Mathematical and Computational Applications

MDPI

Article

Bot Detection on Social Networks Using Persistent Homology

Minh Nguyen [1], Mehmet Aktas [1,*] and Esra Akbas [2]

[1] Department of Mathematics and Statistics, University of Central Oklahoma, Edmond, OK 73034, USA; mnguyen36@uco.edu

[2] Department of Computer Science, Oklahoma State University, Stillwater, OK 74078, USA; eakbas@okstate.edu

* Correspondence: maktas@uco.edu

Received: 24 July 2020; Accepted: 1 September 2020; Published: 4 September 2020

Abstract: The growth of social media in recent years has contributed to an ever-increasing network of user data in every aspect of life. This volume of generated data is becoming a vital asset for the growth of companies and organizations as a powerful tool to gain insights and make crucial decisions. However, data is not always reliable, since primarily, it can be manipulated and disseminated from unreliable sources. In the field of social network analysis, this problem can be tackled by implementing machine learning models that can learn to classify between humans and bots, which are mostly harmful computer programs exploited to shape public opinions and circulate false information on social media. In this paper, we propose a novel topological feature extraction method for bot detection on social networks. We first create weighted ego networks of each user. We then encode the higher-order topological features of ego networks using persistent homology. Finally, we use these extracted features to train a machine learning model and use that model to classify users as bot vs. human. Our experimental results suggest that using the higher-order topological features coming from persistent homology is promising in bot detection and more effective than using classical graph-theoretic structural features.

Keywords: bot detection; ego network; simplicial complex; persistent homology

1. Introduction

Online social networks have been an excellent platform for exchanging ideas and sharing information. On the other hand, excessive utilization of social media may cause various types of illegal activities, such as spam, fake news, and rumor spreading, produced by abnormal users. Most of these activities have automated behavior patterns made by automated websites or apps, which are called bots. It is critical to detect and classify humans vs. bots in the social network to maintain community health, network security and privacy and prevent negative effects on public and individual safety.

Two types of features are used for bot detection on social networks in literature: structural features and behavioral features. While the users' activities and profiles on the social media platform such as tweet content, tweeting behavior, and account properties like external URL ratio are considered as behavioral features, social network topology such as degree, edge count, average betweenness centrality and brokerage are considered as structural features. Studies suggest that behavioral and structural features are different between humans vs. bots.

The identification and suspension of bots are challenging problems for several reasons. First of all, classical structural features that are employed in bot detection, such as degree and betweenness centrality, are usually local and mainly based on differences between either vertex or edge information in networks, or correlations without considering the network topology. On the other hand, other topological structures, such as the connected components and structural holes, also give important

information about network topology [1]. As it is mentioned in [2], the existence and distribution of structural holes in networks can be used as important topological features for network comparison and classification.

Secondly, the classical structural features employ only pairwise relations between users. However, since human communication and interactions can occur in groups of three or more nodes as we see in different real-world applications, we cannot simply describe them as pairwise relations [3]. In social networks, many connections and relationships do not take place between pairs of nodes, but rather are multiple interactions at the level of groups of nodes. Likewise, when we look at the behavioral dynamics of bot and human, we see that humans are constantly exposed to posts and messages by other users, so their probability of engaging in multiple social interactions is bigger than bots [4]. The higher-order multiple interactions among users should also be investigated for better detection. Therefore, there is a critical need for the bot detection methods that take both structural holes and higher-order interactions into consideration.

In this paper, we develop a novel topological method for bot detection problems using both behavioral features and structural features that employ structural holes and higher-order interactions. We first represent each user with a weighted ego network of the user in the network where the edge weights keep the behavioral features. Then, we study the topology of higher-order network structures of these ego networks to extract the structural features, such as structural holes, using persistent homology. Persistent homology is a computational topology tool that extracts topological features of data that persist across multiple scales. The basic idea is giving shape to data by replacing it with a family of simplicial complexes, which can be considered as a higher dimensional generalization of graphs, and then studying the topology of the resulted shapes to gain knowledge about it [5]. It has found applications in many different domains such as biological systems [6], computer vision [7], social network analysis [8] and signal processing [9]. Finally, we employ machine learning algorithms on these features to first reduce the dimension of the features and then classify the users as bot vs. human. To the best of our knowledge, there is no study that uses persistent homology in bot detection in the literature, thus our study first examines how to use persistent homology for the bot detection problem.

