topological structure of the entire social network).

4.2.1 Proposed statistics based features

We use statistical methods to analyze the numerical parameters of accounts. First, we need to get numerical data from a list of friends. To do it we collect the following data for users:

1. The number of friends. Each user has its friend list. To make friends, a user must send a request, and another user must confirm it.
2. The number of groups. In VKontakte users can join groups. Groups can be dedicated to some event or topic.
3. The number of subscriptions. A person can subscribe to another person. The difference between a subscription and a friendship is that you don't need another person's confirmation to subscribe.
4. The number of followers. Followers are people who subscribed to your account.
5. The number of photos. A person can upload some photos.
6. The number of albums. A person can combine photos into albums.
7. The number of posts. Each VKontakte user has his/her microblog. The number of posts is the number of records in this microblog.
8. User id. Each VKontakte user has an ordinal id – users who registered earlier have a lower id value, and newer ones have a higher value.

Based on this data we form distribution of the number of friends, distribution of the number of groups, and so on. For each distribution we construct the following features:

1. Basic statistical metrics: mean, Q1, Q2, Q3 quartiles, and standard deviation.
2. The p-value of the distribution agreement with Benford's law.
3. Gini index.

A summary of the construction of features based on statistical methods is presented in Table 4.1.

Let us describe in more detail the new feature construction methods - Benford's law and Gini index.

Metric (distribution)	**Statistical method**	**Features**
number of friends number of groups number of subscriptions number of followers number of photos number of albums number of posts	mean Q1 Q2 Q3 deviation Benford's law Gini index	7 * 7 = 49 features
id	mean Q1 Q2 Q3 deviation	1 * 5 = 5 features

Table 4.1: Proposed statistical based feature construction scheme.

4.2.1.1 Benford's law

As one of the feature construction approach, we propose to use the agreement with Benford's law. It is already used for detection of fraud [CBC+19, GHD17]. It is important to mention paper [Gol15] where random samples of users for Twitter, Google Plus, Pinterest, Facebook, and LiveJournal social networks were studied. The research showed that friends and followers metrics obey Benford's law (except for Pinterest followers, but it can be simply explained). Although the study was not about bot detection, some parts of datasets that had a low correlation with Benford's distribution (under 0.5%) were represented by spam accounts and bots.

Our hypothesis is that agreement with Benford's law can be used as a feature in bot detection task since bots break Benford's law significantly more often than human users because of their unnatural behavior.

In its simplest form the "Benford's law for the leading digit" is formulated as follows. First, every positive number x has a unique representation in "scientific notation", that is, in the form $S(x) \cdot 10^k$, where $S(x) \in [1, 10)$ is called its *significant* and k is an integer called its *exponent*. The integer part d of $S(x)$ is called its *first digit*. Benford's law definition uses logarithm, and taking into account that for finite datasets frequencies cannot be irrational numbers, the following working definition is used (see [Mil15]):

A dataset satisfies Benford's law for leading digit if the probability of observing a first digit of d is approximately $log_{10}(\frac{d+1}{d})$.

Benford's distribution is valid for the distribution of the first numbers in the decimal system, but does not depend on the unit of measurement (if one convert centimeters to inches, then the distribution will not change).

This is a bit counterintuitive, but the rational explanation lies in the fact that the greater first significant digit of the metric is more difficult to obtain than smaller one. This complexity is described by the logarithmic distribution. For example, uploading 1 photo is easier than 2-9, 10-19 photos is is easier to upload than 20-99 and so on. Benford's Law describes this complexity by considering all ranges.

We propose a feature construction approach that is based on Benford's law tests for metrics of multiple users. The feature is *p-value*. This means that the approach can be used to construct the **feature for a group of accounts** – p-value for distribution of some metrics of the group (number of posts, friends, and so on). To get a feature for an individual account, one can use a friend list of accounts as a group, or other lists of adjacent accounts.

We will calculate p-values for the following distributions of numerical metrics of accounts: number of friends, number of groups, number of followers, number of subscriptions, number of photo albums, number of photos, number of posts, and any other distribution of numerical metric.

We propose to use the following schema: **At first step** for distribution of accounts' metric we get the list of the first significant digit. **At the second step** for distribution of the first significant digit we test the agreement of metric with Benford's distribution using the Kolmogorov-Smirnov test.

The pipeline of that process is presented in Figure 4.9.

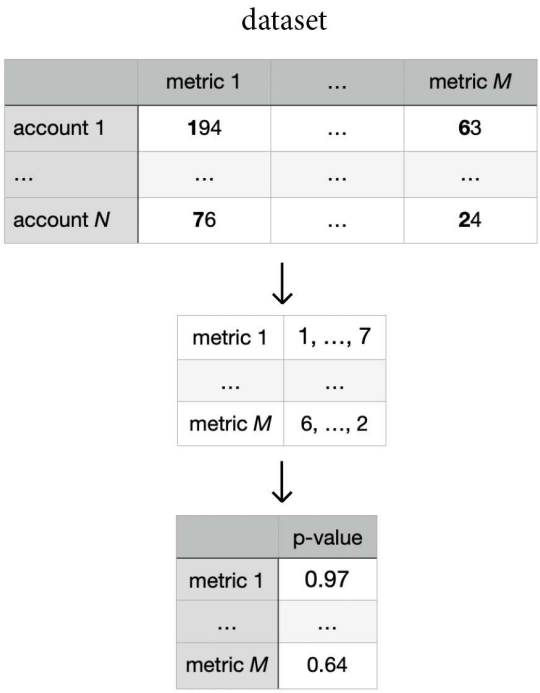

Figure 4.9: Pipeline for calculation of agreement with Benford's law.

4.2.1.2 Gini index

Gini index [Gin12] is a statistical measure that is very popular in economics for the expression of income inequality – how the resources are distributed among the population. It is sometimes used in other areas as well, for example, to assess biodiversity [GG12].

Our hypothesis is that the values for the number of friends, photos, etc. can be interpreted as the "wealth" of a social network account because to increase these numbers users need to do more actions. More filled accounts with more friends, photos, posts, and so on are richer. This can also be justified by the cost of bots - bots with more complete profiles are more expensive. Thus, by performing any actions, the user increases its capital, which is expressed as a metric. So the Gini index can indicate equality or inequality of users in the friend list.

Gini index can be defined [Kak80] using relative mean absolute difference:

$$gini = \frac{\Delta}{2\mu}$$

where

$$\Delta = \frac{1}{n(n-1)} \sum_{n}^{i=1} \sum_{n}^{j=1} |x_i - x_j|$$

x_i and x_j being the wealth of the i^{th} and j^{th} user respectively, n is the total number of users, and μ is the mean of the distribution.

For better understanding, the Gini index can also be represented through the Lorentz curve [Kak80]. The Lorentz curve is a graphical representation of cumulative percentage of total wealth. It can express statements like "top 10% of users own 90% of wealth". We represent it graphically on Figure 4.10, where the y-axis represents the total amount of wealth possessed by the x number of users on the x-axis. Accordingly, users are sorted in ascending order from wealth – richer users are located on the right side of the x-axis, and poorer users on the left. The straight line is the line of equality when all users have the same amount of wealth. Thus, the Gini index is the ratio of area A (between the Lorentz curve and the line of equality) and $A + B$ (total area).

$$gini = \frac{A}{A + B}$$

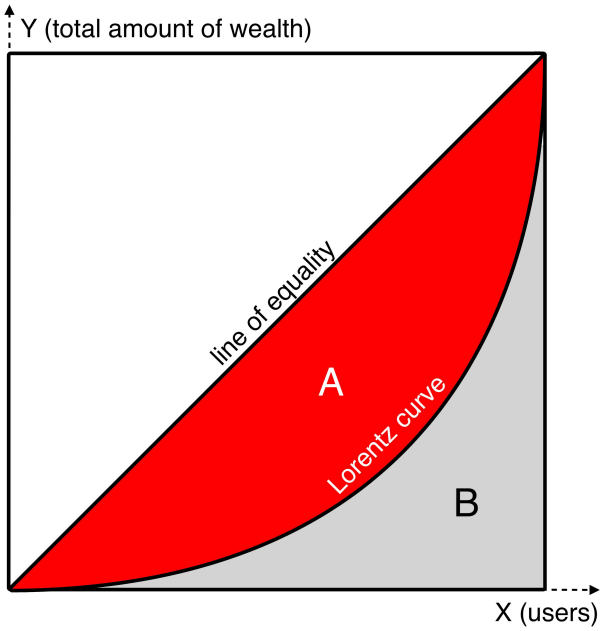

Figure 4.10: Graphical representation of Gini index through the Lorentz curve.

We propose to use the Gini index as a feature of multiple users. As for Benford's Law, the Gini index can be used to construct the feature for a group of accounts. Thus, to analyze an account individually, one can use a group of friends as a data source.

4.2.2 Proposed graph based features

Since using the graph as the source of information, we can apply many graph analysis algorithms to construct features. Most algorithms produce results that need to be converted into a numerical metric for the entire graph to be used in machine learning as features. Therefore, we construct most of the features in two stages: we apply graph algorithms and then analyze the results by statistical methods.

At first step, we use the following graph algorithms [New18] that we classify by type of result:

1. Algorithms that calculate the centrality (some weight) of each vertex in the graph. Thus, we obtain the distribution of the vertex centrality measures. We used several centrality measures: Degree centrality, Closeness centrality, PageRank, VoteRank, Katz centrality, Load centrality, Effective size, Average neighbor degree.

2. Algorithms that calculate the subset of vertices: Dominating Set, Maximal independent set, Isolates set, Bridges.

3. Algorithms that calculate the list of subsets of vertices: Cores, Communities detection based on modularity, and Communities detection based on label propagation.

4. Algorithms that calculate different coefficients: Clustering coefficient, Global efficiency, Degree Assortativity, S-metric, community modularity.

At second step, we calculate next statistics: mean, Q1, Q2, Q3, min, max, standard deviation, number of sets, the relation of a subset to the entire set. The scheme of applying statistical methods to the results of graph algorithms is presented in Table 4.2.

As a feature, we also use the following data, which we receive along with the graphs: distribution of identifiers (ids) of graph vertices, the id of analyzed account, number of vertices in the graph, number of edges in the graph. As the result, we have 78 features that describe the graph numerically.

4.2.3 Graph embeddings based technique

Feature extraction from graph embedding is based on the hypothesis that a social network has a *large-scale topological graph structure* in which all users are combined into one giant component. This large-scale graph is a graph of the entire social network (or has its properties). The idea is to use the location of the account in this structure as a data source for feature construction. Thus, features that we will be presented in this section are based on the distance between friends' locations. We assume that these distances will be different for bots and real users.

First, we need to build such a structure for a large-scale social network friends graph. To do this, we collect a graph consisting of 9 million random non-empty (there is at least one friend) Vkontakte users. In total, VKontakte has about 600 million users and by our empirical observations only 25% of them have friends (since when collecting, only 1/4 of the users returned non-empty values). Thus, we collected about 6% of active VKontakte users. This means that if we analyze an account that has friends, we have a 6% chance of finding it on our large-scale graph structure.

step 1		step 2	
Graph algorithm	**Result**	**Statistics**	**Features**
Degree Closeness PageRank VoteRank Katz Load Effective size Average neighborhood degree	distribution	mean Q1 Q2 Q3 deviation	$8 * 5 = 40$ features
Dominating set Maximal independent set Isolates set Bridges set	set	$\frac{size(Set)}{size(V)}$	$4 * 1 = 4$ features
Community (modularity based) Community (label propagation based)	list of sets (distribution of set's sizes)	mean Q1 Q2 Q3 deviation min max	$2 * 7 = 14$ features
	modularity	-	2 features
K-cores	list of sets	mean Q1 Q2 Q3 deviation number	$1 * 6 = 6$ features
Clustering coefficient Global efficiency Degree assortativity S-metric	coefficient	-	4 features
ids of vertices	-	mean Q1 Q2 Q3 deviation	5 features
id of analyses account number of vertices number of edges	-	-	3 features

Table 4.2: Proposed graph based feature construction scheme.

The existence of a large-scale structure is logical for a social network according to the theory of the six handshakes. As Facebook study [EDF+16] have shown that each user is connected to another user through an average of 3.5 other people. On our data, we got a value of 7.5 handshakes between users using the Dijkstra algorithm (this value is overestimated due to incompleteness of the data, but it is enough to form a structure).

4.2.3.1 Build a graph embedding

We propose to structure the large-scale graph using embeddings. Building graph embeddings is a machine learning technique that is used to encode the graph into the feature space (embedding space), where each vertex will correspond to a vector. Vertices between which there is less distance on the graph (fewer handshakes) will be located at a smaller distance in the embedding space. This method is used for learning on graphs and for visualizing graphs. Such visualization technique is used when graphs are very huge and it makes no sense to display edges or even vertices separately. Instead, the density of the vertices is displayed.

We propose to build a graph embedding and use visualization to verify that structure is formed. In its simplest form, the visualization technique includes 3 steps (see Figure 4.11 where arrows indicate steps): (1) converting vertices into feature vectors; (2) compressing the dimension of a feature set to two-dimensional, that is suitable for 2D visualization; (3) visualize the density of vertices. In this way, the features become the coordinates of the vertices and the density of the vertices can be displayed graphically. The closer the distance between the feature vertices, the closer they are on the graph.

We used the first step to build a series of graph embeddings with different parameters. And we used the second and third steps to make sure that the structure of the large-scale graph is being formed.

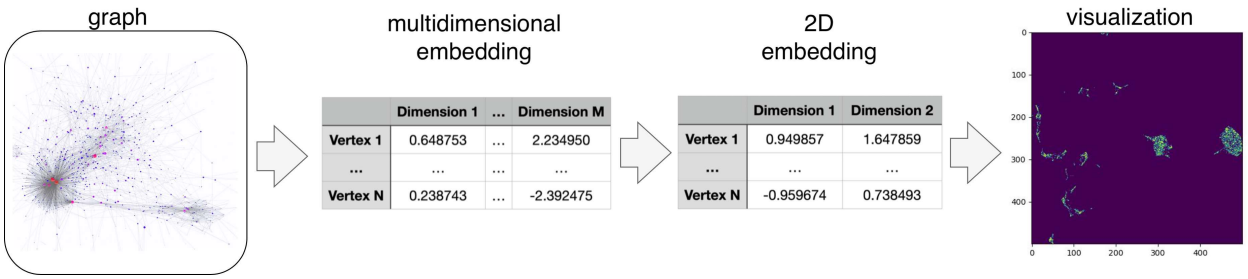

Figure 4.11: Embedding based visualization technique.

To transform vertices into a set of features, we used node2vec. Node2vec is a 2-step framework that learns low-dimensional representations for nodes in a graph by optimizing a neighborhood preserving objective [GL16]. At the moment, this is the best algorithm for transforming a graph into a feature space [GL16]. It uses random walks to generate sequences of vertices (step 1) and learned on them using neural nets with skip-gram architecture (step 2). Skip-gram architecture has been originally developed in the context of natural language processing - neural network with one hidden layer tries to predict the context words given a target word (ex. word2vec [MCCD13]). But in node2vec, instead of a word and sentences, the vertex is the input, and the output neurons are trying to predict the vertices that appear in the chain near the input vertex. After training the model, the output of the hidden layer will be the graph embedding, the dimension of which is equal to the number of neurons in the hidden layer.

To reduce the dimension of a set of features, we used the Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) algorithm [MHM18, CP19]. UMAP is a dimension reduction algorithm that replaced popular t-SNE, as UMAP provides the best speed and results. UMAP constructs the multidimensional graph where each data point is the vertex, and edge weight is a probability that two vertices are connected. To link vertices, UMAP increases their radius in the Riemannian manifold, and those radii that intersect are linked. After, UMAP trying to find a low dimensional representation of a graph with a similar topology by applying a force-directed layout.

To visualize the two-dimensional embedding we used Datashader that aggregates points and rasterizes the result into a grid of a given size [BLSSA17]. Datashader is the most popular framework for high volume data visualization.

To form the embedding, we used the grid-search technique: we went over the number of node2vec parameters (embedding dimensions from 2 to 400). After that, we reduced the number of dimensions to 2D using UMAP. We chose the best result that gave the clearest picture of the large-scale structure. The best result on our data was given by an embedding with a dimension of 7. Its 2D representation is shown in the Figure 4.12.

It should be mentioned that we tried two different strategies to form a graph for embedding. Embedding that presented in the Figure 4.12 is formed from the graph where we *exclude not scanned mutual friends* between scanned users as shown on the right side of Figure 4.13. We also tried to form embedding from the graph that *include not scanned mutual friends* as shown on the left side of Figure 4.13. Including mutual friends would significantly increase the number of users as the unscanned users would be included in the embedding. Also, additional paths between users would be formed – paths through mutual friends. But this strategy leads to embedding that clustered into 2 parts – scanned users and unscanned mutual friends (see the result on the top left side of figure 4.13). Therefore, we abandoned this strategy and use only the graph where we exclude not scanned mutual friends.

Now, based on the location of the account in the embeddings, we can construct features. We will construct features using multidimensional embedding (the result of step 1), so it carries more information. We used steps 2 and 3 to verify that the structure is formed.

