

# A Method of Detecting Bot Networks Based on Graph Clustering in the Recommendation System of Social Network

Yelyzaveta Meleshko<sup>a,b</sup>, Mykola Yakymenko<sup>a</sup> and Serhii Semenov<sup>b</sup>

<sup>a</sup> Central Ukrainian National Technical University, 8, Universytetskyi prosp., Kropyvnytskyi, 25006, Ukraine

<sup>b</sup> National Technical University "Kharkiv Polytechnic Institute", 2, Kyrpychova str., Kharkiv, 61002, Ukraine

## Abstract

This paper proposes a method for detecting a network of bots in the recommendation system based on graph clustering and analysis of user actions. This method is proposed to be used if there are signs of an attack on the recommendation system and a set of probable targets of the attack is revealed. A series of experiments was carried out to test the effectiveness of the proposed method. The input data for the experiments were generated in the developed software simulation model of users and items of the recommendation system that made it possible to simulate the influence of external destabilizing factors on the system such as information attacks. A subsystem of information security of the recommendation system has been developed. Such subsystem consists of a method of detecting an information attack on the recommendation system based on trend analysis in item ratings and a method of detecting bot networks in the recommendation system based on graph clustering and user action analysis. Also, appropriate algorithms have been developed.

## Keywords 1

recommendation systems, data analysis, clustering, information attacks, bot networks, graphs, algorithms, computer simulation, social networks, websites

## 1. Introduction

Most often information attacks on recommendation systems are carried out for marketing purposes in order to increase the ratings of the attacking party's products or decrease the ratings of competitors' products [1-6]. Also attacks on recommendation systems may also be aimed at propagating informational influences during informational conflicts, e.g., for political purposes. Various information influences are often carried out through social networks [7-12]. Recommendation systems as their components have become one of the targets for information attacks in order to implement such influences [1-6, 13-15]. By successfully attacking a social network recommendation system or news aggregator one can change the content and order in which items of news feed are displayed to system users. It can be used not only for marketing but also for political or fraudulent purposes. To implement attacks on recommendation systems bot networks are used, since only a certain set of profiles in the system can influence recommendations formation by their joint actions [1-6, 14, 15].

### 1.1. Related Works and Problem Statement

Mostly in existing studies [4, 14-17] it is proposed to consider the detection of an information attack on the recommendation system as identical to the detection of bot profiles, and for detecting a bot network, all user profiles are clustered based on their statistics by non-hierarchical methods (for example, k-means clustering). After such clustering bots and authentic users get into different

clusters.

Since the detection of bot profiles is quite a resource-intensive task, this paper proposes to split the task of protecting the recommendation system from information attacks into two parts: 1) detection of attack signs and 2) detection and neutralization of bot profiles.

Attack detection can be a less resource-intensive task if it will be realized as tracking the dynamics of ratings of system items (all ratings or only critical ones from the point of view of information security). For example, if the ratings of items to protect start to change rapidly, and new ratings that lead to changes in ratings do not match previous average ratings of these items, you should check for bots among users who have started making such ratings. This approach will reduce the number of user profile checks. First, since they will need to be checked only if an attack is suspected. Secondly, because it will be necessary to check the profiles of not all users, but only those who carry out suspicious actions.

If in the system items with an abnormal change in rating trends are detected (that could hypothetically be the result of a botnet attack), the next logical step is to try to identify user profiles that influenced this change and try to find out if they are connected to a coordinated network or networks.

It is natural that changes in trends in the ratings of items are influenced by all users who gave them ratings equal to the target – high if the item's ratings increased or low if the item's ratings decreased. The set of such users is easily identified by simple queries to the recommendation system database. But it is much more difficult to determine which user profiles are really part of the botnet and perform coordinated actions with other members of the botnet and who simply gave a rating that corresponded to their preferences.

Thus, *the purpose of this work* is to develop a method of detecting bot networks in the recommendation system of social network, which could be used only after detecting signs of an attack to check only those user profiles who interacted with possible attack objects.

The problem of botnet revealing among the profiles of system users can be reduced to the problem of finding a subgraph in the social graph of the recommendation system. The vertices in such a subgraph will be user profiles linked by some joint actions that influenced changes in rating trends of all or most items probable targets of the information attack [4, 18]. Therefore, to develop a method for detecting a bot network we decided to use methods of graph clustering.

## **2. Development of a method for detecting bot networks in the recommendation system**

### **2.1. Detection of signs of information attacks on the recommendation system**

Since it is proposed in this paper to split the problem of protecting the social network recommendation system from information attacks into two parts: detecting signs of attack, searching for bot profiles, we will first consider how to solve the first problem before developing a method for solving the second problem.

In [19] a method to determine the set of probable goals of the information attack of the bot network by analyzing the trends of ratings of items of the recommendation system was proposed. Let's consider the basic principles of this method.

The set of possible targets of the attack will be denoted as  $G$ .

The presence of an information attack on the recommendation system causes a change in the ratings of a group of items (increases or decreases them), but the change in the ratings of items is not a sufficient reason to think that there is an attack, since ratings may change due to the actions of authentic users. Therefore, a number of additional features which may indicate an information attack have been selected. A set of indicators to determine the presence or absence of an information attack on the item of the recommendation system was proposed as follows:

$$Q_{a,i} = \{tr, pr, d_r, d_t, n_r, n_{tr}, n_{rec}\}, \quad (1)$$

where  $tr$  – is a trend of rating dynamics of item  $i$  that can take the following values  $\{-1, 0, 1\}$  – respectively "rating downward trend", "no change" and "upward trend";  $pr$  – is a prediction for the trend of the rating dynamics of item  $i$ , for example, based on the Hurst exponent [20];  $d_r$  – is a variance of ratings of item  $i$ ;  $d_t$  – is a variance of time of grading for item  $i$ ;  $n_r$  – is a number of ratings of item  $i$  at

the studied time interval;  $n_{tr}$  – is a number of target ratings for item  $i$  at the studied time interval;  $n_{rec}$  – is a number of gettings of item  $i$  into user recommendation lists at the studied time interval.