The paper is structured as follows. In Section 2, we give the necessary background for our method. We first give a formal definition to network and ego network of a vertex in the network, then define simplicial complexes and persistent homology. We also provide related work in this section. In Section 3, we introduce the proposed bot detection model by explaining our weighted ego network representation and topological feature extraction method. In Section 4, we present our results on a Twitter data set and compare our results with classical centrality measures. Our final remarks with future work directions are found in Section 5.

2. Background

In this section, we discuss the preliminary concepts for networks, centrality measures, simplicial complexes, and persistent homology. We also elaborate on related work with a particular focus on bot detection and persistent homology in network mining.

2.1. Preliminaries

2.1.1. Graphs

Graphs are structured data representing relationships between objects [10,11]. They are formed by a set of *vertices* (also called nodes) and a set of *edges* that are connections between pairs of vertices. We can see the applications of graph data in many different areas such as (1) social networks consisting of individuals and their interconnections such as Facebook, coauthorship and citation networks of scientists [12–14], (2) protein interaction networks from biological networks that link proteins, which work together to perform some particular biological functions [15,16].

In a formal definition, a graph (network) G is a pair of sets $G = (V, E)$ where V is the set of vertices and $E \subset V \times V$ is the set of edges of the network. There are various types of networks that represent the differences in the relations between vertices. While in an *undirected network*, edges link two vertices symmetrically, in a *directed network*, edges link two vertices asymmetrically. If there is a score for the relationship between vertices that could represent the strength of interaction, it is represented as a *weighted network*. In a weighted network, a weight function $W : E \to \mathbb{R}$ on edges is defined to assign a weight for each edge. Furthermore, if vertices of a graph have a set of attributes describing their properties, such as gender, education, language in social network graphs, we call this graph as an *attributed graph*. In many different real world applications, graphs are modeled in the form of attributed graphs. For example, in social networks such as Facebook and Twitter networks, user nodes have multiple attributes (e.g., education degree, mother tongue, number of followers/followees).

Lastly, we define *ego network* of a vertex in a given network, which is the crucial network structure that we employ for bot detection in this paper.

Definition 1. *For a given graph $G = (V, E)$, k-hop ego network of a vertex v in G is the subgraph $S = (V_S, E_S) \subseteq G$ where V_S includes the ego node v and all other nodes in V that can be reached in k steps from v and E_S includes all the edges between the vertices in V_S in G.*

Example 1. *In Figure 1, we present an ego network of the ego vertex, red node. The subgraph in the green region is the 1-hop ego network, the blue region is the 2-hop ego network and the purple region is the 3-hop ego network.*

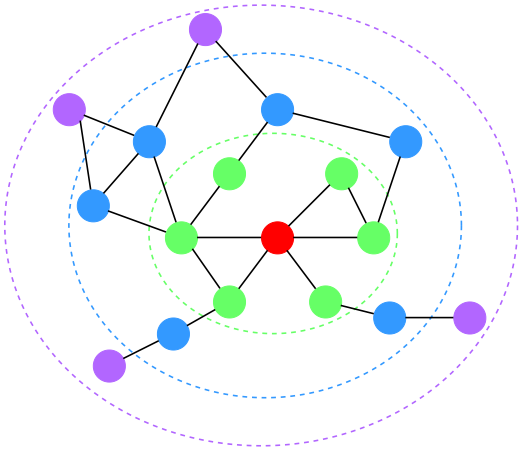


Figure 1. Ego networks of the ego vertex, the red node.

In this paper, we only use the 1-hop ego networks of vertices. Hence, in the rest of the paper, when we say ego network, we actually mean 1-hop ego network.

2.1.2. Centrality Measures

After obtaining an ego network for each user, we can obtain structural features of the ego using the classical centrality measures. The centrality measures are useful for the evaluation of the node importance in networks. Here, we list the commonly used centrality measures in the literature that we later employ to compare with our method in Section 4.3.

1. **In-degree and out-degree centrality** [17]: While the in-degree centrality is the number of incoming edges to each node, the out-degree centrality is the number of outgoing edges from each node.

2. **Betweenness centrality** [17]: This centrality measures how often each graph node appears on the shortest path between two nodes in the graph. Since there can be several shortest paths between two graph nodes s and t, the centrality of node u is

$$bc(u) = \sum_{s,t \neq u} \frac{n_{st}(u)}{N_{st}}$$

where $n_{st}(u)$ is the number of shortest paths from s to t that pass through node u, and N_{st} is the total number of shortest paths from s to t.