4.2.3.2 Proposed graph embedding based features

With this embedding, we can calculate several features that describe the account. Since we are analyzing the account not directly, but through the list of friends, for the analyzed account we are looking for its friends in the embedding. On average, we find about 6% friends, but that's enough for our analysis. Thus, the input data is the list of friends of the analyzed account. Based on this list, we can construct the following features:

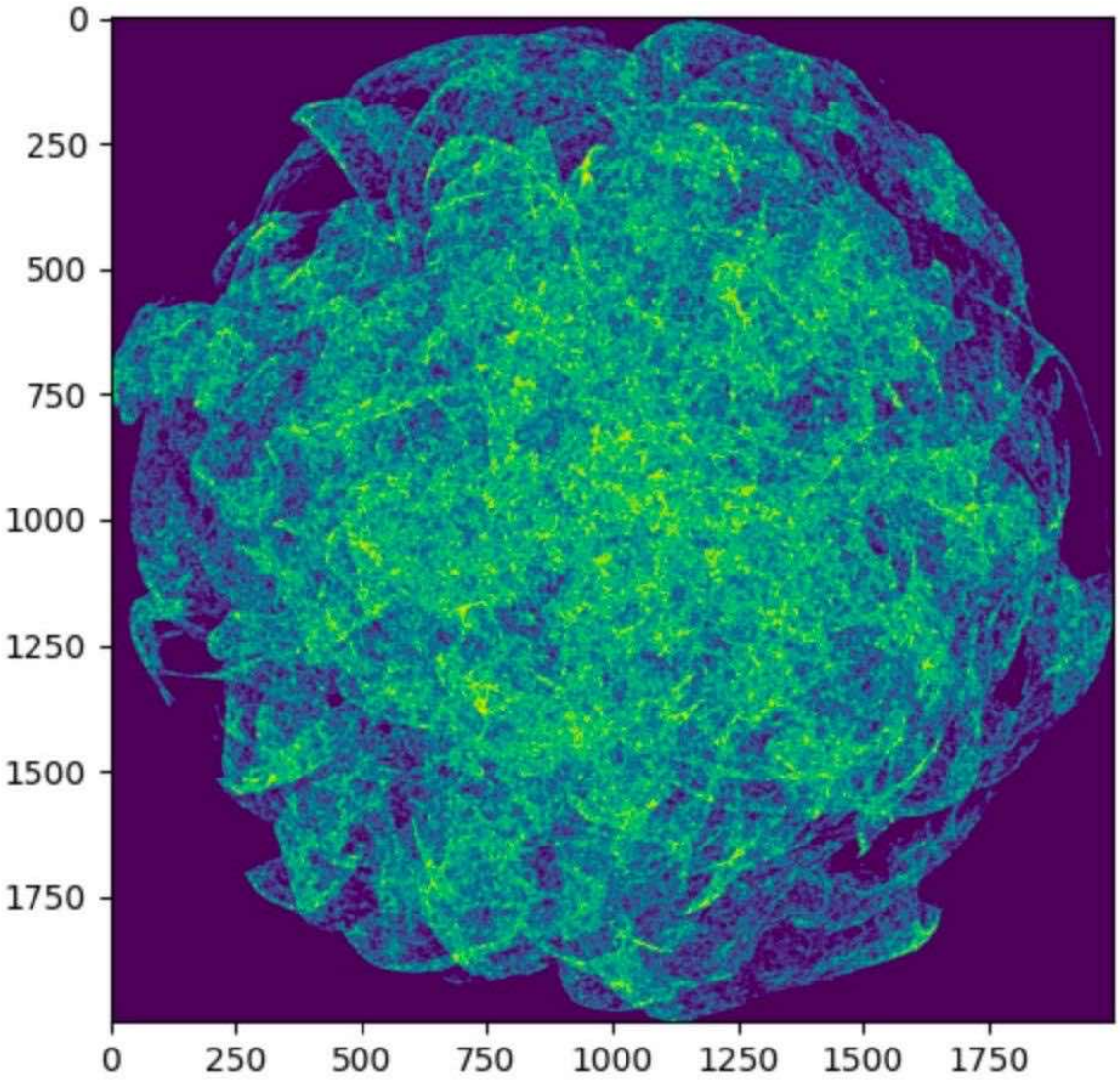

Figure 4.12: Embedding that was was built from 9 million random VKontakte users according to their friends graph.

Average friend distance (AFD). We assume that the distance between the real user's friends and the distance between the bot's friends on a large-scale structure will be different. To construct feature of analyzed account, for each friend in the friend list we calculate the average Euclidean distance between friend and r random another friends:

$$AFD(i, r) = \frac{\sum_1^r euclidean(friend_i, friend_{random(N,i)})}{r},$$

where i – friend index in friend list, r – the number of random friends from the list for which the distance will be calculated (used in order not to iterate over all combinations), $random(N, i)$ – function that returns random index $\in [1, N] \backslash i$, N – the number of friends.

After calculating AFD for all friends from the list, we have the distribution of AFD. The features of the analyzed account will be the statistical mean, Q1, Q2, Q3, and standard deviation of AFD distribution.

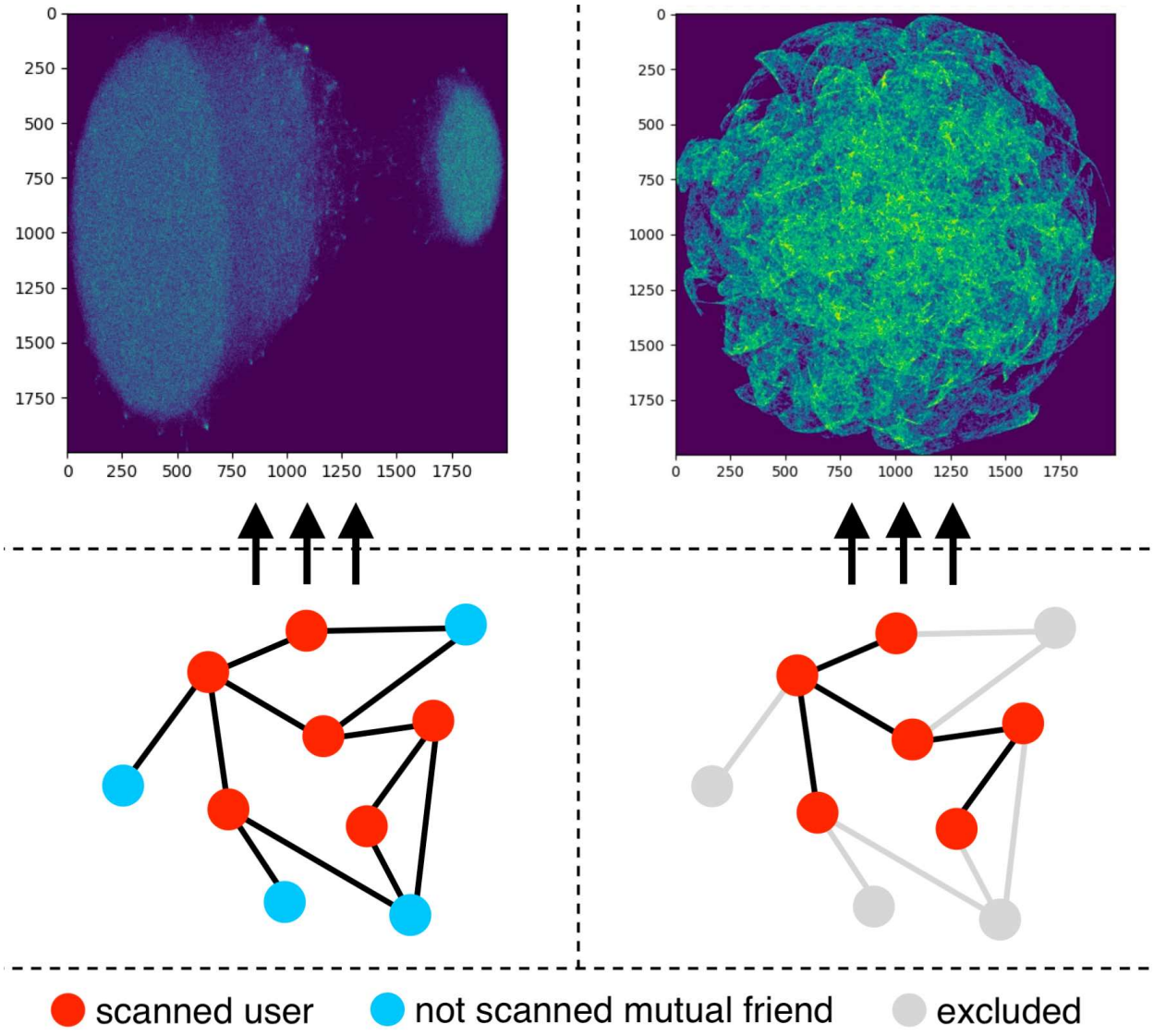

Figure 4.13: 2 strategies to form graph for embedding.

Average zero distance (AZD). We assume that the distance from the real user's friends to the center of the graph and the distance from the bot's friends to the center of the graph will be different. To construct feature of analyzed account, for each friend in the friend list we calculate the euclidean distance between friend and center of the graph:

$$AZD(i) = euclidean(friend_i, zero),$$

where i – friend index in friend list, $zero$ – the coordinates of the center of the graph that correspond to zero vector.

After we use the same statistics as for AFD. Mean, Q1, Q2, Q3, and standard deviation of AZD distribution will be the features of the analyzed account.

Fullness metric (F-metric) – is the fraction of friends that we found in the embedding:

$$F\text{-}metric = \frac{\sum_{i=1}^{N} exist(i)}{N},$$

where i – friend index in friend list, $exist(i)$ – function that returns 1 if i^{th} friend exist in embedding and 0 if not, N – number of friends in friend list.

It should be noted that graph embedding needs to be calculated only once for the entire social network. It can be periodically updated as the social network and users' connections evolve. This means that the proposed features only require the identifiers of friends of the analyzed account to construct features. A summary of the features that are constructed based on the embeddings of the graph with large-scale structure is presented in Table 4.3.

Metric	Statistical method	Features
AFD AZD	mean Q1 Q2 Q3 deviation	2 * 5 = 10 features
F-metric	-	1 feature

Table 4.3: Proposed graph embeddings based feature construction scheme.

4.3 Bot detection techniques

Based on the proposed methods for generating user graphs and methods for constructing features, we propose several techniques for detecting bots and determining their characteristics. We propose 4 techniques: (1) technique for group detection of bots, (2) technique for individual bot detection, (3) technique for determining the quality of bots, (4) technique for determining the price of the bot.

The first three techniques are very similar to each other and have common components. The general scheme is a data pipeline and consists of the following modules:

1. Data source selection – a module that extracts accounts from social network according to social network model (Figure 4.3).
2. Data extraction – a module that extracts data for analysis from accounts.
3. Feature construction – a module that constructs statistical, graph, and graph embedding based features.
4. Classification – decision module.

4.3.1 Technique for group detection of bots

The technique for group detection of bots (Figure 4.15) analyzes a group of accounts and allows one to determine: are there a large number of bots among the analyzed accounts? These accounts must be defined by common action in relation to the object/subject. Therefore, the input is an object or subject. And the output is 0 (there are no bots) or 1 (there are bots).

Data source selection module allows one to extract a group of accounts for analysis. The input is a subject/object. It is necessary to determine the accounts that performed actions in relation to the subject/object. To do this, one can use real graphs of the social network. For example *user* $\xrightarrow{\text{friend}}$ *user*, *user* $\xrightarrow{\text{like}}$ *post*, *user* $\xrightarrow{\text{participate}}$ *group*, and other relations according to model in Figure 4.3. One can also use virtual graphs of the social network. For example graphs that are depicted in Figures 4.6 and 4.7. The output of the module is the list of accounts that are connected in a graph with subject/object (input).

Data extraction module allows one to extract data from the list of accounts. The input is the list of accounts. The module collects for each account from the list: friend list, number of groups, number of subscriptions, number of followers, the number of photos, number of albums, number of posts, user id. From the friend list is extracted the number of friends. The friend graph is extracted from the friend lists in accordance with the scheme shown in the Figure 4.14: each vertex of the graph is present in the list of analyzed accounts plus vertices of mutual friends. The output of the module are:

- distribution of number of friends, groups, subscriptions, followers, photos, albums, posts;
- list of user id (equal to input – list of accounts);
- friend graph.

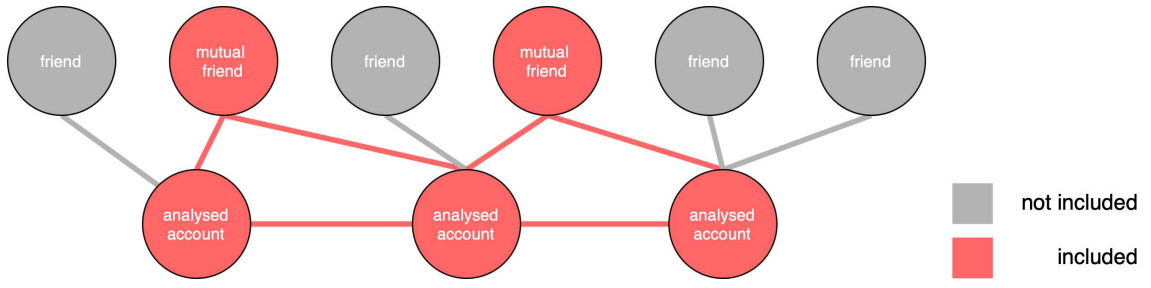

Figure 4.14: Friend graph formation for technique for group detection of bots.

Feature construction construct features from the input of module: distributions, list of user id, and a friend graph. For distribution of parameters we apply feature construction schema based on statistical analysis (Table 4.1), for graph we apply feature construction schema based on graph analysis (Table 4.2). For list of user id we apply eature construction

schema based on graph embedding analysis (Table 4.2). The output of the module are features.

Classification module predicts the class of a group of accounts (1 – bots exist in group, 0 – not exist). To do this we apply Random Forest classification or neural network. The neural network consists of 3 layers: an input layer, one hidden layer with rectified linear activation function (ReLU), and a single output neuron with a sigmoid activation function. The output of the module is 0 or 1.

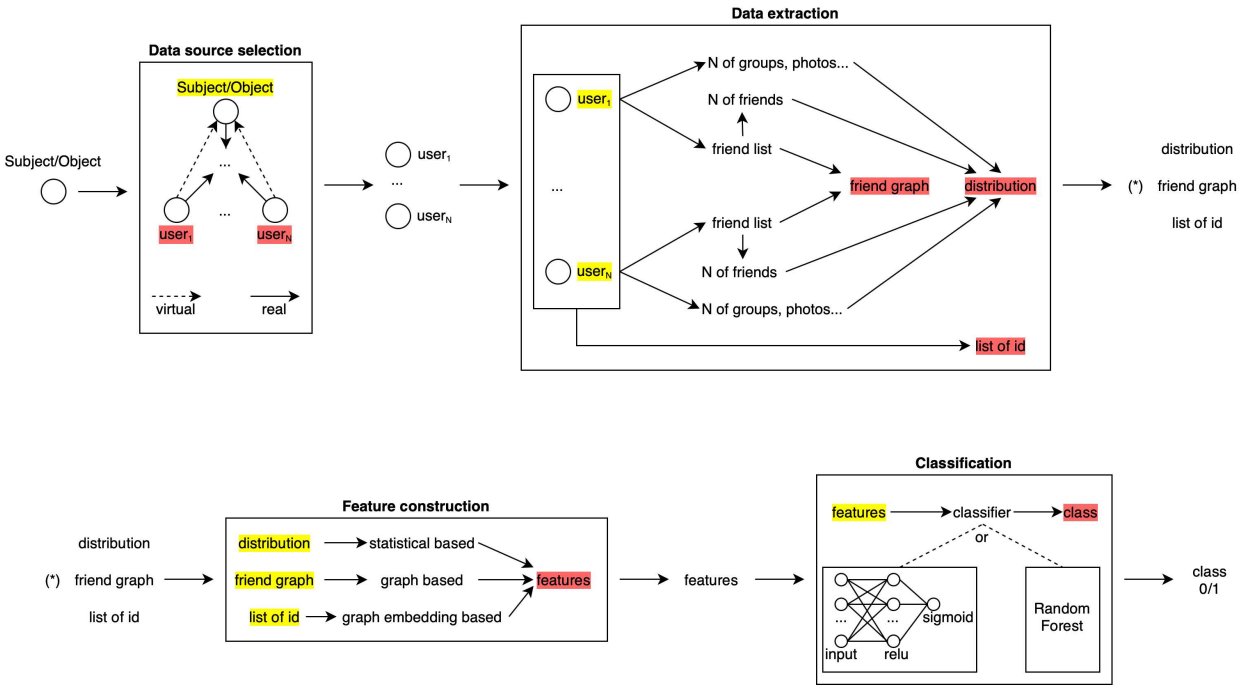

Figure 4.15: Technique for group detection of bots. Inputs of modules are highlighted with yellow, and outputs with red.

4.3.2 Technique for individual bot detection

The technique for individual detection of bots (Figure 4.16) analyzes a single account and determines: is the account a bot? The input is an account. And the output is 0 (a bot) or 1 (not a bot).

Data source selection module. Since the data source (account) is specified in input explicitly, this module is not used.

Data extraction module allows one to extract data from the account. The input is the id of the account. The module collects from this account the friend list (id of friends). For each friend in the list, the module collects the number of groups, subscriptions, followers, photos, albums, posts; friend lists. From friend lists of friends of analyzed account is extracted the number of friends, and friends graph in accordance with the scheme shown in the

Figure 4.8. The output of the module are graph and a number of distributions: distribution of number of friends, groups, subscriptions, followers, photos, albums, posts; list of user id; friend graph.

Feature construction construct features from the input of module: distributions, list of user id, and a friend graph. It works the same way as for the previous technique: statistical based feature construction scheme (Table 4.1) is used for distribution of parameters, graph based feature construction scheme (Table 4.2) is used for friend graph, and graph embedding based feature construction scheme (Table 4.2) is used for the list of friend id. The output of the module are features.

Classification module predict the class of an account (1 – a bot, 0 – not a bot). To do this we apply the same models as for the previous technique: Random Forest classification or neural network (input layer, hidden ReLU layer, and sigmoid output neuron). The output of the module is 0 or 1.

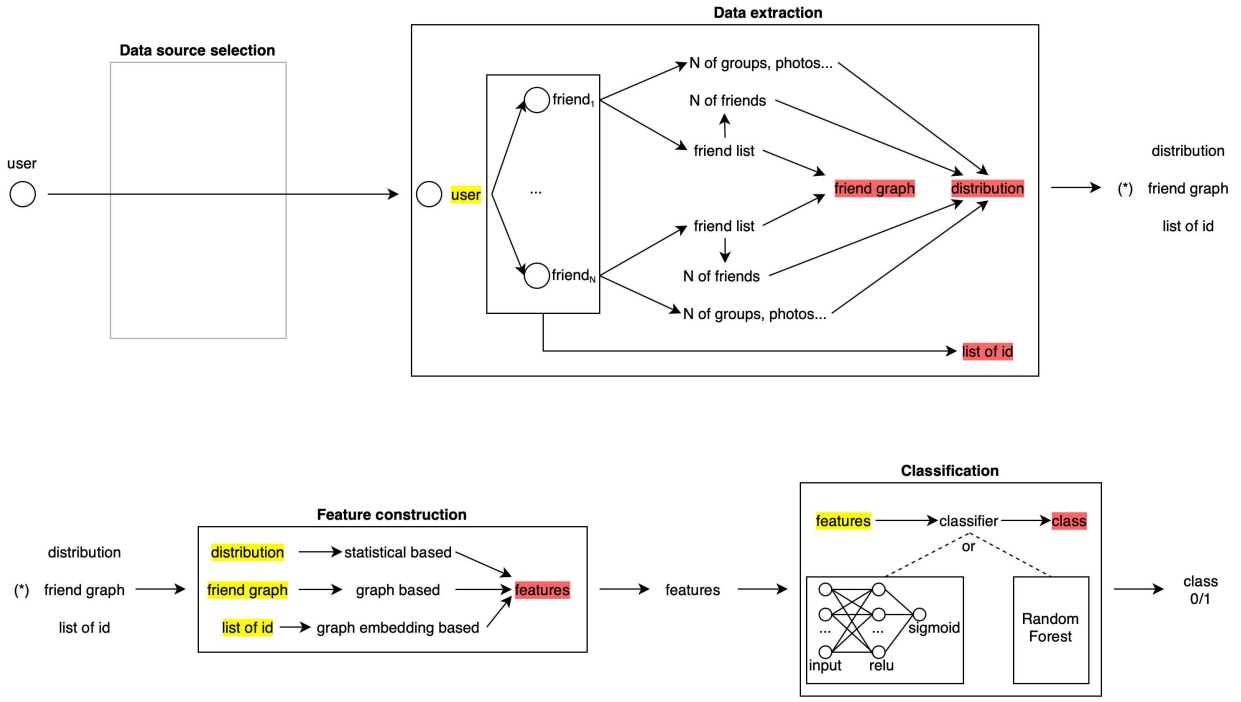

Figure 4.16: Technique for individual bot detection. Inputs of modules are highlighted with yellow, and outputs with red.