We will assume that the attack on the recommendation system occurs when the ratings of one or more items of the system are purposefully changed by the joint actions of a group of system user profiles. Herewith the size of damage from the attack does not always depend on the number of items. Successful change of the ratings of even one item can have great consequences if the item is important and it is, for example, in the social, political or medical sector etc.

It is proposed to determine the presence and type of attack according to the following rules:

**Rule 1.** If the item has any 5 signs from the data: the trend of increasing the rating  $tr_t = 1$ ,  $H > 0.73$ ,  $d_t \leq d_{t,avr}$ ,  $d_r \leq d_{r,avr}$ ,  $n_r > n_{r,avr}$ ,  $n_{tg} > n_{tg,avr}$ ,  $n_{rec} > n_{rec,avr}$ , then we believe that there is a high probability of a rating upgrade attack for this item.

**Rule 2.** If the item has any 5 signs from the data: the trend of decreasing the rating  $tr_t = -1$ ,  $H > 0.73$ ,  $d_t \leq d_{t,avr}$ ,  $d_r \leq d_{r,avr}$ ,  $n_r > n_{r,avr}$ ,  $n_{tg} > n_{tg,avr}$ ,  $n_{rec} < n_{rec,avr}$ , then we believe that there is a high probability of a rating downgrade attack for this item.

Once the attack on the recommendation system is detected and the set of probable targets of bots  $G$  is created, it will be logical to investigate the profiles of all users who influenced the change of item rating trends from this set  $G$ . And in this work, it is precisely such actions that are proposed to be done for the search of bot profiles.

## 2.2. Identification of bot profiles in the recommendation system

To solve the problem of identification of bot profiles in this paper a method based on graph clustering and analysis of user actions was developed using the coefficients of "distrust". The method consists of the following stages:

**Stage 1.** Create a set of suspicious user profiles  $S$ , in which we place the profiles that gave the target estimates  $r_t$  for items from the set  $G$ .

**Stage 2.** Assign to each user from the set  $G$  label *:suspicious* and coefficient of distrust calculated by the next formula:

$$k_{d,i} = \sum_{j \in G} \frac{E_{r_{t,i,j}}}{n_g}, \quad (2)$$

where  $E_{r_{t,i,j}}$  – is a presence of target rating  $r_t$  from user  $i$  for item  $j$  belonging to the set  $G$  of probable targets of the attack, takes values 1, if target rating exists and 0 – in the absence of such a rating from user  $i$  for item  $j$ ;  $n_g$  – is a number of items in the set  $G$  of probable targets of the attack.

**Stage 3.** For each pair of users  $i_1$  and  $i_2$  from the set  $S$ , where  $k_{d,i_1} \geq q$  and  $k_{d,i_2} \geq q$ , create an edge between them with a label *:BotNet*.

**Stage 4.** Perform graph clustering for a subgraph containing vertices with labels *:User* and *:suspicious* and edges with label *:BotNet*. The results of such clustering will be as follows – all bots will get into one large cluster, if there is one botnet, or to several large clusters – if there are several botnets; authentic users will get into different clusters, each of these clusters will contain one user or a small number of users. It is also possible that a certain number of authentic users will get into the cluster with bots, or a group of similar and active authentic users will create a separate cluster if their actions shift the ratings of some items.

**Stage 5.** Define the largest clusters consisting of  $(N_{cr} - e)$  users, where  $N_{cr}$  – is a minimum number of users that may affect the performance of the recommendation system (depends on the parameters of a particular system),  $e$  – is an approximate value of the error when dividing user profiles into clusters. We consider such cluster (or clusters) as a possible bot network (bot networks). For users who do not get into these clusters, remove the edges with the label *:BotNet*. Users who got to the subgraph *BotNet* needed to be further checked analyzing the statistical characteristics of their profiles, for example, using the method proposed in the previous section using neural networks [21]. Also for additional check of profiles from the subgraph *BotNet* one can search for certain features characteristic of bots, for example, poorly completed personal data or extremely high activity (the characteristics of bots depend on the specific system and can become known in the process of

collecting statistics during its operation). One of the common features of bots for many systems can be: the difference between the values of the variance of ratings and the variance of time intervals between ratings in the profiles of bots from the average values of the corresponding variances in the profiles of system users. For a particular system, the features of bots can be: features of profile registration, features of filling the profile with personal information, style of writing and content of comments, user friends list etc. After checking the statistics of individual user profiles that got into the subgraph *BotNet*, it have to be corrected, removing from it users identified by statistics as authentic. If there is a cluster in which all users are recognized as authentic – it should cease to be considered a bot network.

**Stage 6.** Correct the set  $G$  after analyzing the user ratings from the subgraph *BotNet*. It is necessary to check which items users from the botnet have coordinately targeted. Remove from  $G$  items that did not receive at all or received a small percentage of target ratings from users identified as bots. Add to the set  $G$  items that have received target scores from all bots (or a large percentage of bots).

The research to the most well-known and frequently used methods of graph clustering was conducted [22-27]. Consider the methods of graph clustering that can be used in the proposed method of detecting a network of bots and the basic principles of their operation.

**Louvain (or Multilevel) method** – graph clustering method based on multilevel optimization of the modularity function [23, 28]. Modularity is a numerical characteristic that describes the strength of division of a graph into modules, is one measure of the structure of networks or graphs [23, 25, 26]. To assess the modularity the next formula can be used:

$$Q = \frac{1}{2n_e} \sum_{ij} \left( A_{ij} - \frac{d_i d_j}{2n_e} \right) \delta(C_i, C_j), \quad (3)$$

where  $n_e$  – is a number of edges in the graph;  $A$  – is an adjacency matrix of the graph;  $d_i$  – is a number of edges adjacent to the vertex  $i$ ;  $d_j$  – is a number of edges adjacent to the vertex  $j$ ;  $\delta(C_i, C_j)$  – is a delta function equal to one, if  $C_i = C_j$  and zero otherwise.

This value is equal to the difference between the number of edges within the cluster during the current splitting and the number of edges if they were randomly generated [26].