3. **In-closeness and out-closeness centrality** [17]: The in-closeness and out-closeness centralities use the inverse sum of the distance from a node to all other nodes in the graph. The centrality of a node i is

$$cc(i) = \left(\frac{A_i}{N-1}\right)^2 \frac{1}{C_i}$$

where A_i is the number of reachable nodes from node i (not counting i), N is the number of nodes in the graph and C_i is the sum of the distances from node i to all reachable nodes. For in-closeness, the distance measure is from all nodes to node i whereas for out-closeness, it is from node i to all other nodes.

4. **Pagerank centrality** [18]: This centrality uses the random walks on the graph. At each node in the graph, the next node is chosen with the probability from the set of successors of the current node. The centrality score is the average time spent at each node during the random walk.
5. **Hubs and authorities centrality** [19]: These centrality scores are two linked centrality measures that are recursive. The hubs score of a node is the sum of the authorities scores of all its successors. Similarly, the authorities score is the sum of the hubs scores of all its predecessors. The sum of all hubs scores is 1 and the sum of all authorities scores is 1. These scores can be interpreted as the left (hubs) and right (authorities) singular vectors corresponding to the largest singular value of the adjacency matrix.

2.1.3. Simplicial Complex

Simplicial complexes are topological objects in which building blocks are *simplices* (plural for simplex). In Euclidean space, an n-dimensional *simplex* (in short n-simplex) is a convex hull of $n+1$ affinely independent points. For example, a 0-simplex is a point, a 1-simplex is two points connected with an edge, a 2-simplex is a filled triangle and a 3-simplex is a filled tetrahedron.

A *simplicial complex* K is a topological object built from a finite collection of simplices satisfying the following two properties

1. Every face of a simplex in K must also be in K
2. For any two simplices σ_1 and σ_2 in K, if $\sigma_1 \cap \sigma_2 \neq \emptyset$, then $\sigma_1 \cap \sigma_2$ is a common face of both σ_1 and σ_2.

Since graph vertices do not have embedding in any Euclidean space necessarily, here we also give the definition of the abstract simplicial complex.

Definition 2. *An abstract simplicial complex is a pair $A = (V, \Sigma)$ where V is the set of vertices of A and Σ is a subset (called the simplices) of the collection of all non-empty subsets of V, satisfying the conditions that if $\sigma \in \Sigma$, and $\emptyset \neq \tau \subseteq \sigma$, then $\tau \in \Sigma$. Simplices consisting of exactly two vertices are called edges.*

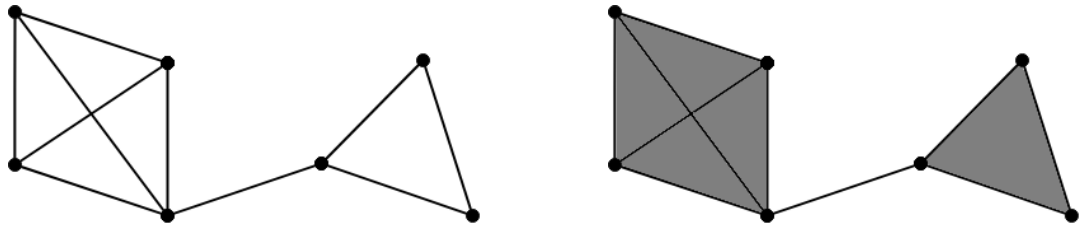
In this paper, we represent networks as an abstract simplicial complex and employ the topological properties of the simplicial complex for bot detection. The properties we study in a simplicial complex are n-dimensional holes. An n-dimensional hole can be considered as voids surrounded by n-simplices. For example, 0-dimensional holes are connected components, 1-dimensional holes are loops bounded by 1-simplicies (i.e., edges), 2-dimensional holes are voids surrounded by 2-simplices (i.e., triangles).

The *simplicial homology* is the area in algebraic topology that extracts the holes in any dimension in simplicial complexes by associating algebraic structures, such as vector spaces, to them. For a given simplicial complex K, simplicial homology associates vector spaces $H_i(K)$ for $i = \{0, 1, 2, \ldots\}$ where the dimension of $H_0(X)$ gives the number of connected components, $H_1(X)$ gives the number of holes, $H_2(X)$ gives the number of voids and so on. We refer readers to [20] for the details of the simplicial homology.

Furthermore, we use the *clique complex* structure to represent undirected networks as simplicial complexes.