4.3.3 Technique for bot's quality estimation

The technique for bot's quality estimation (Figure 4.17) analyzes a single bot and determines its quality in accordance to classification presented in Table 2.1: low, medium, high, or live quality. The input is an account of a bot. And the output is a quality vector with probabilities for each quality class.

Data source selection module. Since the data source (account) is specified in input explicitly, this module is not used.

Data extraction module is identical to data extraction module in individual bot detection technique and extracts next data: distribution of the number of friends, groups, subscriptions, followers, photos, albums, posts; list of user id; friend graph.

Feature construction module is identical to the feature construction module in individual bot detection technique and extracts features.

Classification module predict bot quality (1 – a bot, 0 – not a bot). To do this we apply a neural network that consists of 3 layers: an input layer, one hidden layer with ReLU activation function, and 4 output neurons (one per each class of quality) with softmax activation function. The output of the module is a quality vector with probabilities for each quality class.

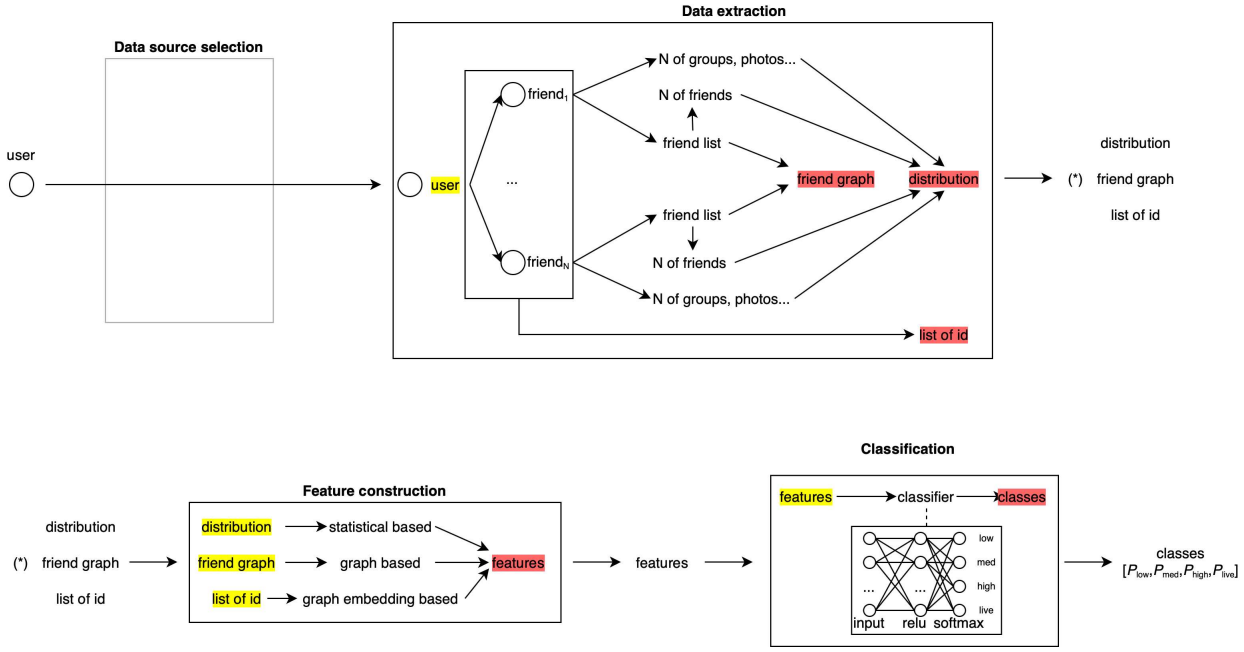

Figure 4.17: Technique for bot's quality estimation. Inputs of modules are highlighted with yellow, and outputs with red.

4.3.4 Technique for bot's price estimation

The technique for bot's price estimation (Figure 4.18) allows one to estimate the cost of attack performed by a single bot (how much the attacker paid for the bot attack). The input is a quality vector with probabilities for each quality class, and bot's action.

To determine the price, we use a table of average prices for bot services based on market analysis. We find prices for an input action (rows). Next, we multiply the price vector

Figure 4.18: Technique for bot's price estimation. Inputs of modules are highlighted with yellow, and outputs with red.

from the table by the input probability vector and calculate the average. The output of the module is the bot's price.

4.3.5 Combination of bot detection techniques

These techniques can be combined into a single toolkit that can be used in different ways based on the tasks of monitoring a social network. We propose the following sequence:

1. Identify the targets of the attack. For this can be used technique for group detection of bots. With this technique, one can quickly (because we analyze many accounts at once) analyze a set of objects/subjects and determine whether an attack has been made on it. For example, the objects of analysis are discussions on a common topic and it is necessary to determine whether bots are present among the commentators. Those discussions that will give a positive class (bots are among the commentators) can be further analyzed by the technique of individual detection.

2. Identify bots. For this can be used technique for individual detection of bots. We suggest using this technique to be able to filter real users from bots. For example, we use the result of the previous technique - discussions in which the activity of bots was determined. Now we can analyze each commentator from these discussions individually and determine if a commentator is a bot.

3. Describe the attack with the complexity of the attack. For this can be used technique for bot's quality estimation. We propose to use this technique only on bots that have been identified by the previous technique. For example, we have identified bots from discussions on a given topic and now we can determine their quality. Based on the quality, one can judge the complexity of the attack. In addition, the attack can be characterized: for example, whether bots of different quality are used in the attack (ex. bots of low quality are liking and bots of high quality are commenting).

4. Describe attack price. We propose to use this technique along with determining the quality of bots. For this can be used technique for bot's price estimation. After calculating the price for each bot, the prices can be added up to get the attack price.

This combination of techniques represents the entire cycle of attack analysis - from target identification to determining the cost of an attack. Depending on the purpose of the

analyst, he/she can use the modules in different combinations. For example, against low quality bots that are usually automated accounts, soft countermeasures (Turing test) will be more effective. Therefore, to assess the effectiveness of countermeasures, it is not necessary to determine the price, but it may be necessary to determine the quality. If the task is to monitor a social network without using countermeasures, one can only use the technique of determining a group of bots to roughly assess the presence of bot activity. If one needs to get a more accurate estimate, one can use only the method of individual detection. Thus, we emphasize by this that the presented sequence is the maximum possible and, depending on goals, the analyst can combine the techniques at his/her own discretion.

4.4 Chapter summary

In this chapter we propose solutions for bot detection problem.

We present the social network semantic model, that used for extraction of adjacent accounts from analysed accounts with the help of graphs. We also present how we will use the extracted graph of friends as main data source for feature construction.

We propose 3 groups of feature construction techniques: (1) statistics based group of features, that includes basic statistics, Benford's law approach, and Gini-index approach to construct features from numerical distributions; (2) graph based group of features, that includes 19 graph algorithms to construct features from graph of adjacent account; (3) graph embeddings based group of features, extracted from the embedding of the graph of random accounts.

We presented 4 techniques for bot detection: (1) technique for group detection of bots, that allows one to determine "are there a large number of bots among the analysed accounts?"; (2) technique for individual bot detection, that allows one to determine "is the analysed account a bot?"; (3) technique for bot's quality estimation, that allows one to determine the bot's quality based on the classification of types of bot's quality; (4) technique for bot's price estimation, that allows one to determine the coast of bot attack. We also present the sequence of bot detection techniques that implements the entire cycle of analysis of bot attack.

In next section we conduct experiments to evaluate efficiency of the proposed approaches and check their ability to detect bots.

Chapter 5

Experimental evaluation of proposed features and techniques

To evaluate the effectiveness of the proposed methods and approaches, we conducted experiments to detect bots on the VKontakte social network. In this chapter, we describe:

1. Bot detection prototype. Using this prototype, we carried out experiments to detect bots on the VKontakte social network.
2. Datasets description. In this section, we describe the collected datasets for training.
3. Feature analysis. In this section, we carry out a correlation analysis of the features to determine which ones are more useful for detecting bots.
4. Experiment plan and results. In this section, we describe how we conducted the experiments and what methods of evaluating the effectiveness we use.

5.1 Bot detection prototype

To conduct experiments, we made a software prototype that implements the entire chain of bot detection pipeline – from collecting data to obtaining a detection result. The architecture of this prototype is presented in Figure 5.1, and it includes 9 modules.

GLASS module. GLASS is an external system for the collection of data from the VKontakte social network. This system was developed in Saint-Petersburg Federal Research Center, and we use it to collect data. In this system, one can specify which accounts and which fields of these accounts need to be scanned. GLASS system used multiple scanners for data collection via social network API. However, increasing the speed can trigger a block of scanner since social networks set a limit on the frequency of requests. GLASS optimize the speed of data collection. Each scanner runs in its own virtual space. So increasing the speed comes down to increasing the number of scanners that work in parallel.

MongoDB database. The GLASS collects the account's data and stores it in the MongoDB database. This database is also an external system.

Mongo to Orient module. This module serves as a proxy that transforms data structures between the external MongoDB GLASS database and our graph database – OrientDB. The module copies unstructured data from MongoDB to OrientDB in order to convert data to a graph in accordance with our semantic model of social network.

OrientDB database. As storage for our system, we select the graph database OrientDB [Com21], as it provides all required functionality of working with graphs and using SQL-like syntax for querying. Also, OrientDB includes a built-in traversal function, a web server that helps to implement necessary API and can be scaled to multiple servers for performance increase. As a database schema, we use a schema that is identical to our proposed model 4.3: it is the hierarchical schema for the VKontakte social network that can aggregate/filter/select vertices and edges using classes and superclasses of the scheme hierarchy.

Data selection module. This module extracts data from OrientDB for analysis. The input is a user ID (in the case of an individual analysis according to technique on Figure 4.16) or a post ID (in the case of a group analysis according to technique on Figure 4.15). The result is either a structure with user data or a structure with a group of users.

Group feature extraction module and **individual feature extraction module**. These modules are used to extract features from data for group and individual detection, respectively. Within these modules, 3 groups of features are extracted – statistical (Table 4.1), graph (Table 4.2) and global graph features (Table 4.3). The calculation of statistical features based on Benford's law occurs in R.

Group of bots detection module, **individual bot detection module** and **bot's quality estimation module**. These modules use a trained random forest model or neural network to predict whether there are bots among users, whether the user is a bot, or what quality the bot has, respectively.

Bot's price estimation module. The module estimates the price of a bot based on its quality and performed action. The relationship between bot quality, action and price was demonstrated by us in our research of the bot market. This module also requires a **table of average prices** for bots of different quality.

Data visualization module. We used this module to be able to look at the graphs of users and bots during model training. The module can visualize graphs in two ways. The first method is visualization in the form of a force layout (for this we use JavaScript and D3.js). The second way is to visualize the embedding of the graph (using a chain of Word2Vec, UMap, and Datashader, as we did when forming a global graph according to the pipeline on Figure 4.11).

All modules, except external ones, are mostly written in Python. To implement machine learning models we used the Keras and scikit-learn libraries. For random forest, we performed a simple grid search hyperparameter optimization. In the neural network, we used one hidden layer, and the output layer was a single neuron with a sigmoid activation function for group and individual detection, or softmax activation function for quality estimation. In the neural network, we used optimization for the hidden layer size and the number of epochs. For features also used min-max scaling.

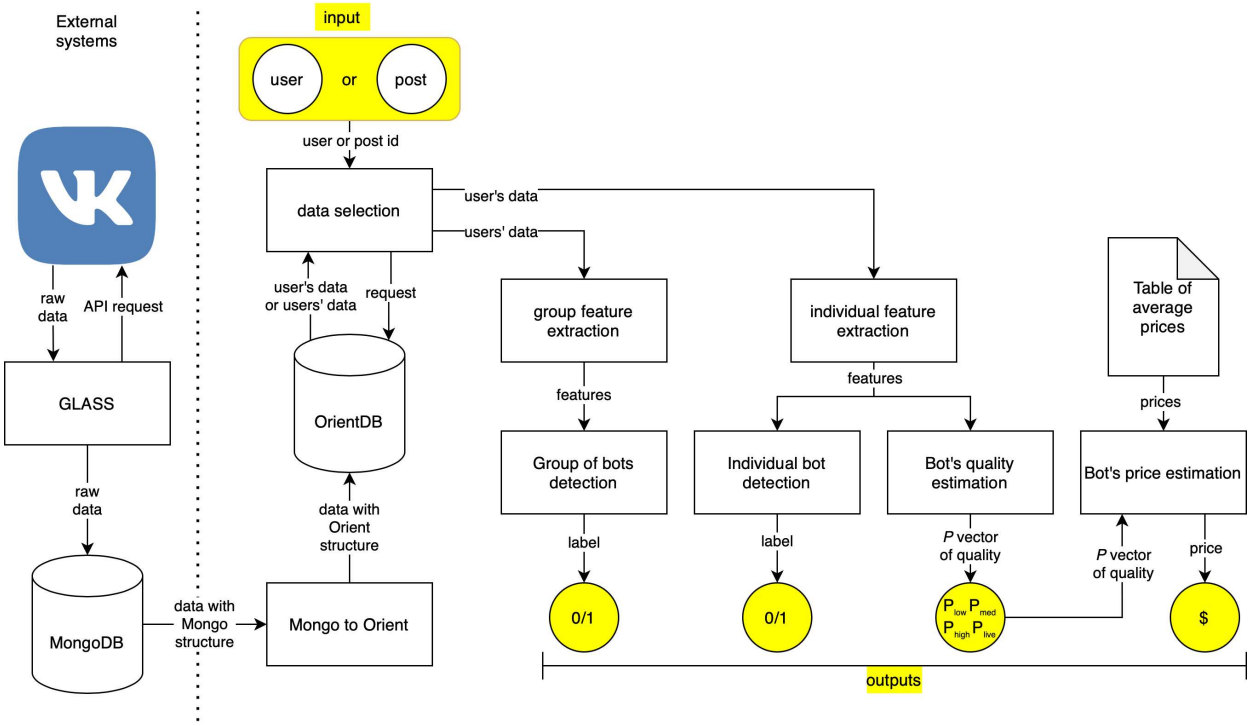

Figure 5.1: Software prototype that implements the bot detection pipeline.

5.2 Data collection

To conduct experiments, we need data about bots and real users of the social network. We decided to collect this data ourselves, because the existing datasets do not satisfy us for a variety of reasons, including:

1. We cannot be sure that bots in existing datasets are really bots (or we cannot verify this). If we train on such datasets, we will get incorrect indicators of detection efficiency. We divide such datasets into the following categories:
   (a) Datasets in which bots are labeled by a human. Such datasets are the most unreliable because humans' ability to detect bots is questionable. This is especially true for high quality bot detection when bots are very similar to real users. Besides, users can simply label real people they don't like as bots. Also, if we train on such datasets, we will try to reproduce the analyst's subjective decisions.
   (b) Datasets in which bots are defined based on the research context. For example, when a bot is defined as an account that promotes radical political views, behaves aggressively, etc. We do not believe that all accounts that use aggressive promotion and all conflicting users are bots – malicious activity can be carried out by real users too.
   (c) Datasets in which bots are accounts blocked by a social network. Social networks block accounts only when there is obvious abnormal activity. Learning on such datasets can make it impossible for the model to detect sophisticated bots.
   (d) Datasets in which bots are labeled using a different bot detection system. Learning on such datasets will never exceed the effectiveness of this bot detection system.

2. In datasets, there is no division of bots by quality or strategy. If we learn from such datasets, we will not understand what types of bots our system can detect.

3. In datasets, there is no division of bots by price. If we use such datasets, we can not estimate the efficiency of the bot's price estimation.

For this reason, we decided to collect bots in the only reliable way – we bought bots from bot traders (see Table 5.1).

5.2.1 Collection of bots and real users for individual detection

To collect bots we found 3 companies that provide promotion services in social networks using bots. Those companies have different strategies on how to manage bots. Each company also provides different bot quality:

1. Vtope company – provide bots. Provide 3 types of quality: standard quality bots (dataset bot 1), high-quality bots (dataset bot 2), and live users (dataset bot 3).

2. Martinismm company – provide bots. Provide 3 types of quality: standard quality bots (dataset bot 4), high-quality bots (dataset bot 5), and live users (dataset bot 6).

3. Vktarget company is an exchange platform, where one can describe the type of promotion and real users perform it for money. The seller provides 2 types of quality: standard quality and high-quality. But since these are users from the exchange platform, we attributed them to a live quality (dataset bot 7 and 8).

So we create 3 groups on VKontakte social network (one for each company) to buy bot promotion. During the week we filled these groups with content because companies refused to provide promotion services to empty groups. We made those groups absurd to make sure that during experiments real users will not accidentally join the group and we will collect only bots. The first group was dedicated to "Wigs for hairless cats" (left side on Figure 5.2). The second – "Deer Registration at the Ministry of Nature" (center on Figure 5.2). The third – "Minibus from Jogeva to Agayakan (right side on Figure 5.2)". During the experiments, only one real user joined the group and we exclude him from the dataset.

For each company we applied the following algorithm: create a post in a group → buy 300 likes under the post from bot-trader with an indication of what quality bots we need → collect accounts that liked the post (only ID of account) to the file → delete the post. After iteration, we perform it again with bots of another quality. Thus, we did not mix bots of different quality and different companies.

We also checked the list of accounts that liked our posts against the list of bots that bot sellers provided as proof of work done. Thanks to these lists, we additionally verified that regular users were not included in our bot dataset.