The value of modularity shows the severity of clusters, it will be:

- equal to one for a complete graph in which all vertices are placed in one cluster;
- equal to zero for division into clusters, in which each vertex is in a separate cluster;
- for unsuccessful partitions modularity can take a negative value.

In essence, the value of modularity can be used to assess the quality of splitting graphs into clusters. Qualitative partitioning is characterized by the fact that the number of internal links within each cluster must be greater than the number of its external links.

The value of modularity characterizes not how much for this partition intra-cluster bonds are denser than inter-cluster, but how much they are denser compared to some initial density. Therefore, there is a comparison with the "null hypothesis", which is that the arcs are distributed randomly, that is, there are no patterns in the distribution of the density of arcs within the graph.

The principle of graph clustering algorithms based on modularity optimization is that at each step of the algorithm each cluster vertex is somehow matched to a cluster, the modularity value is calculated and the graph vertices are redistributed between clusters so as to increase the modularity value. The operation of such algorithms stops when it is no longer possible to improve the value of modularity.

The Louvain method consists of two parts. The first part is a greedy optimization of modularity. The second part of the method is as follows: a new graph is created with metapeaks in the form of found clusters and edges with the total weight of all edges going from one cluster to another (loops with total weights of connections inside the cluster are also created). Such a graph is called a metagraph. The algorithm is restarted on a new graph. This is one of the most well-known methods due to its speed.

**LabelPropagation method** – graph clustering method based on graph labeling. It divides the graph into clusters as follows: each vertex in the graph belongs to the cluster to which most of its neighbors belong, if there are several such clusters, then one of them is chosen by chance [23, 26, 28]. This method is based on the idea that the vertex belongs to the cluster to which the largest number of its neighboring vertices belongs. Consider the principle of operation of this method. At the initial point in time, all vertices are assigned a separate cluster. Each vertex receives a label or color of the

corresponding cluster. Then, for each vertex, it is checked which clusters its neighbors belong to and the cluster membership is redistributed. Due to coincidences, it is important to change the order of vertex traversal at each iteration. The algorithm ends when there is nothing to change – all vertices belong to the same clusters as most of their neighbors. To improve the results, one can use the following trick – run the algorithm several times, save the result of its work each time and choose the best option for splitting the graph. The main advantage of this algorithm is almost linear complexity. The disadvantage of the algorithm is that noisy graphs often combine all vertices into one cluster or a small number of clusters.

## 2.3. Experiments to test the effectiveness of the proposed method on a software model

A series of experiments was carried out to test the developed method of detecting a network of bots. Since there is no information on bot profiles in the available open datasets for testing recommendation systems, it is necessary to create a software simulation model of users and recommendation system items to generate datasets that will also contain bot profiles.

### 2.3.1. Software simulation model of users and items of the recommendation system

A software simulation model of users and items of the recommendation system with the ability to simulate information attacks has been developed. Let's consider the basic principles of work of the developed software simulation model of users and items of the recommendation system.

The main type of attacks on recommendation systems are profile injection attacks, which involve the creation of a number of bot profiles (bot networks) that, by coordinated actions, change the ratings and frequency of getting target items into recommendation lists.

A study of existing models of attacks on recommendation systems was carried out in the paper.

The bot profile can contain the following types of ratings:

- Ratings for items from the set  $I_f$  to simulate the actions of real users. The attacker doesn't try to change these ratings, but instead tries to find values for them that are as close as possible to the real target group of users he seeks to influence.
- Ratings for items from the set  $I_{ti}$ , these are the maximum (or close to them) ones in the system for target items for which the attacker seeks to increase the rating.
- Ratings for items from the set  $I_{td}$ , are the minimum (or close to them) ones in the system for target items for which the attacker seeks to decrease the rating.

The number of target items in the bot can vary from 1 to  $K$  and be contained only in the set  $I_{ti}$  or only in the set  $I_{td}$ , or both, and the number of items to fill the profile – from 0 to  $N$ .

#### Random Attack

In bot profiles the set  $I_f$  will be filled with ratings for randomly selected items. Ratings for selected items will also be selected randomly, but in such way that they are close to the global average score in the system, for example, a normal distribution with a mathematical expectation equal to the global average will be used. The target item will be given the maximal  $r_{max}$  or minimal rating  $r_{min}$ , depending on the targets of the attack. The knowledge and effort required to carry out such an attack is quite minimal – the global average score in many systems can be easily learned directly or through indirect data. This attack is not very effective.

#### Average Attack

Uses the individual averages of each item's ratings to create the set  $I_f$ . More information needs to be gathered for this attack. However, an average attack can be successful even with a small set of elements in  $I_f$ , that reduces the amount of information needed to collect. But the cost of such a reduction in the required data will be a large number of profiles with almost the same set of rated elements that will, of course, be easy to detect. This attack is more effective than random. But it is practically ineffective for collaborative filtering algorithms based on the neighborhood model of item-based type.

The average attack requires a relatively large amount of knowledge about the statistics of the actions of real users in the system. Reasonable defense of the recommendation system from such attacks will make it difficult for the attacker to collect the necessary data. To bypass such defense, other attacks are used, for which the requirements for the amount of knowledge are much lower.

Consider existing attacks that require less knowledge than the average attack, but work more effectively than a random attack.

### **Bandwagon Attack**

The purpose of this attack is to associate the attacked item with a small number of items that are often rated by users (let's call them widely known). An attacker creates profiles of bots that contain a set  $I_f$  of ratings of widely known items. Such profiles are more likely to look like a large number of users, as well-known items are those that are rated by many users. Data for such an attack is quite easy to obtain. So, among the well-known items, several are chosen at random. These items are given maximum ratings along with the target item. Some of the items in the set  $I_f$  can be randomly rated, for example like in a random attack, to vary bot profiles. This is a fairly effective attack but like the average attack becomes ineffective when used against collaborative filtering based on the item-based neighborhood model.