Definition 3. *The clique complex $Cl(G)$ of an undirected graph G is a simplicial complex where vertices of G are its vertices and each k-clique, i.e., the complete subgraphs with k vertices, in G corresponds to a $(k-1)$-simplex in $Cl(G)$.*

Example 2. *In Figure 2, we present a toy graph G (a) and its clique complex (b). The graph has a 4-clique on the left, 2-clique in the middle and 3-clique on the right. Hence, its clique complex has a 3-simplex (tetrahedron) on the left, a 1-simplex (edge) in the middle and a 2-simplex (triangle) on the right.*



(**a**) A graph G (**b**) The clique complex $Cl(G)$ of the graph on the left

Figure 2. An example for constructing the clique complex of a graph (borrowed from [21]).

2.1.4. Persistent Homology

For a given undirected network $G = (V, E)$, the homology of its clique complex may not give interesting information. For example, if we would like to compare two connected graphs, the number of connected components (0-dimensional holes) is 1 for both, hence this information is inadequate for comparison. As a solution, instead of working the clique complex of the whole graph, we can induce a family of simplicial complexes out of the clique complex of the subgraphs of G and track how homology changes in this family. This nested family of simplicial complexes is called *filtration*, here is the formal definition of the filtration of a graph.

Definition 4. *Let $G = (V, E)$ be a graph and $G_\delta \subseteq G$ be a nested family of subgraphs for $\delta \in \mathbb{R}_{\geq 0}$ where $G_{\delta_i} \subseteq G_{\delta_j}$ for $0 \leq \delta_i \leq \delta_j$. Then the nested induced family of simplicial complexes out of the clique complex of G_δ*

$$\{Cl(G_{\delta_i}) \hookrightarrow Cl(G_{\delta_j})\}_{0 \leq \delta_i \leq \delta_j}.$$

is called filtration of G.

One of the popular filtration methods defined on undirected weighted networks is called *weight rank clique filtration*. This filtration, first defined in [22], uses weights as threshold values δ. The authors first rank the edge weights from $w_{\min}$ to $w_{\max}$ (it can also be done in other direction, from $w_{\max}$ to $w_{\min}$ as well) and let the parameter increase from $w_{\min}$ to $w_{\max}$. At each step δ, they just consider the thresholded subgraph $G_\delta \subseteq G$ that is the subgraph with edges of weight larger than δ. Then, they create the clique complex of G_δ to obtain the filtration.

Example 3. *Figure 3 depicts an example for the weight rank clique filtration. For the given weighted network on the left, we first add its vertices for $\delta = 0$ to the filtration. Then, we add the edges with weight 1 when $\delta = 1$ and keep adding other edges when δ equals to their weights. We further add higher-dimensional simplices to the filtration, such as triangles, whenever all their edges are added.*

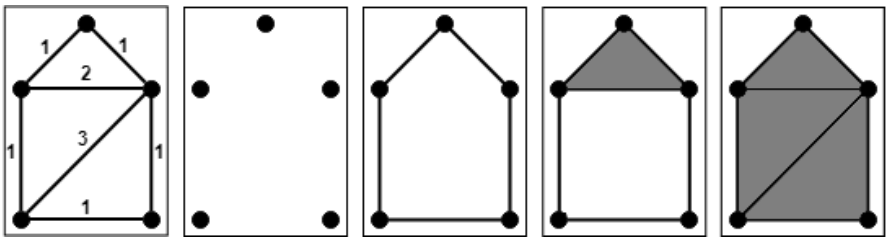


Figure 3. The weight rank clique filtration of the weighted network on the left.

While ranging the δ value in a filtration, some holes (i.e., homological features) may appear and then disappear and the intervals for δ where each hole is *persistent* are the topological features of the network. We record the *birth* of a hole, i.e., the δ value when the hole appears, and the *death* of the hole, i.e., the δ value when the hole disappears. The *persistent homology* just tracks the birth and death of the holes in a filtration for a fixed dimension. For a given network and a filtration, we can encode the birth and death of the holes in a *persistent barcode* as a multiset of intervals bounded below [23].

Example 4. *Figure 4 has the 0- and 1-dimensional persistence barcodes of the filtration in Figure 3. For the 0-dimensional persistent barcode, when $\delta = 0$, there are five disconnected vertices, which means there are five connected components in the simplicial complex. That results in five bars are born at the beginning. When $\delta = 1$, the five edges with weight 1 are added that decreases the total number of connected components to one, hence four bars die at $\delta = 1$. After this point, the number of connected components does not change although we add more edges. Hence, the top bar lives forever (arrowhead at the right of that bar implies this fact).*

For the 1-dimensional persistent barcode, since the first 1-dimensional hole (loop) surrounded with five edges appears when $\delta = 1$, there is a bar born at this value in the 1-dimensional barcode. When $\delta = 2$, this loop splits into two loops, but one of them is filled with a triangle. Hence, that bar still lives. When $\delta = 3$, we add an edge that splits the loop with four edges into two triangles, but triangles are automatically filled as they appear since we use the clique complex. That kills the loop with four edges hence the bar born at $\delta = 1$ dies at $\delta = 3$. Hence, we finalize the persistent barcodes for both dimensions.