Dataset	Type	Company	Count	Descriptions
bot_1	LOW	C	301	Probably bots that are controlled by software.
bot_2	MED	C	295	Probably more user-like bots that are controlled by software.
bot_3	HIGH	C	298	Probably bots that are controlled by human or users from exchange platform.
bot_4	LOW	B	301	Probably bots that are controlled by software.
bot_5	MED	B	303	Probably more user-like bots that are controlled by software.
bot_6	HIGH	B	304	Probably bots that are controlled by human or users from exchange platform.
bot_7	LIVE	A	302	Probably users from the exchange platform.
bot_8	LIVE	A	357	Probably users from the exchange platform with more filled profiles.
user_1	activists community		385	Users from group "velosipedization" that is dedicated to development of bicycle transport.
user_2	mass media community		298	Users from group "belteanews" that is dedicated to Belarussian news.
user_3	developers community		332	Users from group "tproger" that is dedicated to software development.
user_4	sport community		224	Users from group "mhl" that is dedicated to youth hockey.
user_5	mass media community		420	Users from group "true_lentach" that is dedicated to Russian news.
user_6	blogger community		251	Users from group "mcelroy_dub" that is dedicated to re-playing of funny videos.
user_7	commerce community		284	Users from group "sevcableport" that is dedicated to creative space.
user_8	festival community		259	Users from group "bigfestava" that is dedicated to cartoon festival.
user_9	sport community		181	Users from group "hcakbars" that is dedicated to fun community of hockey club.
user_10	developers community		397	Users from group "tnull" that is dedicated to software development and memes.

Table 5.1: Description of a dataset for experiments, which includes bots of different quality from different companies as well as real users.

For Vtope and Martinismm companies the whole process for one post took one day, because the companies were afraid to give likes too quickly, and they are trying to simulate natural growth. For Vktarget company collection took 3 days because Vktarget provides not bots, but referrals. "Referrals" are real users (not operated by a program) who registered on the bot exchange platform and give likes, comments, and other activities for money. Thus, we can be 100% sure that we have collected only bots.

To collect real users we select 10 random posts in different groups. We selected posts that have around 250 likes (datasets user N in Table 5.1) and collect accounts that liked the post (only ID of account). We have chosen such a collection strategy for real users so that bots and real users perform the same action – like the post. But we cannot be sure that there are 100% only real users.

Thus, we got bots that are divided by company (each company has its own management strategy) and quality, and also do not contain false-positive real users (users who were labeled as bots).

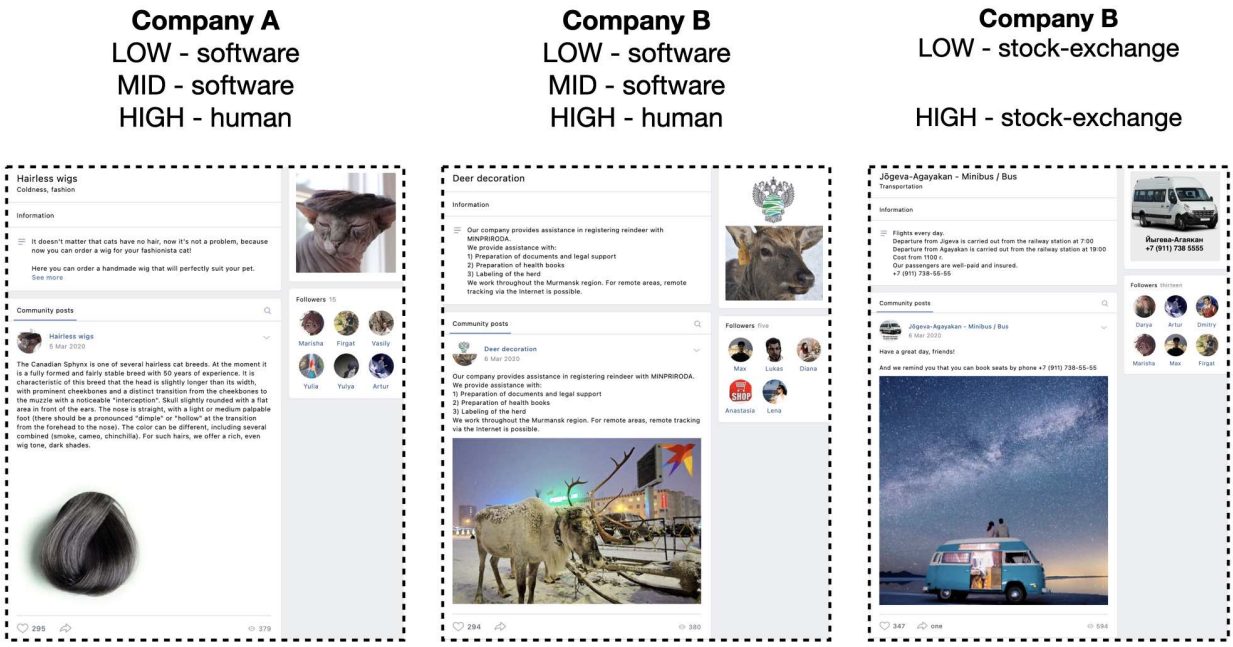

Figure 5.2: 3 fake groups that we created to collect bots. Groups in Russian, the snapshots show their auto-translation.

5.2.2 Dataset generation for detection of a group of bots

Collecting datasets for the detection of a group of bots became another challenge. To do this, it was necessary to collect some sets of accounts – groups with bots and groups without bots. To train and test the model, a large number of such sets with different bot-to-user ratios and different sizes of sets are required. It would be expensive, time-consuming, and risky to buy them from bot seller companies (we didn't want bot sellers to know that we are fake buyers). Therefore, we decided to generate such sets from the list of individual accounts that were already collected.

We have implemented a generator that accepts the following parameters as input:

1. Group size – the number of accounts that should be in a group to check ability for detection of different sizes of groups. The smaller the group size, the more combinations we can generate in which the accounts will have a small intersection. For us, the intersection of no more than 30% is acceptable.

2. Ratio of bots and real users – to check ability for detection when there are many or few bots among users.

The schema of generator is presented in Figure 5.3. And it includes the following steps:

1. Divide datasets that are presented in Table 5.1 to train and test and user and bot samples. The result is sublists of datasets (that are in 5.1) for training and testing.
2. For each bot dataset we select several random bots equal to the required number (group $size * ratio$). The result is the sublist with the required number of bots. We generate $\frac{length(dataset)}{size*ratio}$ such sublists.
3. For each bot sublist we select a random user dataset. For this dataset, we select several random users equal to the required number (group $size * (1 - ratio)$). The result is the sublist with the required number of users.
4. We unite bot and user sublists. As result, we have the group of accounts of required size and bot-to-user ratio.
5. To generate groups without bots we do the pipeline that is similar to step 3. For each united group, we select a random user dataset. For this dataset, we select several random users equal to the required number (group $size$). The result is the group with the required number of users and without bots.

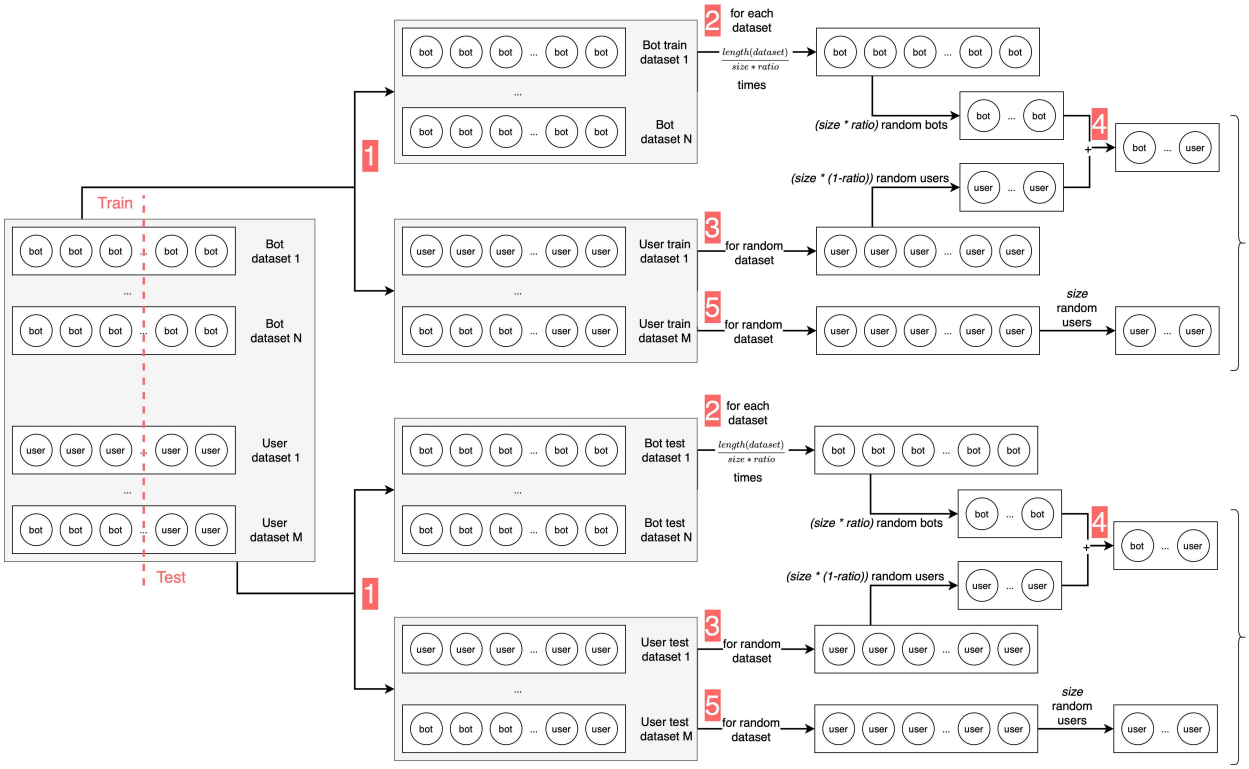

Figure 5.3: Schema for generation groups with bots and groups without bots for training and testing with specific bot-to-user ratio and group size.

Such generation schema allows us to generate groups with variable group size and bot-to-user ratio. So we can check how efficiency scores are distributed over the groups with different sizes and ratios.

5.2.3 Summary of the collected data

As the result, we have 5492 accounts – 2461 bot accounts and 3031 user accounts. Bot accounts are labeled with bot quality (LOW, MED, HIGH, or LIVE) and bot company (A,B,C). User accounts are labeled with VKontakte group's name from each they were collected. The summary of collected accounts is presented in Table 5.1.

We also generate 40806 groups from 5492 accounts – 20403 groups with bots and 20403 groups without bots. Groups are labeled with group size and bot-to-user ratio. When generating, we reduced the number of generated groups as the ratio and group size increased so that the same accounts would not be repeated frequently in different groups. The summary of generated groups is presented in Figure 5.1.

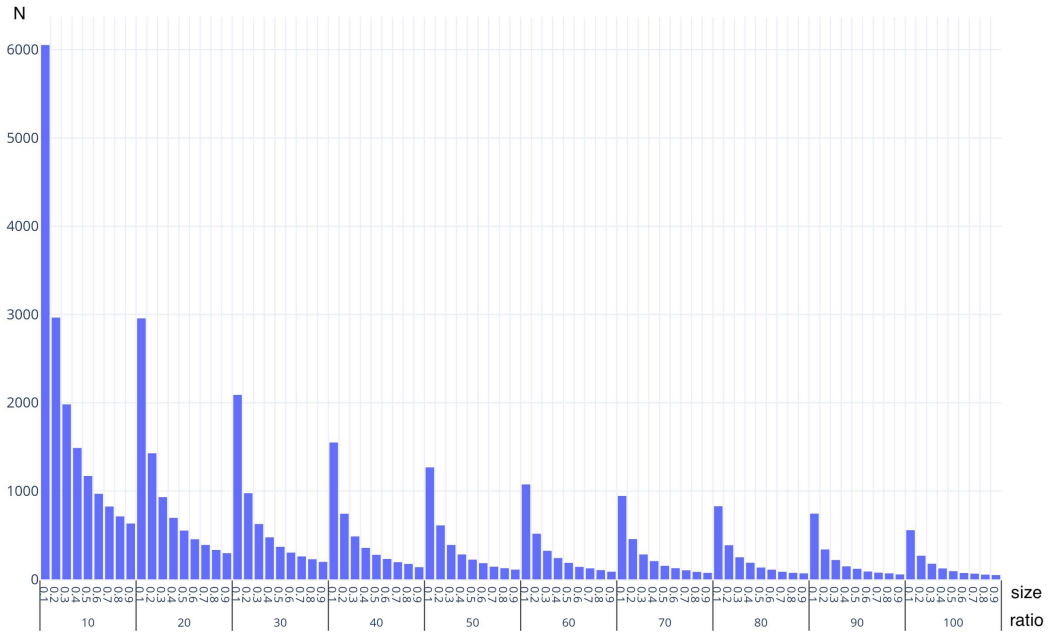

Figure 5.4: Distribution of the number of generated groups over ratio and group size.

Also, we have the actual price of the bot since we bought them. This price does not reflect the cost of selling the bot (how much the seller is willing to sell it), but the cost of renting the bot (how much the seller is ready to perform a certain action). In other terms, this reflects the cost of the attack — how much the attacker paid.

Thus, we have the data necessary to conduct experiments. We used bot and user accounts in individual detection and quality detection. We used generated groups in the detection of a group of bots. And we used bot accounts with their prices for price estimation.

5.3 Feature analysis

One of the stages of the experiment is to evaluate the usefulness (informativity) of features for detecting bots. We carry out 2 stages of this analysis:

Analysis of multicollinearity of features. If features are linearly correlated with each other, this means that one feature can be predicted from the value of the other. In such features, there is no benefit in training, since they duplicate each other. Therefore, from the set of found multicollinear features can be removed all features except one, thereby simplifying the model and increasing the calculation speed.

In this analysis, we use Pearson's correlation, since features are represented by a continuous value in the range from 0 to 1. For each feature, in turn, we check with which other features it has a strong correlation > 0.9 and leave only one feature from this set, which has the fastest computation speed. To do this, we also calculated the average calculation time for each group of features that are presented in Figure 5.5.

It should be noted that the implementation also influenced the computation speed that is presented in Figure 5.5. Benford's law is slower to execute than other algorithms, since it is calculated in R, while the rest of the code is written in Python. The running time of other algorithms largely depends on the size of the graph. We present the average values that were obtained after calculating many graphs of friends of different sizes.

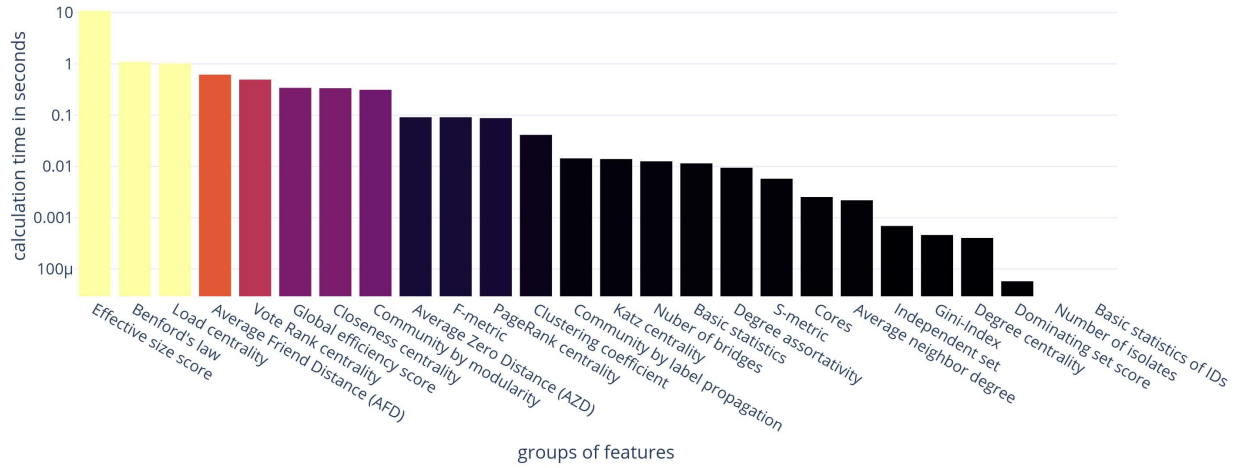

Figure 5.5: Average calculation time for each group of features.

Feature and label (bot/not bot) correlation analysis. This analysis allows one to identify features that are correlated with the label, thereby highlighting features that are more informative for detecting bots. It is worth noting that this is not always true. This test evaluates the informativity of a feature as the ability of a feature to divide the sample into 2 classes, while there are situations when features alone are uninformative, but they can be informative in various combinations. Therefore, such an assessment of informativity is indirect. In this analysis, we use Spearman's correlation because the label is a rank variable.

We also provide a calculation of the informativity of groups of new features (Benford's law, Gini index and graph embeddings). To do this, we use a feature selection approach based on machine learning models, in which we train the model on a group and interpret the classification efficiency as the informativity of the features.

We analyze for two datasets: a dataset with features of groups of accounts and a dataset with features of individual accounts.

Feature analysis of individual accounts. In the first step, we remove multicolinear features. For individual accounts, we removed 50 features out of 138. The final modulo correlation matrix is shown in the Figure 5.6. In the second step, we highlight features that correlate with the label ≥ 0.1. The final chart of features informativity is shown in the Figure 5.7.

Feature analysis of groups of accounts. In the first step, for groups of accounts, we removed 49 features out of 137. The final modulo correlation matrix is shown in the Figure 5.8. In the second step, we highlight features that correlate with the label ≥ 0.1. The final chart of features informativity is shown in the Figure 5.9.

5.3.0.1 Proposed features informativity.

Lets look at the informativity of the new features that we have proposed.

Benford's law based group of features has a low informativity in comparison with other features. The best informativity for individual accounts have *benf_followers* with informativity ≈ 0.09. For groups of accounts *benf_groups* feature have informativity ≈ 0.06.

Gini index based group of features has a high informativity. *gini_gr* have the 9th place in informativity chart for individual accounts with score ≈ 0.4, and 5 other features that are based on Gini index have informativity *>0.1*. For groups of accounts *gini_subscr* feature is in the 5th place in informativity chart, and 3 other Gini index based features have informativity *>0.1*

Also for individual accounts 4 AZD based features, 2 AFD based features and F-metric feature have informativity *>0.1* with highest score of ≈ 0.45 for F-metric. For groups of accounts only F-metric have informativity *>0.1*.