### **Segment Attack**

The basic idea of this attack is to change the rating of the item in the target group of users with known or easily predictable preferences. The target item will be rated only in a certain segment of users, so that it gets recommendations only to them. Otherwise, if the target item gets in recommendations for users from other segments, it may start receiving low ratings from them and the number of such low ratings will be greater than artificially inflated ones. To carry out such an attack, an attacker need to find real users who belong to the target segment and collect data on the ratings that they usually give to system items. As with the bandwagon attack, it is usually determined which items in the target segment are widely known. These items are assigned the maximal rating together with the target item. To ensure maximum effect from the attack, some items for the set  $I_f$  are randomly selected and receive minimal ratings, which allows the attacker to make the bot profiles various. This attack is effective against collaborative filtering algorithms based on the item-based neighborhood model.

It should be noted that all of the attack models discussed above can be used to decrease item ratings, but there are specialized attacks that work better than others exactly to decrease ratings.

Consider attack models designed to decrease ratings of recommendation system items.

### **Love/Hate Attack**

This attack is very simple – no knowledge to be required. The target item is given a minimal rating  $r_{min}$ , and the items for the set  $I_f$  are chosen randomly and receive maximal ratings  $r_{max}$ . Despite its extreme simplicity, this is one of the most effective downgrade attacks against collaborative filtering algorithms based on the user-based neighborhood model.

### **Reverse Bandwagon Attack**

This is variation of the above-described bandwagon attack that selects widely known items for  $I_f$  that are rated low by the vast majority of users. These items are rated low in bot profiles, and low ratings are assigned to the target item. Thus the target item begins to be associated with items that are not liked by a large number of users, and this increases the probability that low ratings will be predicted for the item and it will not be included in the lists of recommendations. Although this attack is not as effective as the average attack with a lot of knowledge for user-based systems, it is a very effective attack to downgrade against item-based systems.

Low-level attacks use widely known items to fill a bot's profile with ratings for them. In this way, an attacker can create a profile similar to the average user by examining the ratings of only widely known items.

If an attacker knows exactly which algorithm the referral system uses, he can gather more information to attack.

Thus, attacks can be classified into:

- *Attacks with little knowledge* – this type of attack does not require detailed knowledge of the distribution of scores in the system. It requires system-independent knowledge that can be easily obtained through public sources of information.
- *Attacks with a lot of knowledge* – an attacker needs to have as much knowledge as possible

about the system's algorithms and the distribution of ratings of system items. For example, some attacks require the attacker to know the mean and standard deviation for each item in the system. An example of an attack with a lot of knowledge is a popular attack.

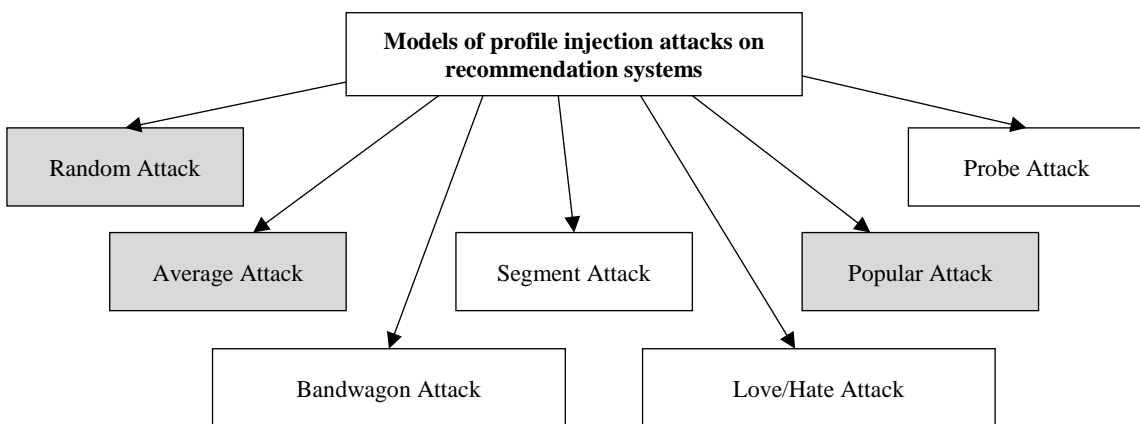
#### Popular Attack

Assume that the system uses a standard user-based collaborative filtering algorithm, where similarity between users is determined by Pearson correlation. As in the bandwagon attack, the set  $I_f$  is filled using well-known system items. However, this does not guarantee a high similarity between the bot profile and the real profiles. Therefore, the popular attack uses the average values of the ratings of selected well-known items to fill the set  $I_f$ . And in order to vary the profile data assign to some randomly selected items in  $I_f$  ratings equal to  $(r_{min} + 1)$  or  $r_{min}$ , depending on whether the average rating for the item is higher or lower. This strategy will lead to positive correlations between bot profiles and authentic profiles. It doesn't take a lot of knowledge to identify well-known items, but a lot of information needs to be gathered to determine the average ratings of selected items. A popular attack can also be easily tuned for downgrade attacks. This attack can be detected by comparing profiles in the system – the bot profiles of one bot network will be very similar.

#### Probe Attack

The more similar bot ratings are to those of real users, the more difficult it is to recognize bot profiles. Knowledge of the real preferences of different segments of users can be obtained from the system itself through a probe attack. To carry out this attack the attacker creates a seed profile and then uses it to obtain recommendations from the system. These recommendations are created on the basis of system information about real users, so the use of the received recommendations will allow the attacker to create bot profiles more similar to real users. The attacker can probe a small number of users to then affect a small group, as in a segmental attack, or a large part – to obtain information, for example, for an average attack. An attacker needs to use only a small number of seed profiles in order for the recommendation system to provide him with the necessary information in the form of recommendations.

Our studies of models of information attacks on recommendation systems have shown that the easiest to implement and the least resource-intensive attacks are random and average ones, and the most effective and inconspicuous (although quite resource-intensive) attack is a popular attack. Other studied models of attacks in fact are their modifications, for example, designed to attack only a certain segment (segment attack), or to attack only to lower ratings (love/hate attack), or to reduce resource consumption by reducing the inconspicuous of the botnet (bandwagon attack, love/hate attack). Therefore, in the software simulation model of users and items of the recommendation network, we decided to model botnets based on three main models of attacks – random, average and popular attack.