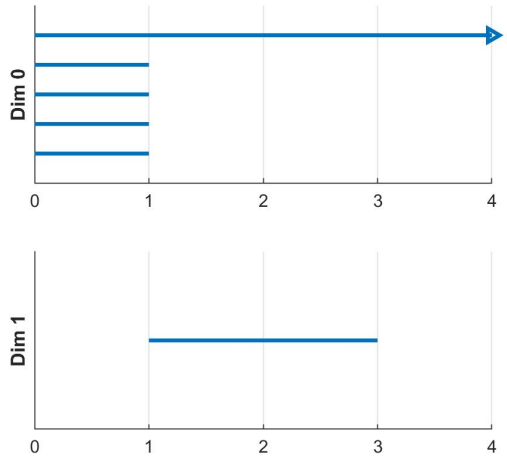


Figure 4. 0-dimensional and 1-dimensional persistent barcodes of the filtration in Figure 3.

Lastly, we can compare two networks by comparing their persistent barcodes using the *bottleneck distance*. The bottleneck distance is a stable and robust distance [24]. If we consider each interval in a persistent diagram as a point (birth, death) in the extended real plane $\bar{\mathbb{R}}^2$ [25], the bottleneck distance measures the distance between two persistence barcodes by the maximum distance between two points (intervals) in a matching from one persistent diagram to the other one. We refer readers to the survey paper [21] for more details of persistent homology on networks.

2.2. Related Work

In this section, we briefly discuss the related work in the areas of bot detection and persistent homology in network mining.

2.2.1. Bot Detection

The malicious activity of bots has led to the necessity of bot detection techniques [26,27]. Most bot detection methods are based on supervised machine learning with different types of extracted features from labeled data [28,29]. Mainly two types of features are used for bot detection on social networks in literature [30,31]: structural features and behavioral features. While the users' activities and profiles on the social media platform such as tweet content, tweeting behavior, and account properties like external URL ratio are considered as behavioral features, social network topology such as number of followers, number of friends, and the follower ratio are considered as structural features [32–34]. Studies suggest that behavioral and structural features are different between humans vs. bots, hence using both of them improves the accuracy of the model [35].

RTbust [36] extract features from retweeting behaviors of users and cluster the users with these features. Accounts belonging to large clusters characterized by malicious retweeting patterns are labeled as bots. Davis et al. [35] propose the BotorNot technique that applies more than one thousand different features including structural features and behavioral features based on account recent activities. They categorized their features into six groups; (1) network features to capture various dimensions of information diffusion patterns; (2) user features related to the account of users; (3) friends features related to an account's social contacts; (4) temporal features to capture timing patterns of content generation and consumption; (5) content features and (6) sentiment features extracted from posts of users.

Although state-of-the-art bot detection models show promising performance, they still face generalization [37] and scalability challenges, which greatly limit their applications. Since Twitter bots are constantly changing their behavior to evade detection and supervised methods are non-adaptive and thus they may not be able to identify novel bot behaviors. Juan et al. [37] evaluate bot classifiers by testing them on unseen bot classes and find out that they do not generalize well to unseen bot classes. Yang et al. [38] propose an efficient method that uses minimal account metadata. They collect all available datasets and combine their features to build a rich model. Moreover, the BotWalk [39] method is proposed as an unsupervised approach to identify evolving bots. They use behavioral features consisting of metadata, content, temporal, and network-based features.

For other bot detection methods, we refer readers to the survey paper [40].

2.2.2. Persistent Homology in Network Mining

Nowadays, persistent homology has found applications in different network mining problems. We can categorize these applications into two groups: single graph analysis and multiple graphs analysis. In the first group, persistent homology helps to detect global structural features of a single network such as the complexity and distributions of strongly connected regions. While some applications use the edge weights for evaluation of a single graph, others analyze the evaluation of a single graph over time. For example, persistent homology is used to analyze brain networks to examine the abnormal white matter in maltreated children [41] and to show that the sparse correlation gets a huge group separation between normal and stress-exposed children [42]. As another example,

the authors in [43] use persistent homology to propose that cancer therapy can be guided by changes in the complexity of protein–protein interaction networks.