This analysis of informativity is based on Spearman's correlation and does not take into account the informativity of features in pairs or groups. To analyse a group of features, we additionally present the results of informativity analysis based on the use of machine learning models. The selection of features based on machine learning models is the informativity calculation technique, were one group of features is selected, the model is trained on it, and the classification efficiency is interpreted as the informativeness of the features. In Table 5.2 are presented informativity based on AUC-ROC score of random forest classifier for the

following groups of features: Benford's law, Gini index, and graph embedding analysis based features (AFD, AZD and F-metric).

group of features	individual accounts (AUC-ROC)	group accounts (AUC-ROC)
Benford's law	0.692	0.548
Gini index	0.820	0.728
Graph embedding analysis	0.640	0.522

Table 5.2: Informativity for groups of features based on the machine learning models, in which AUC-ROC is interpreted as informativity.

Thus, it can be indirectly established that the proposed features are effective for detecting individual bots. However, for detecting groups of bots, only the Gini index features are highly effective. Nevertheless, we include features with low informativity in the vector of features, because it is possible that an other combination of low informativity features can give high informativity. But we made sure that, in general, the proposed features can be used in model training for bot detection.

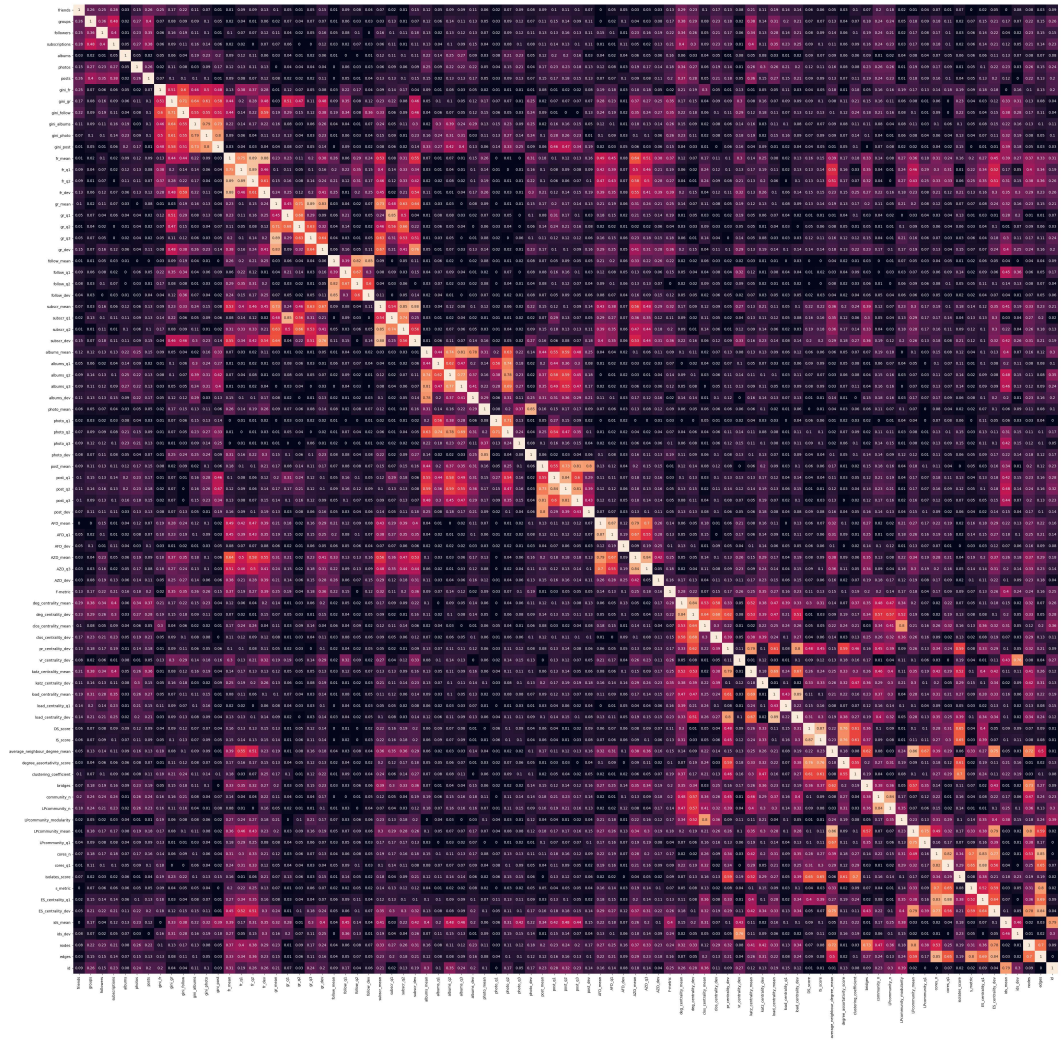

Figure 5.6: Modulo < 0.9 correlation between features for dataset of individual accounts.

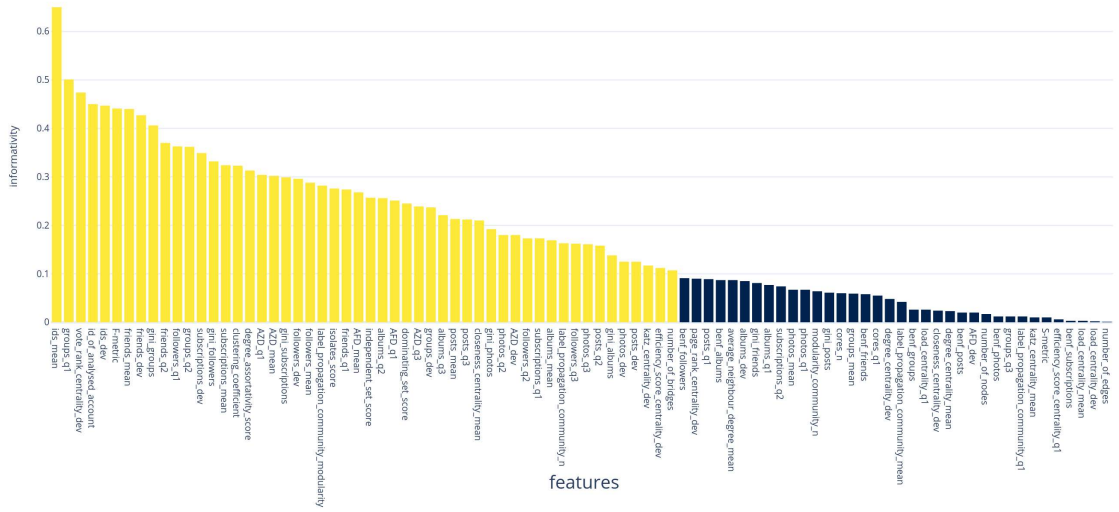

Figure 5.7: Features informativity for dataset of individual accounts, where features that have a correlation with the label ≥ 0.1 are highlighted with yellow.

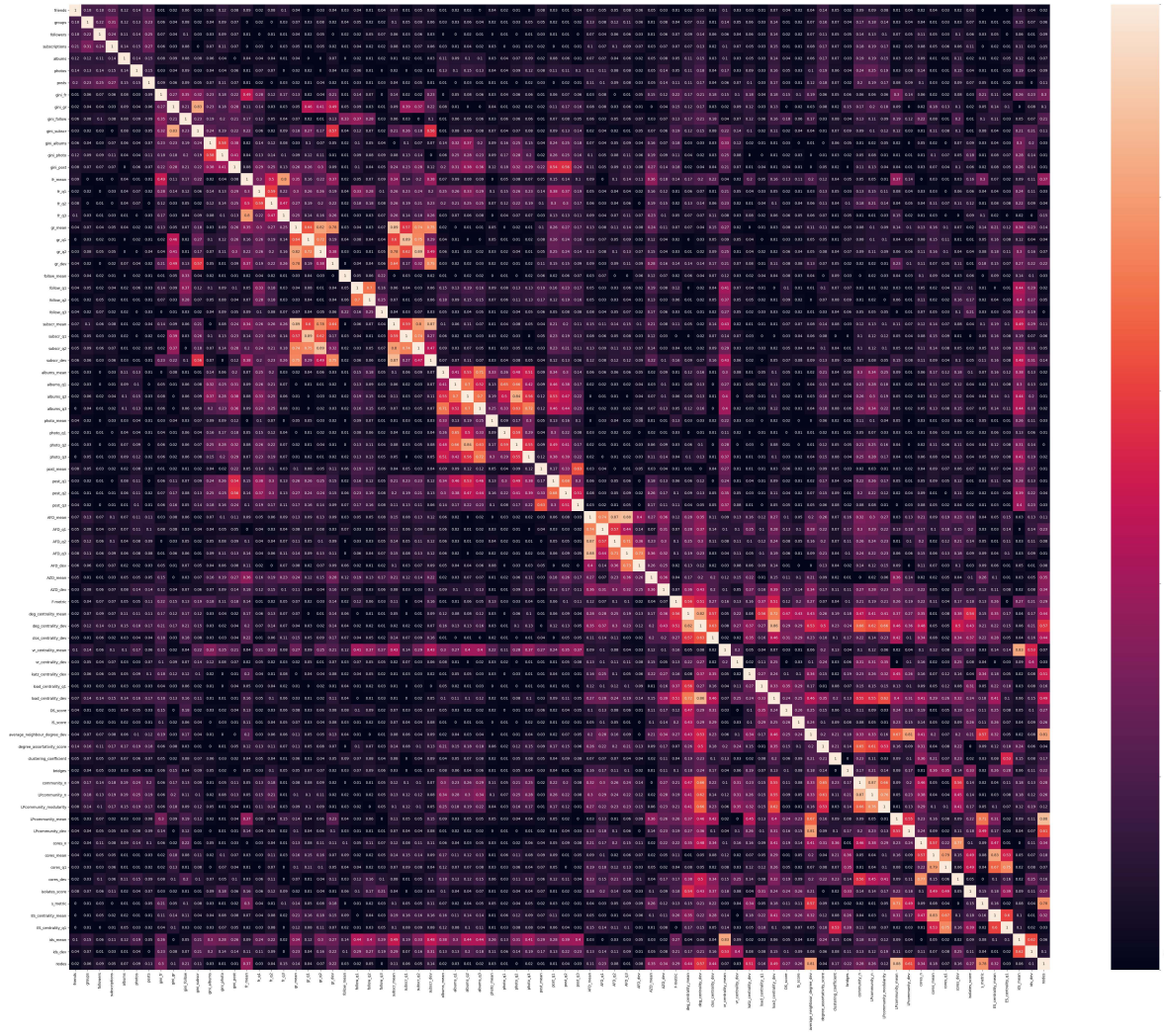

Figure 5.8: Modulo < 0.9 correlation between features for dataset of groups of accounts.

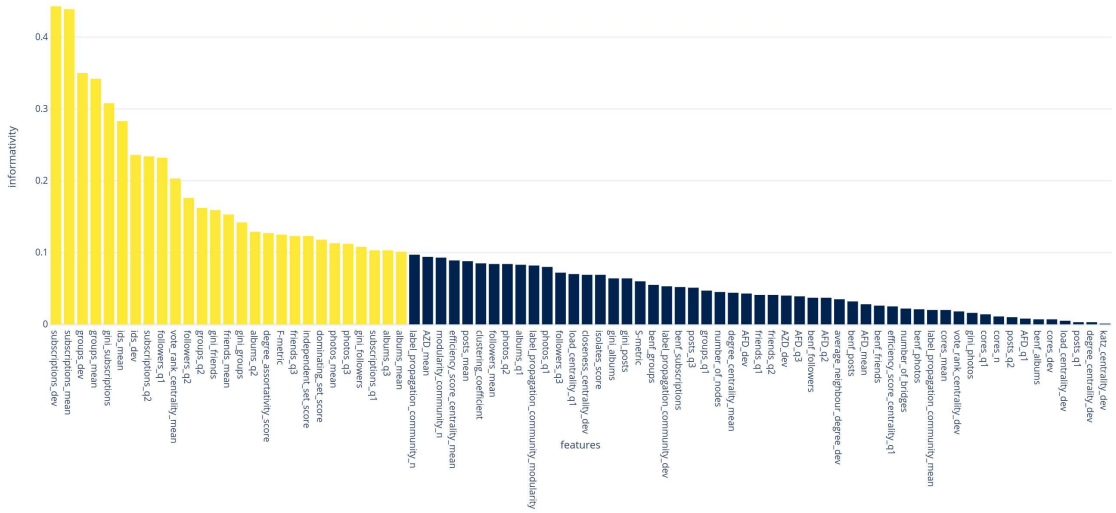

Figure 5.9: Features informativity for dataset of groups of accounts, where features that have a correlation with the label ≥ 0.1 are highlighted with yellow.

5.4 Experiment plan

In experiments, we evaluate the efficiency of recognizing a group of bots, individual recognition of a bot, recognition of the quality of a bot, and recognition of the price of a bot. To do this, we conduct the following series of experiments:

1. Detection of a group of bots tests. We generated a dataset in which groups are divided by bot-to-user ratio and group size. This allows us to assess the ratio and size threshold for training and testing, where the proposed detection technique would be efficient. That's why we propose efficiency tests according to multiple schemes:
   (a) Standard test. In this test, models were learned on all training samples and testing in all validation samples and samples over group's size and bot-to-user ratio.
   (b) Standard test with a threshold. In this test, models were learned on a training sample that was filtered with a minimum ratio. We test in all validation samples, validation samples that were filtered with minimum ratio, and samples over group's size and bot-to-user ratio. This test allows us to get an estimate of the effectiveness when we limit our model – when we say that our model detects bots only if they have reached a certain critical mass in the group (some critical threshold).

2. Individual detection tests. We collected a dataset in which bots are divided by quality and by company. This allows us to assess how resistant the proposed bot detection approaches are to new types of bots. That's why we propose efficiency tests according to multiple schemes:
   (a) Standard test. Tests in which the division into training and validation samples was random across the entire dataset. This means that the training and validation set included bots of all companies and all qualities. This test evaluates the overall performance of individual bot detection.
   (b) Standard test with feature selection. Standard tests in which features with very high correlation were removed. This test will allow one to determine whether the dimension decreasing of the feature set affected the quality of the classification.
   (c) Bot's quality resistance test (cross quality $validation\_sample = LOW/MED/HIGH$). The training sample contains bots of one quality, and the validation sample contains another. Such a test shows the resistance of the method to the quality of bots. Both samples also include users.
   (d) Bot's management strategy resistance test (cross-company $validation\_sample = A/B/C$). The training sample contains bots of one company, and the validation sample contains another. Such a test shows the resistance of the method to the management strategies that differ from one bot company to another. Both samples also include users.

3. Quality estimation tests. In this test, we evaluate the bot quality recognition efficiency. We calculate the bias of the predicted quality to the real one on our quality scale.

4. Price estimation tests. In this test, we evaluate the efficiency of the bot's price recognition. We simply compare the price we paid for the bots when collecting datasets to the price that our model predicts.

In the detection of a group of bots tests and individual detection tests we propose to evaluate the effectiveness of bot detection using the following metrics:

1. Area under ROC-curve (AUC-ROC) – the common integrated measure for binary classification. We suggest using it as the main measure of efficiency in tests that solves binary classification problem.
2. F1 score – a weighted average of the precision and recall that is very popular in papers dedicated to bot detection.
3. Precision – give an understanding of false positives, which is extremely important for social network protection systems, as it assesses how strong countermeasures can be applied. For example, to automatically block or freeze pages, one needs to have an extremely low number of false positives.
4. Recall – describes the ability to detect bots.

In some tests we provide overall score – we summarize efficiency metrics over different types of samples as average with standard deviation. This situation occurs in cross tests in individual detection (cross quality and cross-company tests), as well as in the detection of a group of bots tests when we summarize the scores across groups of different sizes and with different ratios of bots.

For group detection of bots, we also show the distribution of efficiency ratings by group size and by the ratio of bots.

For quality estimation tests and price estimation tests, we developed our efficiency measures, which are described in relevant subsections below.

5.4.1 Detection of a group of bots tests

For the detection of a group of bots, we conduct 2 types of tests. In these tests, we use groups of accounts that we generate from bot and user accounts.

The purpose of these experiments is to evaluate the overall detection efficiency of a group of bots, as well as the detection efficiency when we constrain the model:

1. Standard test. The first experiment evaluates overall efficiency.
2. Standard test with the threshold. In the second experiment that is devoted to the limited model – we say that it is not so important for us to detect bots if there are very few of them in the group, but we don't want a lot of false positives. Thus, we make relaxation for our model in cases when it is very difficult and not very necessary to recognize presence of bots.

5.4.1.1 Standard test

In the standard test, models were learned on all groups with different ratios and sizes. The scores for random forest and neural network classifiers over ratio and size are presented in Figure 5.10. In Figure 5.10 we don't provide F1 and accuracy as they are very similar to AUC ROC. Precision and recall are the most important scores here, as they allow one to evaluate false positives and the ability of the model to recognize the presence of bots.

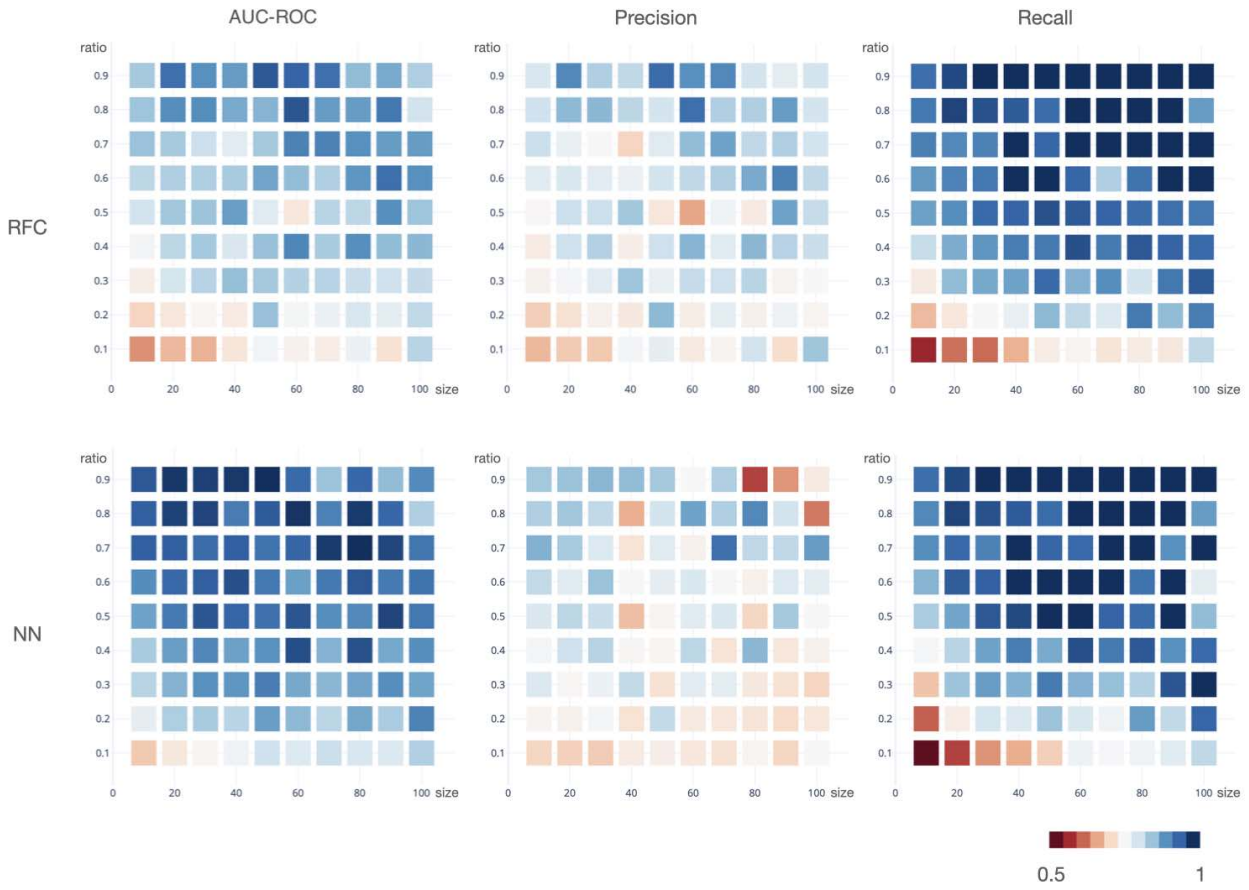

Figure 5.10: Distribution of AUC ROC, precision, and recall in the detection of a group of bots standard tests over ratio and group size.