**Figure 1:** Types of models of profile injection attacks on recommendation systems

The proposed method of software simulation of users and items of the recommendation system consists of the following parts:

- Generation of the structure of the social graph of the recommendation system [29].
- Simulation of the behavior of users and bots of the recommendation system.

The generation of the structure of the social graph of the recommendation system was carried out on the basis of theory complex networks [22, 30-32] and the Barabashi-Albert (BA) model [22, 30], which creates scale-free networks based on 2 conditions:

**Growth.** Starting with a small number of nodes  $n_0$ , on each new time iteration one new node is added with  $n$  links (where  $n \leq n_0$ ) connecting a new node to  $n$  different existing nodes.

**Preferential attachment.** Probability  $P$  that the new node connect to some existing node  $i$  is the higher the more connections the  $i$ -th node has:

$$P_i = \frac{k_i}{\sum_j k_j}, \quad (4)$$

where  $k_i$  – is a degree of  $i$ -th node, and the denominator calculates the sum of all degrees of existing nodes in the network.

To generate the recommendation system the following subgraphs are used: 1) Users-Friends, Users-Followers, Posts-Published, Posts-Viewed and Posts-Liked – are created by the generator of the social network graph, 2) Users-Similarity, Posts-Similarity and Posts-Recommended – are created by the recommendation system.

**Stages of the developed method:**

**Stage 1.** A non-oriented subgraph Users-Friends is generated based on BA.

**Stage 2.** A non-oriented subgraph Users-Followers is generated based on BA.

**Stage 3.** Subgraphs Users-Friends and Users-Followers are combined into one graph.

**Stage 4.** An oriented subgraph Posts-Published is generated based on modified BA. In the first iteration  $n_0$  users are randomly selected to "create"  $m_0$  posts. Probability post publication by some user:

$$P_i = \frac{k_{1i} + k_{2i} + k_{3i}}{\sum_j (k_{1j} + k_{2j} + k_{3j})}, \quad (5)$$

where  $k_{1i}$  – is a number of friends of  $i$ -th node,  $k_{2i}$  – is a number of subscribers of the  $i$ -th node,  $k_{3i}$  – is a number of posts of the  $i$ -th node, and the denominator calculates the sum of all these values for all existing nodes in the network.

**Stage 5.** Subgraph Posts-Published is joined to the general graph.

**Stage 6.** An oriented subgraph Posts-Viewed is generated based on BA. Probability the post to be reviewed:

$$P_i = \frac{q_{1i} + q_{2i} + q_{3i}}{\sum_j (q_{1j} + q_{2j} + q_{3j})}, \quad (6)$$

where  $q_{1i}$  – is a number of friends of the author of the  $i$ -th post,  $q_{2i}$  – is a number of subscribers to the author of the  $i$ -th post,  $q_{3i}$  – is a number of views of the  $i$ -th post, and the denominator calculates the sum of all these values for all existing nodes in the network.

**Stage 7.** Subgraph Posts-Viewed is joined to the general graph.

**Stage 8.** Subgraph Posts-Liked is generated. Probability of getting a like:

$$P_i = \frac{q_{1i} + q_{2i} + q_{3i} + q_{4i}}{\sum_j (q_{1j} + q_{2j} + q_{3j} + q_{4j})}, \quad (7)$$

where  $q_{1i}$  – is a number of friends of the author of the  $i$ -th post,  $q_{2i}$  – is a number of subscribers to the author of the  $i$ -th post,  $q_{3i}$  – is a number of views of the  $i$ -th post,  $q_{4i}$  – is a number of likes of the  $i$ -th post, and the denominator calculates the sum of all these values for all existing nodes in the network.

**Stage 9.** Subgraph Posts-Liked is joined to the general graph.

**Stage 10.** Subgraphs Users-Similarity, Posts-Similarity and Posts-Recommended are generated by the algorithms of the selected recommendation system.

Studies of the parameters of the generated social graphs [29] (in particular such as density, diameter of a network, clustering coefficient et al.) showed their correspondence to the properties of real social networks.

Modeling of user behavior was based on the generation of a dynamic graph Users-Ratings-Items.

**Stages of the developed method:**

**Stage 1.** Initialization of user parameters and system items.

**Stage 2.** Creating a "Seed" of social graph.



**Stage 3.** Simulation of time iterations. At each time iteration, a number of users, items, and bots join the model. Users and bots rate items.

**Stage 4.** Stopping the model and saving the resulting data set to a file.

Generating ratings for regular users:

$$d_{u,m} = \frac{\sum_{i=0}^n |f_{u,i} - f_{m,i}|}{n}, \quad (8)$$

$$r_{u,m} = \Psi(5d_{u,m} + b_u + b_m), \quad (9)$$

where  $d_{u,m}$  – is a distance between user  $u$  and item  $m$  in the multidimensional space of hidden factors;  $n$  – is a number of hidden factors;  $f_{u,i}$  – is a  $i$ -th hidden factor of user  $u$ ;  $f_{m,i}$  – is a  $i$ -th hidden factor of item  $m$ ;  $b_u$  – is a user offset in items' evaluation (level of requirements to content);  $b_m$  – is a shift of the item in obtaining ratings (content quality level);  $\Psi()$  – is an function that converts the resulting fractional number into a discrete number from a set of ratings [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0].

Among the existing attack models for software simulation we chose random, average and popular attacks – since these models are the most commonly used, universal and illustrate different attack strategies.

Generating a ratings for bots:

$$r_{u,m} = \begin{cases} \text{attackPattern()}, & \text{if item – "ordinary"} \\ 5.0, & \text{if item – "target"} \end{cases}, \quad (10)$$

where  $\text{attackPattern}()$  – is a function generating ratings for non-target items according to the type of attack model selected.

### 2.3.2. The results of experiments to verify the effectiveness of the proposed method of detecting bot networks

Data sets for experiments were generated in the developed software simulation model of the recommendation system. The format and statistical features of the data were generated as close as possible to the corresponding characteristics of MovieLens Datasets [33]. Information attacks were modeled using random, average, and popular attack models [2-5]. The Label Propagation algorithm was used as a graph clustering algorithm.

In the series of experiments, 5%, 10%, 20% and 30% of bots were generated in different series, all other users of the system are authentic. In different experiments bots had different numbers of targets to attack: 1, 5, 10, 15, 20, 25, 30, 35, 40 and 45 targets. Also 20% of profiles with a high level of activity in the recommendation system were generated among authentic users. Such users gave significantly more ratings than the average user. System items were generated to belong to one of the 19 clusters.

The bot search algorithm for the studied recommendation system had the following parameters:

- $q = 0.05$  – that is, the user has set target ratings of at least 5% of the items that are identified by the information security subsystem as probable targets of the attack. The threshold is small because it is not known how accurately the targets of the attack are recognized, there may be many items with the ratings that have changed naturally.

- The elimination of user profiles from the *BotNet* subgraph is based on the values of the variance of ratings and the variance of time intervals between ratings. Authentic users who mistakenly get into the *BotNet* subgraph are users whose profile data satisfy the following rule:

$$|D_r - D_{r,avr}| < 0.15 \text{ AND } (D_{t,r} > 72 \text{ AND } D_{t,r} < 600),$$

where  $D_r$  – is a variance of ratings in the user profile,  $D_{r,avr}$  – is a truncated mean variance of ratings in the system user profile (30% of the extreme values were truncated, i.e., value equal to the maximum possible percentage of bots in the system in question),  $D_{t,r}$  – is a variance of time intervals between rates. The variance of ratings in the profiles of authentic users is not significantly different from the average variance (the attacker of system can not know this value, he can only estimate it, so can not accurately reproduce this characteristic when creating bot profiles). And checking the variance of time intervals between ratings is as follows – users who have too identical or too different time intervals between ratings should be considered suspicious, while the variance of time intervals between ratings of authentic users is usually in a certain range.

In the table 1 the results of a series of experiments to verify the accuracy of the developed method of identification of bot profiles are show. The following notations of attack models are used: RA – random attack, AA – average attack, PA – popular attack.

**Table 1**

The results of testing the accuracy of the developed method of identifying bot profiles in the recommendation system based on graph clustering and analysis of user actions for different models of attacks and different numbers of bots and attack targets

No. of experiment	Percentage of bots among the system profiles, %	Number of targets for each bot, pcs.	Attack type	Number of correctly recognized targets, %	Number of incorrectly recognized as targets, %	Precision of bot detection	Recall of bot detection	Type I errors, %	Type II errors, %	RMSE of bot recognition
1	5	1	RA	100.000	24.1206	0.666666	0.800000	2.1052	20.0000	0.173205
2	5	5	RA	40.0000	22.0513	0.571428	0.800000	3.1578	20.0000	0.200000
3	5	10	RA	60.0000	21.0526	0.800000	0.800000	1.0526	20.0000	0.141421
4	10	1	RA	0.00000	22.1106	0.375000	0.600000	5.2631	40.0000	0.264575
5	10	5	RA	60.0000	24.1025	0.538461	0.700000	6.6666	30.0000	0.300000
6	10	10	RA	90.0000	21.5789	0.888888	0.800000	1.1111	20.0000	0.173205
7	20	1	RA	100.000	30.6532	0.692307	0.900000	4.4444	10.0000	0.223606
8	20	5	RA	100.000	23.5897	0.700000	0.700000	7.5000	30.0000	0.346410
9	20	10	RA	80.0000	18.9473	0.850000	0.850000	3.7500	15.0000	0.244948
10	30	1	RA	100.000	33.6683	0.823529	0.700000	3.7500	30.0000	0.300000
11	30	5	RA	80.0000	29.7435	0.896551	0.866666	4.2857	13.3333	0.264575
12	30	10	RA	80.0000	23.6842	0.884615	0.766666	4.2857	23.3333	0.316227
<b>Average:</b>				<b>74.1666</b>	<b>24.6085</b>	<b>0.723953</b>	<b>0.773611</b>	<b>3.9476</b>	<b>22.6388</b>	<b>0.245681</b>
13	5	1	AA	100.000	18.0905	0.625000	1.000000	0.0000	3.26080	0.173205
14	5	5	AA	100.000	21.0257	0.555555	1.000000	0.0000	4.39560	0.200000
15	5	10	AA	60.0000	21.5789	1.000000	1.000000	0.0000	0.00000	0.000000
16	10	1	AA	100.000	24.6231	0.588235	1.000000	0.0000	7.77770	0.264575
17	10	5	AA	60.0000	25.6410	0.900000	0.900000	1.1111	10.0000	0.141421
18	10	10	AA	70.0000	25.2631	0.833333	1.000000	2.2222	0.00000	0.141421
19	20	1	AA	100.000	25.1256	0.740740	1.000000	8.7500	0.00000	0.264575
20	20	5	AA	100.000	22.0512	0.850000	0.850000	3.7500	15.0000	0.244948
21	20	10	AA	60.0000	13.6842	0.900000	0.900000	2.5000	10.0000	0.200000
22	30	1	AA	60.0000	14.5728	0.909090	1.000000	4.2857	0.00000	0.173205
23	30	5	AA	100.000	17.9487	0.928571	0.866666	2.8571	13.3333	0.244948
24	30	10	AA	60.0000	11.5789	0.925925	0.833333	2.8571	16.6666	0.264575
<b>Average:</b>				<b>80.8333</b>	<b>20.0986</b>	<b>0.813037</b>	<b>0.945833</b>	<b>2.3611</b>	<b>6.70283</b>	<b>0.192739</b>
25	5	1	PA	75.8793	24.1206	0.714285	1.000000	2.1052	0.00000	0.141421
26	5	5	PA	40.0000	17.9488	0.625000	1.000000	3.1578	0.00000	0.173205
27	5	10	PA	60.0000	22.6315	0.416666	1.000000	7.3684	0.00000	0.264575
28	10	1	PA	100.000	22.1105	0.666666	1.000000	5.5555	0.00000	0.223606
29	10	5	PA	80.0000	30.2564	0.900000	0.900000	1.1111	0.10000	0.141421
30	10	10	PA	90.0000	19.4736	0.888888	0.800000	1.1111	0.20000	0.173205
31	20	1	PA	0.00000	30.1507	0.000000	0.000000	6.2500	1.00000	0.500000
32	20	5	PA	100.000	14.3589	0.800000	0.600000	3.7500	0.40000	0.331662
33	20	10	PA	100.000	17.8947	0.869565	1.000000	3.7500	0.00000	0.173205
34	30	1	PA	100.000	28.6432	0.909090	1.000000	4.2857	0.00000	0.173205
35	30	5	PA	100.000	25.6410	0.900000	0.600000	2.8571	0.40000	0.374165
36	30	10	PA	100.000	17.3684	0.892857	0.833333	4.2857	0.16666	0.282842
<b>Average:</b>				<b>78.8232</b>	<b>22.5498</b>	<b>0.715251</b>	<b>0.811111</b>	<b>3.7990</b>	<b>0.18889</b>	<b>0.246043</b>

To assess the quality of the developed method the following metrics were selected.