Since the number of groups for different ratios and sizes differs (see Figure 5.4), we consider the overall score as the score's mean plus minus standard deviation (see Table 5.3).

Model	Accuracy	Precision	Recall	F1	AUC-ROC
RFC $\overline{x} \pm \sigma$	0.832 ± 0.071	0.795 ± 0.062	0.900 ± 0.107	0.842 ± 0.075	0.831 ± 0.070
NN $\overline{x} \pm \sigma$	0.808 ± 0.076	0.769 ± 0.066	0.895 ± 0.114	0.817 ± 0.082	0.902 ± 0.068

Table 5.3: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the detection of a group of bots standard tests.

5.4.1.2 Standard test with the threshold

As we can see from the results of standard tests on Figure 5.10, scores are low for groups with a low ratio of bots to users. Therefore, we introduce a condition that the model can detect presence of bots if they have reached a certain critical threshold in the group.

In this test, models where learned only on the groups in which the bots have reached a ratio of 0.4 and higher. We chose this threshold based on the results of the standard test, which shows that with a ratio of less than 0.4, the scores are significantly worse. The scores for the standard test with the threshold are presented in Figure 5.11.

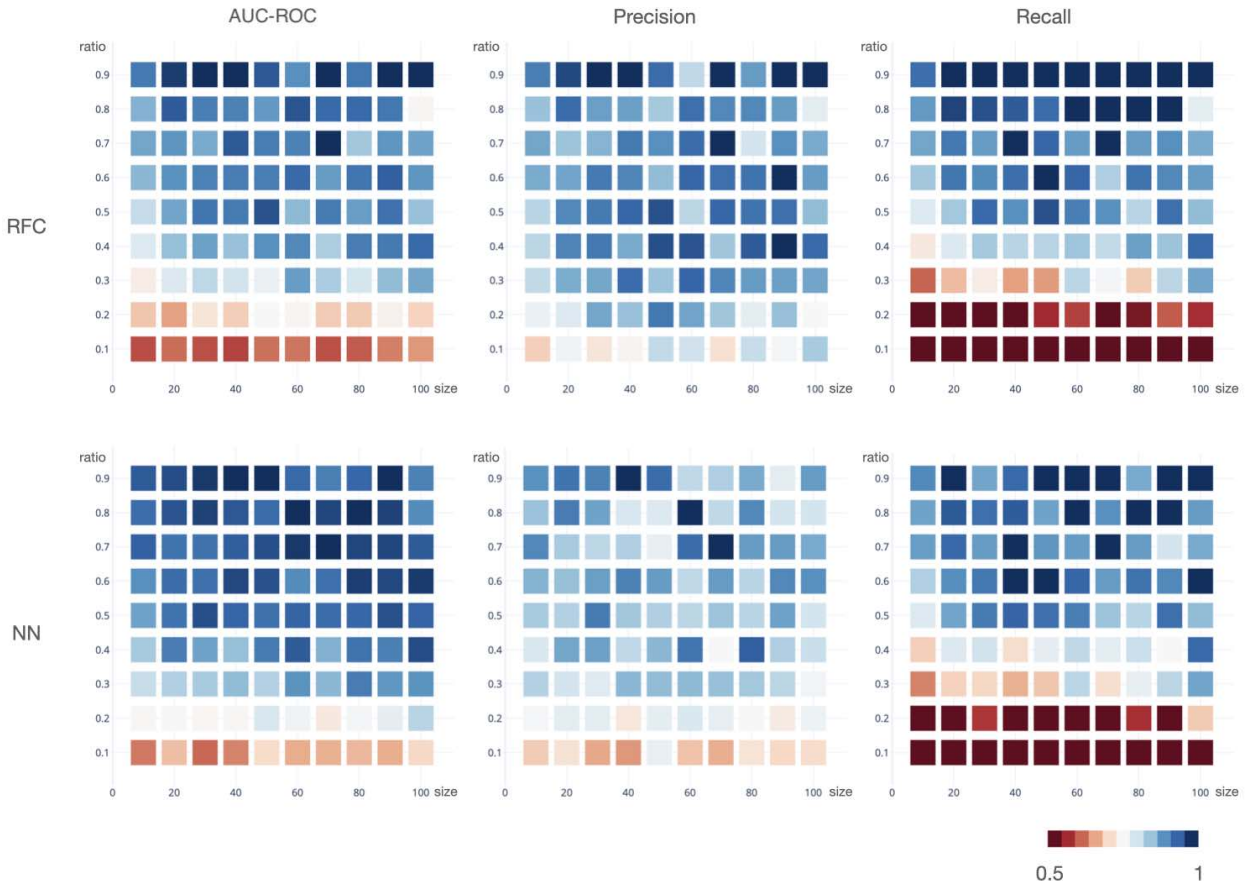

Figure 5.11: Distribution of AUC ROC, precision, and recall in the detection of a group of bots standard tests with the threshold over ratio and group size.

As one can see in Figure 5.11, we almost completely lose the ability to detect presence of bots when the ratio of bots to users is low, but we have significantly reduced the rate of false positives in all types of groups in comparison with standard tests in Figure 5.10. Thus, we say that the model can identify presence of bots only if there are more than 30% bots in the group.

Average estimates are presented in the Table 5.4. We present 2 estimates that were obtained when applying the threshold in training – for all groups and groups with a ratio

greater than 0.3.

Model	Accuracy	Precision	Recall	F1	AUC-ROC
RFC $\overline{x} \pm \sigma$	0.843 ± 0.122	0.884 ± 0.073	0.781 ± 0.229	0.814 ± 0.175	0.844 ± 0.120
NN $\overline{x} \pm \sigma$	0.812 ± 0.115	0.830 ± 0.078	0.776 ± 0.219	0.782 ± 0.167	0.886 ± 0.108
RFC $ratio > 0.3$ $\overline{x} \pm \sigma$	0.913 ± 0.053	0.913 ± 0.056	0.917 ± 0.071	0.913 ± 0.053	0.913 ± 0.053
NN $ratio > 0.3$ $\overline{x} \pm \sigma$	0.878 ± 0.053	0.865 ± 0.057	0.902 ± 0.080	0.873 ± 0.070	0.950 ± 0.037

Table 5.4: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the detection of a group of bots standard tests with the threshold for all validation samples and validation samples with $ratio > 0.3$.

5.4.2 Individual detection

For individual detection, we conduct 4 types of tests. In these tests, we use individual accounts that we collected. The purpose of these experiments is to evaluate the overall detection efficiency of individual bot detection, as well as the detection efficiency when we meet a new type of bot:

1. Standard test. The first experiment evaluates overall efficiency.
2. Standard test with feature selection. This experiment evaluates efficiency when we remove multicollinear features and features that have no linear correlation with the label.
3. Bot's quality resistance test. Simulates recognition of bot of new quality.
4. Bot's management strategy resistance test. Simulates recognition of bot of a new company.

Standard test.

This test is a classic one in data science for evaluating the effectiveness of models. We randomly divided the sample into training and validation samples at a ratio of 0.7 and 0.3 to the original sample. The results are presented in Table 5.5.

Model	Accuracy	Precision	Recall	F1	AUC-ROC
RFC	0.864	0.892	0.796	0.842	0.859
NN	0.864	0.881	0.806	0.843	0.919

Table 5.5: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in standard test.

Standard test with feature selection.

In this test, we check if our efficiency scores will drop if we remove multicolinear features that have a linear correlation with other features *geq* 0.9. These features have been identified in the Feature Analysis section for the dataset of individual accounts. The results are presented in Table 5.6.

Model	Accuracy	Precision	Recall	F1	AUC-ROC
RFC	0.892	0.923	0.710	0.802	0.841
NN	0.883	0.909	0.695	0.787	0.923

Table 5.6: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in standard test with feature selection.

Bot's quality resistance test.

In this test, we check the case where our model is used with a new type of bots. The idea is that we *include only one type of bots in the validation set* and *exclude this type of bots from the training set*. Thus, the model is trying to recognize bots that it hasn't trained on.

This situation simulates the mutation of bots when bots change qualitatively and we first encounter them. In total, we have 4 types of bots, divided by quality. Thus, we conduct 4 tests, in each of which one type of bot is excluded from the training sample. In Table 5.7 we give the final estimate as the average of all estimates with their standard deviation.

Model and split strategy	Accuracy	Precision	Recall	F1	AUC-ROC
RFC test=LOW	0.910	0.852	0.935	0.891	0.914
RFC test=MID	0.917	0.757	0.956	0.845	0.930
RFC test=HIGH	0.772	0.825	0.584	0.684	0.747
RFC test=LIVE	0.783	0.911	0.601	0.724	0.774
RFC $\overline{x} \pm \sigma$	0.846 ± 0.078	0.836 ± 0.064	0.769 ± 0.204	0.786 ± 0.098	0.841 ± 0.094
NN test=LOW	0.830	0.709	0.970	0.389	0.955
NN test=MID	0.870	0.649	0.987	0.239	0.975
NN test=HIGH	0.770	0.846	0.555	0.308	0.787
NN test=LIVE	0.737	0.851	0.538	0.336	0.846
NN $\overline{x} \pm \sigma$	0.802 ± 0.06	0.764 ± 0.101	0.763 ± 0.25	0.318 ± 0.062	0.891 ± 0.09

Table 5.7: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the quality-split test, where the model was not trained on the data of one of the qualities and is trying to recognize it.

Bot's company resistance test.

This test is similar to the previous one, but instead of the quality of bots, we cross out the companies from the training sample. This situation simulates the emergence of bots from a new company when bots qualitatively change their management strategy. In total, we have 3 types of bots, divided by company. Thus, we conduct 3 tests, in each of which one type of bot is excluded from the training sample. We give the final estimate as the average of all estimates with their standard deviation.

Model and split strategy	Accuracy	Precision	Recall	F1	AUC-ROC
RFC test=A	0.911	0.867	0.956	0.909	0.914
RFC test=B	0.781	0.867	0.607	0.714	0.766
RFC test=C	0.778	0.902	0.596	0.718	0.769
RFC ave	0.823 ± 0.076	0.879 ± 0.020	0.720 ± 0.204	0.781 ± 0.111	0.816 ± 0.084
NN test=A	0.869	0.830	0.904	0.441	0.933
NN test=B	0.747	0.851	0.531	0.312	0.802
NN test=C	0.772	0.817	0.669	0.388	0.838
NN ave	0.796 ± 0.064	0.833 ± 0.017	0.701 ± 0.188	0.380 ± 0.065	0.858 ± 0.068

Table 5.8: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the company-split test, where the model was not trained on the data of one of the companies and is trying to recognize it.

5.4.3 Quality estimation tests

In this test, we check the model's ability to determine the quality of bots. Bot quality is a categorical parameter that consistently ranges from low to high.

We define the classifier performance measure as the average shift between the predicted bot class and the real class on the bot quality scale (see Table 2.1). A shift by 0 means that the quality of the bot was recognized correctly. A shift of 3 is the maximum shift when we say that a bot of LOW quality is LIVE quality bot.

We present the distribution of the bias and the mean bias value along with its standard deviation in Table 5.9.

Model	$shift = 0$ percent	$shift = 1$ percent	$shift = 2$ percent	$shift = 3$ percent	$\overline{shift} \pm \sigma$ value
NN	0.648	0.131	0.153	0.067	0.639 ± 0.970

Table 5.9: Classification performance for Neural Network (NN) in the quality estimation test.

5.4.4 Price estimation tests

In this test, we check how accurately we can predict the price of a bot. The proposed technique for bot's price estimation multiplies the price vector by the predicted quality vector of bots. Therefore, the effectiveness of price prediction depends on the quality of the bot market analysis and the effectiveness of predicting the quality of bots. It is also necessary to take into account that the price of bots differs from one bot seller to another, so we cannot achieve an ideal price prediction because we leverage average prices on the market.

Based on the analysis of the bot market (presented in section 2.2.4), we made a table of average prices for the social network VKontakte, which is presented in the Table 5.10.

Action	LOW quality	MED quality	HIGH quality	LIVE quality
friend	0.196	0.333	0.517	0.967
participate	0.236	0.333	0.508	0.789
like	0.069	0.086	0.214	0.324
repost	0.149	0.209	0.328	0.750
view	0.054	0.057	0.100	0.123
comment	0.090	0.845	1.600	7.675
poll	0.054	0.341	0.430	2.570

Table 5.10: Table of average prices for bots depending on the quality of the bot and the action performed (per one bot in Russian rubles).

Since we ordered only the "like" action from the bot sellers, we can check the price for this action only. How much we paid for bots to each bot seller are presented in the table 5.11.

Company	LOW quality	MED quality	HIGH quality	LIVE quality
A	0.112	0.168	0.224	-
B	0.090	0.120	0.320	-
C	-	-	-	0.225

Table 5.11: The price of the bots we paid to the bot sellers (per one bot in Russian rubles).

We define the metric of a bot's price prediction efficiency as a shift in value between the predicted price and the real price. In the Table 5.12 we present average estimates with a standard deviation in percentage terms and Russian rubles.

Shift	LOW quality	MED quality	HIGH quality	LIVE quality	All bots
in percent	0.532 ± 0.419	0.287 ± 0.276	0.305 ± 0.125	0.302 ± 0.151	0.364 ± 0.279
in Rubles	0.053 ± 0.041	0.04 ± 0.033	0.083 ± 0.036	0.068 ± 0.034	0.065 ± 0.039

Table 5.12: Shift in the prediction of the bot's price as a percentage of the real price and rubles.

5.5 Chapter summary

In this chapter, we conduct experiments to evaluate the bot detection efficiency of the proposed approaches.

We presented the VKontakte bot detection prototype that implements the proposed solutions in form of a software and graph database.

We presented the collected data about real users and bots, that we bought directly from bot-sellers by creating fake groups that bots should have attacked. We also presented the dataset generation scheme that we used to generate datasets of groups of bots with different bot-to-user ratios and different sizes of a group.

We performed the feature analysis by analysis of multicollinear features and estimate the features informativity.

We conduct 4 groups of experiments to evaluate the efficiency of each bot detection technique: (1) experiments to detection of a group of bots that include standard test and test with threshold; (2) experiments to individual detection that includes standard test, test with feature selection, bot's quality resistance test and bot's company resistance test; (3) experiment to quality estimation test; (4) experiment to price estimation test.

In the next chapter, we discuss the results of experiments in the context of the goals of the thesis and the stated bot's detection problems. We also discuss legal and ethical issues of bot detection, as well as possible scenarios of usage of the proposed solutions.

Chapter 6

Discussion

In this section, we discuss the results obtained, how effective they are for detecting bots, and how they can be applied in practice. This chapter is divided into the following sections:

1. Analysis of bot detection efficiency – the section is devoted to the analysis of the results of experiments and how good they are.
2. Legal and ethical issues – the section is devoted to the questions of legal and ethical compliance when collecting training datasets from bot-sellers, as well as question of accusing someone of being a bot.
3. Applicability evaluation – the section is devoted to the analysis of possible scenarios for the application of the proposed approaches.

6.1 Thesis results analysis

The experiments carried out made it possible to obtain efficiency estimates for each of the modules of the bot detection system. Based on them, one can determine the strengths and weaknesses of the proposed solutions. Based on them, it is possible to assess in general how effective the full cycle of attack detection is: detecting the target of an attack $\longrightarrow$ detecting bots $\longrightarrow$ predicting quality $\longrightarrow$ predicting price.

We will determine the effectiveness based on the requirements defined in Bot Detection Challenges section:

1. Requirement to low API complexity. We compare the API complexity for the proposed solutions individually and together with existing solutions.
2. Requirement to resistance to camouflage. We analyze the effectiveness of bot detection depending on the camouflage strategy they choose.
3. Requirement to detection of a group of bots. We analyze the possibility and efficiency of detecting a group of bots and the advantages of this approach in comparison with analogs.
4. Requirement to high efficiency of detection. We analyze the assessment of the effectiveness of detecting bots with analogs.
5. Requirement to determining the price and quality of bots. We analyze the possibility and efficiency of estimation of bot's quality and price.

6.1.1 Requirement to low API complexity

The proposed solution consists of four elements, three of which use data from a social network:

1. Detection of group of bots that allows one to detect the target of an attack. Requires receiving from the social network the data of accounts included in the group.
2. Individual bot detection. Requires getting the data of friends of the analyzed account from the social network.
3. Estimation of bot's quality. Requires getting the data of friends of the analyzed account from the social network.
4. Estimation of bot's price. Doesn't require getting data from a social network. The price is determined based on the quality of the bot and the average price table.

Therefore, the problem of API complexity is inherent for the first three components: detection of group of bots, individual bot detection and estimation of bot's quality. We will look at how well the proposed solution works for each of these components individually and together. To do this, we present formulas for calculating the number of API requests, where N is the number of analysed accounts.

For detection of a group of bots, to perform a scan, the system needs to make next number of requests:

$$API\ complexity = N_{accounts} * 7 + 1$$

It is necessary to obtain a list of accounts in a group (equal to +1 in formula) and for each account to obtain list of friends, number of groups, subscriptions, followers, photos, albums, posts (equal to $N_{accounts} * 7$ in formula).

Techniques of individual bot detection and estimation of bot's price use the same data for analysis. Therefore, to perform a scan, the system needs to make next number of requests:

$$API\ complexity = N_{friends} * 7 + 1$$

It is necessary to obtain a list of friends of analysed account (equal to +1 in formula) and for each friend to obtain list of friends, number of groups, subscriptions, followers, photos, albums, posts (equal to $N_{friends} * 7$ in formula).

Estimation of bot's price do not require data from the social network. Therefore, the API complexity for this technique is equal 0.

The API complexity of full cycle of bot detection looks like:

$$API\ complexity = 1 + N_{accounts} * 7 + \sum_{i=1}^{N_{accounts}} * N_{friends,i} * 7$$

The main property that determines the complexity is that for the analysis we use the list of friends and 7 numerical profile indicators as input data.

Let's try to compare this complexity with analogs. We cannot determine the API complexity of similar systems, since it depends on the type of social network, the set of used features, etc. Therefore, we cannot define constants in the API complexity formula, but we can define a variable on which the complexity mainly depends. For our solution, such variable is the number of friends. In other bot detection solutions, the main data source for features is user generated content, so we can define 2 types of systems for comparison:

1. Systems that detect bots based on text – API complexity will depends on the number of posts written by user.
2. Systems that detect bots based on media – API complexity will depends on the number of photos uploaded by user.

Based on the data we collected during experiments, we determined the average number of friends, posts, photos and videos an account had. Table 6.1 shows average values.

source type	friends	posts	photos
mean	354	1158	314

Table 6.1: Average number of friends, posts and photos for one account based on the data collected during the experiment.

It can be seen that much fewer accounts need to be crawled to detect bots using the friends list, compared to text-based analysis, when you need to scan a user's posts. For analysis based on media content, one need to scan less content, but not much less.

Separately, it should be noted the obvious fact that the amount of data used by our solution is significantly less in volume. The proposed system makes a decision based on a list of ids of friends and 7 digits, which is obviously much less than all the text written by the user or uploaded photos. This means that our system requires less data from social networks and generally reduces the load on their infrastructure.

6.1.2 Requirement to resistance to camouflage

In the Bot Detection Challenges section, we've identified the main ways bots try to disguise themselves as real people: (1) use privacy settings, (2) use account of a real person, (3) generate profile (generate profile fields, photos and text content, friendslist graph structure).

We propose to evaluate how well our solution is resistant to each type of camouflage.

Camouflage strategy "use of privacy settings". Since in the proposed approach we do not analyze the user's profile, text or media data, access to his account is not important to us. Only friends' accounts are used for analysis, and even if some of them are closed, the remaining friends with open fields will be enough for analysis. For comparison, methods based on the analysis of a user profile, text or media data will not be able to analyze a private profile at all.

It should be noted that for a profile that is closed by privacy settings, we cannot get a list of friends, which is necessary to analyze them. This limitation can be easily circumvented – it is necessary to collect a complete graph of friends of the entire social network once, and determine the list of friends in the opposite way. It is necessary to find the closed analyzed account in the lists of friends of other users.

Camouflage strategy "use account of a real person". We can indirectly determine the resistance of the proposed techniques to this camouflage by the results of the "Bot's quality resistance test" that is presented in the section . In this test, we estimate how efficiency scores are distributed among bots of different quality. LIVE quality corresponds to bots from exchange platforms, so they use accounts of real people.

We see that the recall for detection of LIVE bots using random forest model has decreased from ≈0.79 to ≈0.60. At the same time, the precision is still high ≈0.91. This means that the ability to detect bots is maintained, while the rate of false positives remains low.

Camouflage strategy "generate profile fields" and generate "photos and text content". The proposed approach does not analyze the user's profile fields, text, or photos, so these camouflage techniques do not affect the proposed approach.

Camouflage strategy "generate friendslist graph structure". We cannot establish which bots from our dataset used this type of camouflage, and which did not.

We assume that these are more likely to be low quality bots, since LOW quality bots are usually generated accounts. MED and HIGH quality can contain a mix of generated accounts, stolen and purchased, therefore, it is less reliable to proceed from their estimates. The detection of LOW quality bots remains high with precision of ≈0.85 and recall of ≈0.93.

Thus, we can conclude that the proposed solution is resistant to various bot's camouflage techniques.

6.1.3 Requirement to detection of a group of bots

Due to the limited amount of resources, it is impractical to scan every social network account. Scanning each account means that the API complexity increases by N times, where N is the number of accounts in the analyzed group. Therefore, a technique for detecting a group of bots was proposed, to detect the presence of bots before individual analysis of each account.

Prescanning is necessary when there is a group of information objects and it is necessary to determine whether an attack was made. Instead of separately analyzing each account that interacted with the object, the method suggests analyzing a group of accounts as a whole. Thus, one can first analyze the groups of accounts and then analyze individual accounts only for those groups that gave a positive result (there are bots in the group). Such strategy allows one to reduce the search area.

Detection of a group of bots tests shows that it is difficult to identify bots if the bot-to-user ratio is too low. This is also influenced by the size of the group itself – the proposed approach is worse at detecting bots in small groups.

Therefore, for groups of accounts with a low bot-to-user ratio, relaxation can be introduced by stating that the method will not detect bots if the bot-to-user ratio is too low. We have demonstrated in the threshold test that this strategy works. If we introduce a limitation in the $ratio > 0.3$, the method demonstrates a significant reduction in the frequency of false positives – precision increases from $\approx$0.79 to $\approx$0.88 for all bot-to-user ratios, and increases to $\approx$0.91 for $ratio > 0.3$ with AUC-ROC$\approx$0.91 and recall$\approx$0.91.

This efficiency is sufficient to successfully apply the prescan strategy and recognise the presence of bots.

6.1.4 Requirement to high efficiency

The experimental results show that using the proposed approach, it is possible to achieve high detection quality. The main quality metric is the area under ROC-curve (AUC-ROC), which is a common metric for binary classification tasks. We also pay special attention to the metrics characterizing false positives - Precision with the false-positive rate (FPR).

The results are comparable to those obtained by other researchers. As follows from the reviews [OMAAK20, KŞ17], most of the classifiers developed by researchers have about 0.9 AUC ROC or higher. Standard test shows that it is possible to achieve very good performance (AUC ROC and precision more than 0.9 for neural network) relying only on data about the friends graph and profile statistical data of friends. Standard test with feature selection shows that the number of features can be significantly reduced, and deleting a number of highly correlated features doesn't affect much the classification quality.

Bot's quality resistance test and bot's company resistance test show that it is extremely important to have bots of different quality and from different companies in the training sample. Not all types of bots in the training sample are interchangeable. We assess these results as the ability of our approach to recognizing if bots mutate (change their control strategy [SCM11]). But at the same time, the average AUC-ROC is still ≈0.9 for quality and ≈0.8 for company tests.

More detailed conclusions can be drawn from indiovidual bot detection quality-split and company-slot tests.

From cross quality-split test we can draw a slightly trivial conclusion that by training on HIGH and LIVE quality bots (table rows without *test=HIGH/LIVE*), it will be possible to recognize low and MED quality bots with greater efficiency (AUC-ROC ≈ 0.9) in comparison with training only on LOW and MED quality bots (AUC-ROC ≈ 0.75). This refers to the fact that it is better to include high quality bots in training datasets.

Cross company-split test shows that the quality of the classification is more influenced by the management strategy implemented by the company. For example, one company (compared to another) may pay more attention to creating a good friend structure for a bot. Therefore, the estimates for split by companies vary greatly.

Thus, the developed approach has high efficiency indicators with a fairly high degree of resistance to bots mutating in terms of quality and company (new management strategy).

6.1.5 Requirement to determining the price and quality of bots

Determining the quality and price of bots is an innovative solution, as we have not seen anything like this. This was achieved thanks to the fact that we have collected a high-quality dataset in which there are quality labels for bots. In addition, since we bought bots, we know their cost.

The test for determining the quality of bots showed that for 65% of bots, the quality was assessed correctly. At the same time, if we consider quality as a linear scale from 1 (low) to 4 (live), the average shift on this scale was 0.64 ± 0.97. This means that in general, for most bots, one can fairly accurately determine the quality using the proposed approach.

The technique of estimating price showed that, on average, the technique was wrong by 35% when pricing with a bigger price shift for bots of low quality (for them, the shift is 50%). Nevertheless, this allows an approximate estimate of the cost of an attack in the range.

Thus, the proposed approaches make it possible to determine the quality of bots and the price of bots, thereby describing the attack.

6.1.6 Summary

An important aspect of the proposed approach is also that it is only based on the analysis of a graph formed by friends and their numerical parameters. Such technique has several advantages:

- A small amount of information that is needed for analysis. Unlike text analysis, statistical data and media content, graph analysis does not require a lot of information to download. The amount of information required can be expressed in API requests, the number of which is depends on the number of user's friends.
- There are no language features. Language features depend on many parameters, which are often difficult to take into account together. The texts can be very different for people who speak different languages, from different countries, with different levels of income, education, age, etc. Graph structures are a more universal source of information and do not depend on the characteristics of various social groups.
- If the account is closed by the privacy settings, the list of its friends can be set indirectly. To do this, one needs to get a graph of friends of the entire social network and find the analyzed account in the friendslists of other users. Getting a graph of friends for the entire social network is a difficult, but a completely solvable task. This means that the graph-based approach allows one to detect bots whose profiles are closed by privacy settings or even blocked.
- It is more difficult for bots to mimic the natural structure of a friend graph (in comparison with filling the profile and content). The user's friend graph consists of many overlapping communities and his friends are highly connected. The bot is forced to add random people (who most likely will not be connected) as friends. Or the bot will try to mimic the structure of friends, which will have anomalous features.

Globally - if think about bot detection task as a part of intrusion detection/prevention system that is countering information attacks on social networks, we believe that the effectiveness of the proposed approach allows it to be used with a number of countermeasures. Social networks can use it to detect bots and apply soft severity countermeasures (captcha) and medium severity countermeasures (speed reduction, verification requirement, etc.). The small amount of data required for analysis also allows the use of an approach not only for social networks but also for administrators of online communities to clear participants from bots.

The aspect of applicability should also be noted. The presented experiment was conducted on the social network VKontakte. Despite the fact that there are graphs of friends in almost all social networks, the effectiveness of the proposed solutions for other platforms should be checked separately. We believe that social networks that are similar to VKontakte will have similar results, because their users form similar structures – for example, Facebook and LinkedIn.

6.2 Legal and ethical issues

In this thesis, bot detection and characterization techniques use supervised machine learning methods to make decisions. This is the best one for bots detection and has many advantages. The decisions made by such a system are based on real data and does not depend on the subjective opinions of the operator. But supervised learning methods require a training sample that will contain bots.

We have proposed the best (in terms of objectivity) approach for the such sample creation. Bots are collected directly from sellers, and does not require the presence of operators that manually mark up data. Using our approach, it turns out to achieve 100% valid training sample – in the collected training sample, all accounts are guaranteed to be bots and one can even mark their characteristics (one can get them from sellers).

The need to rent bots in order to train the proposed approaches raises a number of legal and ethical questions that need to be answered. Let's divide these questions into 3 controversial points.

Is it legal and ethical to create fake groups to collect bots?

To rent bots from bot-sellers and don't arouse suspicion, one needs to create a bot attack target. To do this, we created 3 fake groups in the VKontakte social network. We filled this group with absurd material to make an illusion that these groups are real.

According to the rules of a social network, it is prohibited to create groups that mislead users, distribute censored information (different social networks have different criteria, but usually it is child pornography, drug trafficking, terrorism, etc.). Our groups does not contain such content. Also we specifically tried to make our groups unattractive to real users. One group is devoted to the registration of deer in a region where deer cannot be found. The second group offers wigs for hairless cats. And the third offers a route taxi from a city in Estonia to a distant Siberian village. The only ones who saw our groups were bots.

From the point of view of the law, it is also prohibited to distribute censored information. It varies from country to country, and social networks duplicate censorship descriptions in their user agreements. Thus, we have not violated any laws.

In terms of ethics, our fake groups did not have any impact on real users. Therefore, the ethical side of the issue is not relevant as well.

Thus, we believe that the creation of fake groups that are made in such a way does not violate legal and ethical rules. Those who will use our detection techniques can use the same approach to create a training sample. Of course, it is necessary to try to keep the life span of these groups short, and also to observe that real users do not join it.

Is it legal and ethical to rent bot activity to collect bots?

We have repeatedly mentioned that we believe that a high-quality dataset can only be assembled by purchasing or renting bots directly from sellers. In this thesis, we rented bots from 3 bot selling companies. We asked them to give likes to the posts in our fake groups. We collected data related to these accounts and labeled their quality and price based on the information from bot-sellers. Anyone who will repeat such a collection of bots may wonder how legal and ethical it is. From a legal point of view, we propose to look at how the usage of bots is regulated by laws on the example of 3 countries: Unites States, France and Russia.

In United States there is a "Better Online Tickets Sales Act" (BOTS Act of 2016) [Con21b]. This law prohibits the use of bots or other automated methods to purchase tickets for events. This law theoretically can be applied to social network if tickets are sold in it. The method we used has nothing to do with buying tickets. The act also states that it is legal to create or use software or systems to: (1) investigate, or further the enforcement or defense of, alleged violations; or (2) identify and analyze flaws and vulnerabilities of security measures to advance the state of knowledge in the field of computer system security or to assist in the development of computer security products. Therefore, it is legal to collect information about bots, as we increase the effectiveness of security tools.

Separately, we can mention "Bot Disclosure and Accountability Act of 2019" [Con21a] and it has not been passed at the moment. This act enforces transparency requirements on social media companies regarding the use of social media bots that replicate human activity, and prohibit from using social media bots as a mean to disguise online political advertising or deceive voters. This law aims to protect the electoral cycle. Although not yet adopted, it does not prohibit the collection of bot data.

France has several laws that govern social media [PAC20]. In particular, they regulate dissemination of fake news, and defamatory speech and introduces limits on paid political advertising. Fake information is often spread by bots and these laws require such information to be detected and tagged. At the same time, the laws require countering to accounts that massively disseminate fake information. Therefore, we can say that French law supports our type of research. As in US, France tries to protect the electoral cycle. Therefore, when buying bots, it is necessary that they do not affect real users, especially politically.

In Russia the usage of bots is not regulated in any way, but unlike other countries in Russia there is a department whose task is to monitor the laws compliance in the Internet - Federal Service for Supervision of Communications, Information Technology and Mass Media (Roskomnadzor). Roskomnadzor demanded that the Telegram messenger block bots that collected data from Russians. It also tried to block Telegram for not removing information about terrorism, and also slows down Twitter in Russia for not removing pornographic, pro-narcotic and suicidal content. Thus, Roskomnadzor counteracts malicious bots, but does not prohibit the use of them if Russian laws are not violated.

From all this, we can conclude that the use of bots is not regulated by laws in any way. Instead, the laws define actions that can violate the law. In this thesis, there was no impact on real users due to the actions of bots, and the actions of bots were in no way associated with the spread of fake news, pornography, etc. So the laws were not violated, and our method can be used to collect bots. Also, laws support the development of security tools that detect bots.

From an ethical point of view, we justified that buying bots will not harm anyone. We deliberately made fake groups to make sure that bots would not have an impact on real users. So using such a "sandbox" is the ethical way to get data about bots.

Is it ethical to blame someone of being a bot according to the our tool result?

In the thesis, a lot of attention is paid to the evaluation of the effectiveness of the proposed techniques and false positive results. The main priority for us was to reduce false positives (precision efficiency metric), even at the expense of the ability to detect bots (recall efficiency metric). But there are still some false positives. For example, if the false positive rate is around 10%, this means that if we analyse a random user, our tool with a 10% probability says that this real user is a bot. Therefore, the classification result cannot be used to blame people. On the other hand, if an analysis of 1000 users showed that 500 of them are bots (a certain threshold has been exceeded), then we can conclude that there is a bot attack.

Typically, the intrusion detection system handles such final decisions and decides whether to apply any countermeasures. This intrusion detection system is architecturally above the bot detection system. If there is no such system, and the bot detection tool is used by itself, then a person makes a decision based on the detection results in context - what actions the bot performed, what damage it causes, etc. Even if a user will use the bot detection tool, nothing prevents him/her from using soft countermeasures – to conduct a Turing test or simply add the account to the blacklist.

Thus, the usage of the proposed methods is legal and ethical. Moreover, bot trading is not illegal. It is illegal to use bots for malicious purposes. We are confident that the development of such methods is supported by laws and morals. However, one need to be sure that during the data collection, bots will not have any impact on real users.

6.3 Application proposals

In this section, we provide application guidelines and theoretical considerations on how and to whom the proposed solutions will be useful. In the Scientific Context chapter, we identified the main opposing parties that counteract bots: social networks, governments, commercial and non-profit companies, users. We will consider the possibilities of using the proposed solutions from their point of view.

6.3.1 For social networks

Social networks counteracts bots in order not to lose their reputation in the eyes of civil society and the state. Examples of the implementation of the proposed approach by social networks are:

1. Monitoring. From time to time, a social network can check its users for bots. But since the precision is ≈ 0.9, there will be a lot of false positives during such a check. In order not to harm the real user who were falsely recognized as bots, you can introduce soft countermeasures that are not essential for the user but will significantly complicate the work of the bots. For example, the introduction of a Turing test or the requirement to undergo additional verification via SMS or email. Such countermeasures will significantly complicate the work of automated bots (usually low to medium quality bots).

2. Complaints. Checking for bots can be carried out when users complain about a suspicious account. In case of a positive result, the account can be frozen (deactivated with the requirement to pass verification).

3. Reputation system. A social network can build its recommendation systems in such a way that accounts that have been recognized as bots do not fall into them. For example, if algorithms like page rank are used to calculate the importance of content, you can assign less weight to the actions of bots. Thus, the content that has been popularized by bots will have no weight and this will significantly complicate the propaganda carried out by the bots.

Since social networks do not have API limitations on access to data, for them the API complexity is not so important.

It should be noted that usually social networks have already implemented mechanisms to combat bots in some form. In this case, they can be complemented by our proposed approaches.

Thus, implementing bot detection can improve social media rankings and user trust.