*Precision* – a measure that characterizes how many positive triggers of the information security subsystem of bot detection are correct. It was calculated by the formula:

$$precision = \frac{tp}{tp + fp}, \quad (11)$$

where  $tp$  – is a correctly detected bot;  $fp$  – is an authentic user incorrectly identified as a bot.

*Recall (or Sensitivity)* – a measure that characterizes the ability of the information security subsystem to correctly detect the profiles of bots without taking into account false positives. It was determined by the formula:

$$recall = \frac{tp}{tp + fn} \quad (12)$$

where  $tp$  – is a correctly detected bot;  $fn$  – is a bot incorrectly identified as an authentic user.

*RMSE* – root mean square error:

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}} \quad (13)$$

As shown by a series of experiments from Table 1, the precision of bot detection by the developed method in average is 0.72 for a random attack, 0.81 for an average attack, and 0.71 for a popular attack.

The accuracy of bot recognition by the developed method for the popular attack model was also investigated in more detail (Table 2).

**Table 2**

The average values of the accuracy of the developed method for different numbers of information attack targets in bots

No	Percentage of bots among the system profiles, %	Number of targets for each bot, pcs.	Precision of bot detection	Recall of bot detection	RMSE of classification of users on bots and authentic
1	10	1	0.571391191	0.843750000	0.240591404
2	10	5	0.750070242	0.925000000	0.208735147
3	10	10	0.751688997	0.937500000	0.208005683
4	10	15	0.757313520	0.950000000	0.194432974
5	10	20	0.759444202	0.931250000	0.191765580
6	10	25	0.780512370	0.925000000	0.189773836
7	10	30	0.781431763	0.875000000	0.192072089
8	10	35	0.684367195	0.737500000	0.242168980
9	10	40	0.676923250	0.581250000	0.267032399
10	10	45	0.641781656	0.568750000	0.274543689
<b>Average:</b>			<b>0.715492439</b>	<b>0.827500000</b>	<b>0.220912178</b>

The experiments have shown that the detection precision of bots by the method developed for a popular attack in average is 0.71, completeness is 0.82, and RMSE is 0.22. The worst results were obtained when the target for the attack of bots was single, then the accuracy of the developed method dropped to 0.57 on average. The highest precision was observed when there were 25-30 targets of the attacks, in that case it reached a value of 0.78.

### 3. Conclusions

Method of detecting botnets in the recommendation system based on graph clustering and analysis of user actions was developed. This method allows to detect botnets and distinguish them by sets of attack objects. Also, appropriate algorithms have been developed. A subsystem of information security of the recommendation system has been developed. This subsystem consists of a method of detecting an information attack on the recommendation system based on trend analysis of item ratings

and a method of detecting botnets in the recommendation system based on graph clustering and analysis of user actions. The developed information security subsystem allows providing higher robustness of the system to external destabilizing factors represented by information attacks in contrast to existing methods.

Our experiments have shown that the precision of bot detection by the developed method averages 0.72 for a random attack, 0.81 for a medium attack, and 0.71 for a popular attack. The worst results were obtained when the target for the attack of bots was single, then the accuracy of the developed method, for example, for a popular attack averages 0.57. The highest precision was observed when there were 25-30 targets of the attack and then it reached a value of 0.78 for a popular attack. In case of a successful bot attack which have shifted target ratings, the presence of a botnet (and a significant percentage of bots from it) was always detected. All user profiles identified as bots should be verified using existing bot identification methods based on statistical analysis of individual profile data. This allows to refine the results and remove from this set of user profiles those are mistakenly identified as bots.

## 4. References

- [1] T. Kumari, B. Punam, A Comprehensive Study of Shilling Attacks in Recommender Systems, *IJCSI International Journal of Computer Science Issues* 14(4) (2017). URL: <https://www.ijcsi.org/articles/A-comprehensive-study-of-shilling-attacks-in-recommender-systems.php>.
- [2] P. Kaur, S. Goel, Shilling attack models in recommender system, in: *Proceedings of the International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, 2016, pp. 1-5. URL: <https://ieeexplore.ieee.org/document/7824865/>.
- [3] I. Gunes, C. Kaleli, A. Bilge, H. Polat, Shilling attacks against recommender systems: a comprehensive survey, *Artificial Intelligence Review* 42 (2014). URL: <https://doi.org/10.1007/s10462-012-9364-9>.
- [4] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), *Recommender Systems Handbook*, Springer, Boston, 2011. URL: <https://doi.org/10.1007/978-0-387-85820-3>.
- [5] B. Mobasher, R. Burke, R. Bhaumik, C. Williams, Effective attack models for shilling item-based collaborative filtering system, in: *Proceedings of the 2005 WebKDD Workshop*, held in conjunction with ACM SIGKDD 2005, Chicago, Illinois, 2005, pp. 13-23. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.7759&rep=rep1&type=pdf#page=23>.
- [6] S. K. Lam, J. Riedl, Shilling recommender systems for fun and profit, in: *Proceedings of the 13th International World Wide Web Conference*, 2004, pp. 393-402. URL: <https://dl.acm.org/doi/10.1145/988672.988726>.
- [7] O. V. Kurban, Modern information wars in social online networks, *Information Society* 23 (2016) 85-90. URL: [http://nbuv.gov.ua/UJRN/is\\_2016\\_23\\_15](http://nbuv.gov.ua/UJRN/is_2016_23_15). (in Ukrainian)
- [8] D. A. Gubanov, D. A. Novikov, A. G. Chhartishvili, *Social networks: models of information influence, management and confrontation*, Publishing House of Physical and Mathematical Literature, Moscow, 2010. (in Russian)
- [9] V. P. Horbulin, O. H. Dodonov, D. V. Lande, *Information operations and security of society: threats, counteraction, modeling: monograph*, Intertechnology, Kyiv, 2009. (in Ukrainian)
- [10] H. A. Ostapenko, *Information operations and attacks in socio-technical systems*, Hotline - Telecom, Moscow, 2007. (in Russian)
- [11] V. M. Bohush, O. K. Yudin, *Information security of the state*, MK-Press, Kyiv, 2005. (in Ukrainian)
- [12] O. S. Ulichev, Ye. V. Meleshko, D. Sawicki, S. Smailova, Computer modeling of dissemination of informational influences in social networks with different strategies of information distributors, in: *Proc. SPIE 11176, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, Wilga, Poland, 2019, 111761T.
- [13] B. Mobasher, R. Burke, R. Bhaumik, C. Williams, Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness, *ACM Transactions on Internet Technology* 7(4) (2007). URL: <https://doi.org/10.1145/1278366.1278372>.