6.3.2 For government

Governments oppose bots to combat information that is harmful from the point of view of the law (for example, for Russia, this is the propaganda of drugs, Nazism, etc.) and lobbying (interference in elections and other sensitive areas). Examples of the implementation of the proposed approach by governments are:

1. Center for Monitoring Internet Space. Law enforcement agencies of some countries have departments responsible for monitoring the Internet space for compliance with legislation (for example, in Russia it is Roskomnadzor). Methods to identify bots can be used to analyze the dissemination of malicious information from the point of view of the law as well as impact assessments. Governments cannot apply countermeasures within social networks – but they can request social networks to delete information or take other countermeasures.
2. Social network forensics. Law enforcement agencies can use the proposed methods in order to characterize attacks on social networks. Using the proposed approaches, it is possible to determine whether the activity have artificial (bots) or natural (not bots) nature, to create a model of an attacker based on the quality of bots, to determine the price of attack, etc. In general, the developed approaches can be used to calculate a number of parameters for risk assessment process.

It should be noted that states, unlike other parties, governments cannot introduce countermeasures. They can ask the social network to introduce countermeasures, or to introduce countermeasures outside of social networks – block social network, bring charges of breaking the law, etc. Also, we can note the high interest of governments in detecting bots during electoral cycles.

Thus, implementing bot detection can improve national security.

6.3.3 For commercial and non-profit companies

Commercial and non-profit companies counteract bots to protect their reputation as part of countering competitors' attacks. This can happen when competitors leave bad reviews about a product in use, spread rumors, etc. Examples of the implementation of the proposed approach by companies are:

1. Clearing public pages from bots. Companies that have public pages in form of the social network community can recognize bots and delete them or add to black lists. If the social network provides flexible community management settings, then other countermeasures can be applied and used - such as ban on commenting, dislike, etc.

Thus, implementing bot detection can improve user trust.

6.3.4 Civil society

Civil society is the goal of the attacking, so individuals can use the proposed solution to protect themselves. Examples of the implementation of the proposed approach by civil society are:

1. Clearing public pages from bots – recognize bots and delete them or add to black lists.
2. Trust in the seller – the user can check the reviews for the product to make sure that they were left by real people.
3. Network activism. Individuals can use the proposed methods to detect attacks on other social network pages in order to give publicity.

Thus, implementing bot detection can improve trust of users to each other, as well as to protect users from opinion manipulation.

6.4 Chapter summary

In this chapter we discuss the results of experiments in a context of the goals of thesis and the stated bot's detection problems. We showed that the proposed solutions are satisfy stated requirements, thus, the goal of the thesis was achieved.

We also discus legal and ethical issues of bot detection. We showed that the proposed solutions for collection training datasets do not carry legal or ethical risks, for those who will implement the proposed solutions.

We also presented the possible scenarios for using the proposed solutions for detecting bots. This scenarios includes proposals for social networks, governments, commercial and non-profit companies and civil society.

In the next chapter, we summarize the results of the thesis.

Chapter 7

Conclusion

The goal of the thesis is to develop a solution for recognizing and characterizing bots attacks on social networks. The presented solution works with a relatively small amount of data regardless of language features, detects bots with hidden profiles, provides a sufficient detection quality, and estimates bot quality and bot price. We have successfully achieved this goal by:

1. developing the technique for detection of a group of bots in a social network;
2. developing the technique for individual bot detection in a social network;
3. developing the technique for social network bot quality estimation;
4. developing the technique for social network bot price estimation.

This set of techniques allows to implement the full cycle attack detection pipeline: detecting the target of an attack using detection of a group of bots $\longrightarrow$ detecting bots individually among positive groups $\longrightarrow$ estimating the bot quality among positive accounts $\longrightarrow$ estimating the price of attack based on the bot quality.

In order to create these techniques and evaluate their efficiency, we also solved many sub-problems:

1. analysed the bot market pricing and provide bot classification by bot quality and strategies for bot creation, management and camouflaging;
2. developed the semantic model of VKontakte social network for extraction of adjacent account from an analysed account;
3. developed the feature construction technique based on the statistics, including novel techniques based on the agreement with Benford's law and Gini index;
4. developed the feature construction technique based on analysis of friend-graph of a social network;
5. developed the novel feature construction technique based on analysis of global graph of a social network;
6. collected the dataset with bots and real users directly from companies that trading bots by creating fake groups, which bots should have attacked, and generate datasets of groups of bots with different bot-to-user ratio and different group size;

7. developed the software prototype for bot detection in VKontakte social network and conduct the experiments;
8. estimated features by analysis of multicolinearity and informativity;
9. conducted 8 types of experiments that can be separated to 4 groups: detection of a group of a bots, individual detection, quality estimation; (4) price estimation;
10. determined the effectiveness of proposed solutions based on the requirements defined;
11. answered the questions of legal and ethical compliance when collecting training datasets from bot-sellers, as well as question of accusing someone of being a bot;
12. considered possible scenarios for the application of the proposed bot detection approaches.

As a result, the proposed approach solves all 5 problems that researchers usually face:

1. Bot camouflaging problem – we have demonstrated that the proposed solution is resistant to the emergence of bots of new quality.
2. The problem of linguistic and behavioral universality – the proposed approach does not depend in any way on the language features.
3. The problem of limiting access to data – we have demonstrated that the proposed solution requires a similar or less number of requests to the social network, and significantly less data in volume to perform analysis.
4. Bad training datasets problem – we have demonstrated that it is possible to assemble a high-quality dataset that is devoid of subjectivity and has additional quality and price labels. Using this dataset we were able to develop techniques for determining the quality and price of a bot.
5. The problem of detecting a group of bots – we have demonstrated a group-scan technology that significantly reduces the bot-search area.

The research results were also partially used in the next projects:

1. Grant of Russian Science Foundation № 18-71-10094 "Monitoring and counteraction to malicious influence in the information space of social networks", 2018-2021
2. Grant of Russian Science Foundation № 18-11-00302 "Intelligent digital network content processing for effective detection and counteraction of inappropriate, dubious and harmful information", 2018-2020

The results of the thesis are used by the Russian state agency "Roskomnadzor" and by the "International Digital Forensics Center" to detect bots on social networks.

In the final conclusion, we believe that the proposed approaches to detecting and characterizing bots will make the fight against them more effective. The efforts of many researchers around the world are thrown into solving this problem, and we are making our contribution to this area. We hope that the use of the proposed techniques will lead to greater transparency, trust and fairness in the society, which we all belong to.

Bibliography

[AAEA18] Sadam Al-Azani and El-Sayed M El-Alfy. Detection of arabic spam tweets using word embedding and machine learning. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–5. IEEE, 2018.

[AAK+17] Kayode Sakariyah Adewole, Nor Badrul Anuar, Amirrudin Kamsin, Kasturi Dewi Varathan, and Syed Abdul Razak. Malicious accounts: Dark of the social networks. *Journal of Network and Computer Applications*, 79:41–67, 2017.

[AC18] SP Mukherjee Bikas K Sinha Asis and Kumar Chattopadhyay. *Statistical methods in social science research*. Springer, 2018.

[aut21] Unknown author. search4faces – service for finding people by photo, 2021.

[B+18] Sebastian Bay et al. The black market for social media manipulation. *NATO StratCom COE*, 2018.

[BLSSA17] James Bednar, Open Source Tech Lead, and Anaconda Senior Solutions Architect. Big data visualization with datashader. *Anaconda, Inc.: Austin, TX, USA*, page 12, 2017.

[CBC+19] Andrea Cerioli, Lucio Barabesi, Andrea Cerasa, Mario Menegatti, and Domenico Perrotta. Newcomb–benford law and the detection of frauds in international trade. *Proceedings of the National Academy of Sciences*, 116(1):106–115, 2019.

[CHM16] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. Debot: Twitter bot detection via warped correlation. In *Icdm*, pages 817–822, 2016.

[Com21] OrientDB Community. Orientdb, 2021.

[Con21a] United States Congress. S.2125 - bot disclosure and accountability act of 2019, 2021.

[Con21b] United States Congress. S.3183 - bots act of 2016, 2021.

[CP19] Andy Coenen and Adam Pearce. Understanding umap. *Google PAIR*, 2019.

[CWZ+16] Chao Chen, Yu Wang, Jun Zhang, Yang Xiang, Wanlei Zhou, and Geyong Min. Statistical features-based real-time detection of drifted twitter spam. *IEEE Transactions on Information Forensics and Security*, 12(4):914–925, 2016.

[DAD18] Ali Dorri, Mahdi Abadi, and Mahila Dadfarnia. Socialbothunter: Botnet detection in twitter-like social networking services using semi-supervised collective classification. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic*

and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pages 496–503. IEEE, 2018.

[DKS14] John P Dickerson, Vadim Kagan, and VS Subrahmanian. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 620–627. IEEE, 2014.

[DL18] Guozhu Dong and Huan Liu. *Feature engineering for machine learning and data analytics*. CRC Press, 2018.

[DVF+16] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Botornot: A system to evaluate social bots. In *Proceedings of the 25th international conference companion on world wide web*, pages 273–274, 2016.

[EDF+16] Sergey Edunov, Carlos Diuk, Ismail Onur Filiz, Smriti Bhagat, and Moira Burke. Three and a half degrees of separation. *Research at Facebook*, 694, 2016.

[ES21] Amy Mitchell Elisa Shearer. News use across social media platforms in 2020, 2021.

[Fer20] Emilio Ferrara. # covid-19 on twitter: Bots, conspiracies, and social media activism. *arXiv preprint arXiv:2004.09531*, 2020.

[FRE+17] Robert Faris, Hal Roberts, Bruce Etling, Nikki Bourassa, Ethan Zuckerman, and Yochai Benkler. Partisanship, propaganda, and disinformation: Online media and the 2016 us presidential election. *Berkman Klein Center Research Publication*, 6, 2017.

[GG12] Radu Cornel Guiasu and Silviu Guiasu. The weighted gini-simpson index: revitalizing an old index of biodiversity. *International Journal of Ecology*, 2012, 2012.

[GHD17] Nicolas Gauvrit, Jean-Charles Houillon, and Jean-Paul Delahaye. Generalized benford's law as a lie detector. *Advances in cognitive psychology*, 13(2):121, 2017.

[Gin12] Corrado Gini. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E*, 1912.

[GKL08] Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th international conference on World Wide Web*, pages 645–654, 2008.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[Gol15] Jennifer Golbeck. Benford's law applies to online social networks. *PloS one*, 10(8), 2015.

[Goo21] Google. Vision ai, 2021.

[GPAT17] Christian Grimme, Mike Preuss, Lena Adam, and Heike Trautmann. Social bots: Human-like by means of human control? *Big data*, 5(4):279–293, 2017.

[HJ20] Maryam Heidari and James H Jones. Using bert to extract topic-independent sentiment features for social media bot detection. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0542–0547. IEEE, 2020.

[HJU20] Maryam Heidari, James H Jones, and Ozlem Uzuner. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 480–487. IEEE, 2020.

[HLZ18] Chih-Chung Hsu, Chia-Yen Lee, and Yi-Xiu Zhuang. Learning to detect fake face images in the wild. In *2018 International Symposium on Computer, Consumer and Control (IS3C)*, pages 388–391. IEEE, 2018.

[JCZ+16] Zhiwei Jin, Juan Cao, Yongdong Zhang, Jianshe Zhou, and Qi Tian. Novel visual and statistical image features for microblogs news verification. *IEEE transactions on multimedia*, 19(3):598–608, 2016.

[KA13] Ahmad Kamal and Muhammad Abulaish. Statistical features identification for sentiment analysis using machine learning techniques. In *2013 International Symposium on Computational and Business Intelligence*, pages 178–181. IEEE, 2013.

[Kak80] Nanak C Kakwani. *Income inequality and poverty*. World Bank New York, 1980.

[KB17] Veton Këpuska and Gamal Bohouta. Comparing speech recognition systems (microsoft api, google api and cmu sphinx). *Int. J. Eng. Res. Appl*, 7(03):20–24, 2017.

[KG20] Ilia Karpov and Ekaterina Glazkova. Detecting automatically managed accounts in online social networks: Graph embedding approach. *arXiv preprint arXiv:2010.07923*, 2020.

[KMY+20] MI Kuptsov, VA Minaev, SL Yablochnikov, VB Dzobelova, and AV Sharopatova. Some statistical features of the information exchange in social networks. In *2020 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, pages 1–4. IEEE, 2020.

[KŞ17] Arzum Karataş and Serap Şahin. A review on social bot detection techniques and research directions. In *Proc. Int. Security and Cryptology Conference Turkey*, pages 156–161, 2017.

[LC17] Levada-Center. Channels of information, 2017.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MCKM17] Amanda Minnich, Nikan Chavoshi, Danai Koutra, and Abdullah Mueen. Botwalk: Efficient adaptive exploration of twitter bot networks. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*, pages 467–474, 2017.

[MHM18] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[Mil15] Steven J Miller. *Benford's Law*. Princeton University Press, 2015.

[MS20] Hadi Mansourifar and Weidong Shi. One-shot gan generated fake face detection. *arXiv preprint arXiv:2003.12244*, 2020.

[New18] Mark Newman. *Networks*. Oxford university press, 2018.

[NMM+19] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019(5):532–1, 2019.

[NWS02] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the national academy of sciences*, 99(suppl 1):2566–2572, 2002.

[OES19] Birgit Oberer, Alptekin Erkollar, and Anna Stein. Social bots–act like a human, think like a bot. In *Digitalisierung und Kommunikation*, pages 311–327. Springer, 2019.

[OMAAK20] Mariam Orabi, Djedjiga Mouheb, Zaher Al Aghbari, and Ibrahim Kamel. Detection of bots in social media: a systematic review. *Information Processing & Management*, 57(4):102250, 2020.

[PAC20] Francesco Pierri, Alessandro Artoni, and Stefano Ceri. Investigating italian disinformation spreading on twitter in the context of 2019 european elections. *PloS one*, 15(1):e0227821, 2020.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[PdPLS19] Marco Polignano, Marco Giuseppe de Pinto, Pasquale Lops, and Giovanni Semeraro. Identification of bot accounts in twitter using 2d cnns on user-generated contents. In *CLEF (Working Notes)*, 2019.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[SAD+16] Venkatramanan S Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. The darpa twitter bot challenge. *Computer*, 49(6):38–46, 2016.

[SCM11] Tao Stein, Erdong Chen, and Karan Mangla. Facebook immune system. In *Proceedings of the 4th workshop on social network systems*, pages 1–8, 2011.

[SCV+17] Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Alessandro Flammini, and Filippo Menczer. The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*, 96:104, 2017.

[STZ18] Kirill Skorniakov, Denis Turdakov, and Andrey Zhabotinsky. Make social networks clean again: Graph embedding and stacking classifiers for bot detection. In *CIKM Workshops*, 2018.

[SZ21] Gadzhi Saidov and Yuri Zdanovich. Find clone – service for finding people by photo, 2021.

[SZC19] Peining Shi, Zhiyong Zhang, and Kim-Kwang Raymond Choo. Detecting malicious social bots based on clickstream sequences. *IEEE Access*, 7:28855–28862, 2019.

[UKBM11] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[Wan21] Philip Wang. thispersondoesnotexist – service for generating photos of non-existent people, 2021.

[WN19] Feng Wei and Uyen Trang Nguyen. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 101–109. IEEE, 2019.

LIST OF PAPERS BASED ON THESIS

1. Maxim Kolomeets, Andrey Chechulin, and Igor Kotenko Bot detection by friends graph in social networks *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, Vol. 12, No. 2, 2021. (SCImago Q2).

2. M. Kolomeets, A. Benachour, D. El Baz, A. Chechulin, M. Strecker, and I. Kotenko Reference architecture for social networks graph analysis *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, Vol. 10, No. 4, 2019, P. 109-125. (SCImago Q2)

3. M. Kolomeets, A. Chechulin, and I. Kotenko Social networks analysis by graph algorithms on the example of the VKontakte social network *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, Vol. 10, No. 2, 2019, P. 55-75. (SCImago Q2).

4. M. Kolomeets, D. Levshun, S. Soloviev, A. Chechulin, and I. Kotenko Social networks bot detection using Benford's law *13th International Conference on Security of Information and Networks (SIN 2020)*, 4-7 November 2020, Istanbul, Turkey. Proceedings by ACM International Conference Proceeding Series (ICPS). 2020. ACM, New York, NY, USA, 2020, P. 1-8.

5. M. Kolomeets, O. Tushkanova, D. Levshun, and A. Chechulin Camouflaged bot detection using the friend list *29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Valladolid, Spain, March 10-12, 2021, P. 253-259.

6. M. Kolomeets and A. Chechulin Analysis of the Malicious Bots Market *Conference of Open Innovation Association, FRUCT. 29th Conference of Open Innovations Association FRUCT*, Tampere, Finland, May 12-14, 2021, P. 199-205.

7. L.Vitkova, I. Kotenko, M. Kolomeets, O. Tushkanova, and A. Chechulin Hybrid Approach for Bots Detection in Social Networks Based on Topological, Textual and Statistical Features *4nd International Scientific Conference "Intelligent information technologies for industry" (IITI'19)*, December 2-7, 2019, Ostrava-Prague, Czech Republic // Advances in Intelligent Systems and Computing, Springer, vol. 1156, 2020, P.412-421.

8. L. Vitkova and M. Kolomeets Approach to Identification and Analysis of Information Sources in Social Networks *13th International Symposium on Intelligent Distributed Computing (IDC 2019)*, Saint-Petersburg, Russia, 7-9 October 2019. Studies in Computational Intelligence, Vol. 868, 2020, P. 285-293, ISSN 1860-949X.

9. M. Kolomeyets, A. Chechulin, and I. Kotenko The technique of structuring social network graphs for visual analysis of user groups to counter inappropriate, dubious and harmful information *2nd International Scientific-Practical Conference Fuzzy Technologies in the Industry (FTI 2018)*, October 23-25, Ulyanovsk, Russia, 2018, P.87-95.

10. M. Kolomeets and K. Zhernova Visual analysis of social network bots in augmented reality *XVII St. Petersburg International Conference "Regional Informatics (RI-2020)"*, St. Petersburg, October 28-30, part 1., 2020, P. 141-142 (in Russian).

11. M. Kolomeets , I. Kotenko, and A. Chechulin Using virtual and augmented reality for visualization of cybersecurity data *Information Security. Inside*, No. 5 (77), 2017, P.58-63 (VAK, in Russian).

12. M. Kolomeets, A. Chechulin, E. Doinikova, and I. Kotenko Cybersecurity metrics visualization technique *Journal of Instrument Engineering*, T.61, No. 10, 2018, P.873-880 (VAK, in Russian).

13. K. Zhernova, M. Kolomeets, I. Kotenko, and A. Chechulin Application of an adaptive touch interface in information security applications *Cybersecurity issues*, No. 1 (35), 2020 P. 18-28 (VAK, in Russian).