- [14] P. A. Chirita, W. Nejdl, C. Zamfir, Preventing shilling attacks in online recommender systems, in: Proceedings of the ACM Workshop on Web Information and Data Management, 2005, pp. 67-74.
- [15] M. P. O'Mahony, N. J. Hurley, G. C. M. Silvestre, Promoting recommendations: An attack on collaborative filtering, in: A. Hameurlain, R. Cicchetti, R. Traunmüller (Eds.), DEXA 2002, volume 2453 of Lecture Notes in Computer Science, 2002, pp. 494-503.
- [16] W. Zhou, J. Wen, Q. Qu, J. Zeng, T. Cheng, Shilling attack detection for recommender systems based on credibility of group users and rating time series, PLoS ONE 13(5): e0196533 (2018). URL: <https://doi.org/10.1371/journal.pone.0196533>.
- [17] W. Zhou, J. Wen, Y. S. Koh, S. Alam, G. Dobbie, Attack detection in recommender systems based on target item analysis, in: International Joint Conference on Neural Networks (IJCNN 2014), Beijing, 2014, pp. 332-339. URL: <https://ieeexplore.ieee.org/document/6889419>.
- [18] Ye. V. Meleshko, Analysis of the structure of a social network in terms of information security, in: Proceedings of the XVIII International scientific-practical seminar "Combinatorial configurations and their applications", Kirovograd, Ukraine, Apr. 2016, pp. 93-97. (in Ukrainian)
- [19] Ye. Meleshko, The method of detecting information attack objects in recommendation system based on the analysis of rating trends, Innovative technologies and scientific solutions for industries 3 (13) (2020) 52-57. URL: <https://doi.org/10.30837/ITSSI.2020.13.052>.
- [20] Ju. A. Kalush, V. M. Loginov, Hurst index and its hidden properties, Siberian Journal of Industrial Mathematics 5(4) (2002) 29-37.
- [21] Ye. Meleshko, O. Drieiev, H. Drieieva, Method of identification bot profiles based on neural networks in recommendation systems, Advanced Information Systems 4(2) (2020) 24-28. URL: <https://doi.org/10.20998/2522-9052.2020.2.05>.
- [22] A.-L. Barabási, Network science, Cambridge University Press, 2018, URL: <http://networksciencebook.com/>.
- [23] E. S. Nykyshyn, Methods for identifying communities in social graphs, 2016. URL: <http://www.machinelearning.ru/>. (in Russian)
- [24] N. F. Gusarova, Analysis of social networks. Basic concepts and metrics, ITMO University, Saint Petersburg, 2016. (in Russian)
- [25] M. I. Kolomejchenko, I. V. Poljakov, A. A. Chepovskij, A. M. Chepovskij, Selection of communities in the graph of interacting objects, Fundamental and Applied Mathematics 21(3) (2016) 131-139. (in Russian)
- [26] J. Foreman, Data Smart: Using Data Science to Transform Information into Insight, 1st edition, Wiley, 2013.
- [27] M. Plantić, M. Crampes, Survey on Social Community Detection, in: N. Ramzan, R. van Zwol, J. S. Lee, K. Clüver, X.S. Hua (Eds.), Social Media Retrieval, Computer Communications and Networks, Springer, London, 2013, pp. 65-85. URL: [https://doi.org/10.1007/978-1-4471-4555-4\\_4](https://doi.org/10.1007/978-1-4471-4555-4_4)
- [28] Neo4j Documentation. Official website of the graph database Neo4j, 2021. URL: <https://neo4j.com/docs/>.
- [29] Ye. Meleshko, Computer model of virtual social network with recommendation system, Innovative technologies and scientific solutions for industries 2(8) (2019) 80-85. URL: <https://doi.org/10.30837/2522-9818.2019.8.080>.
- [30] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of Modern Physics 74 (2002) 47-97. URL: <https://doi.org/10.1103/RevModPhys.74.47>.
- [31] M. M. Bernovskiy, N. N. Kuziurn, Random graphs, models and generators of scaleless graphs, Proceedings of the Institute for System Programming, Russian Academy of Sciences 22 (2012) 419-432. URL: <https://cyberleninka.ru/article/n/sluchaynye-grafy-modeli-i-generatory-bezmasshtabnyh-grafov>. (in Russian)
- [32] V. A. Traag, Algorithms and Dynamical Models for Communities and Reputation in Social Networks, Springer International Publishing, 2014. URL: <https://doi.org/10.1007/978-3-319-06391-1>.
- [33] F. M. Harper, J. A. Konstan, The MovieLens Datasets: History and Context, ACM Transactions on Interactive Intelligent Systems (TiiS), 5(4) (2015). URL: <https://doi.org/10.1145/2827872>.