In multi graph settings, persistent homology features of graphs are used to compare and classify the graphs. For example, the authors in [44] use persistent homology to find the collaboration patterns in collaboration networks. They first create the networks using five journals from the mathematics community and six journals from the engineering community. They show that they can lower bound distance between two higher-order networks, which is in general computationally expensive, with a computationally less expensive distance, namely the bottleneck distance, between their persistence barcodes. They use these lower bounds to classify the networks, distinguish the collaboration patterns of the engineering and mathematics community, and also discriminate engineering communities with different research interests. In [21], the authors classify ego networks of vertices in a social network using their persistent barcodes as features. In this paper, they study the Amazon purchasing network and classify the ego networks of purchased items based on their category, such as book, music, video. They first define diffusion Fréchet function on vertices to measure the node importance and use sub-level filtration on these function values for persistent barcode creation. Also, in [45], the authors introduce a new weighted-kernel for persistence images and a metric-learning framework to learn the best weight-function for these kernels from labeled data. They apply this learned kernel for the graph classification problem. In [46,47], the authors define a new filtration, called Dowker filtration, which is also sensitive to edge directions on directed networks and use this filtration to classify the hippocampal networks. Horak et al. study random graphs using the clique complex [48]. In [49], the authors use persistent homology for metric graph comparison. In [50], persistent homology is employed to quantify structural changes in time-varying (dynamic) graphs.

For other methods and applications of persistent homology on networks, we refer readers to the survey paper [21].

3. Methodology

In this section, we introduce the main parts of our method. We first show how to create a weighted ego network of a user. In the second part, we explain how we encode and extract the topological features of ego networks using weight rank clique filtration in persistent homology computation. Finally, we outline our bot detection algorithm that employs these topological features.

3.1. Weighted Ego Network Representation

In this paper, we study the social network defined on Twitter where vertices represent users and edges represent follower/followee relation. This network is a directed network since there is a direction in the follower/followee relation. We create an ego network for each user. An ego network of a vertex in a network represents the connection of each vertex to the rest of the network and hence can be used as a distinct feature for each observation. In this study, the ego network of each user is constructed based on the follower-followee relation and initial weight of each edge is calculated by the number of followers and followees possessed by users. Then, we update edge weights based on the strength of the relation. First, we define the *degree of strength* between two vertices as the strength of the relation.

Definition 5. *For any direct edge $e = A \rightarrow B$ from the source node A to the target node B, the degree of strength is measured as*

$$s(e) = ln\left(\frac{B_{pre}}{A_{suc}}\right)$$

where B_{pre} is the number of B's followers (i.e., in-coming edges) and A_{suc} is the number of A's followees (i.e., outgoing edges).

The degree of strength reflects how potentially strong a relationship based on the number of incoming and outgoing edges inherently. A positive degree of strength indicates a strong attractive connection to the end node, while a negative degree of strength shows a potentially weak attractive connection between the two nodes.

Using the degree of strength, we can now define the edge weight.

Definition 6. *Given an ego network $\mathscr{E}$ of a vertex in a network, the weight of any directed edge $e = A \to B$ is assigned as*

$$w(e) = L \cdot \left(1 - \frac{1}{1 + e^{(-s(e)+\mu_{\mathscr{E}})/\sigma_{\mathscr{E}}}}\right)$$

where $s(e)$ is e's degree of strength, L is the scaling parameter, $\mu_{\mathscr{E}}$ and $\sigma_{\mathscr{E}}$ are the mean and the standard deviation of the degree of strength of the edges in the ego network $\mathscr{E}$ respectively.

For the persistent homology computation, we need to scale the degree of strength within the scope of each ego network. We achieve this by implementing the logistic function using the mean and standard deviation for the degree of strength of each ego network. This results in the edge weights defined locally for each ego network, and as a whole can be used as a distinct feature for bot detection.

In our persistent homology computation, we do not take the direction of the edges into consideration and represent the network as an undirected network. In this representation, if there is only one directed edge in the directed network between two vertices, for example $A \to B$, then its corresponding edge weight in the undirected network is the weight of this directed edge. However, if there are edges in both directions in the directed network, for example $A \to B$ and $B \to A$, then we simply take the maximum of these weights as the edge weight in the undirected network. The reason why we take the maximum here is that if there is a strong relation between two users even in one direction, we consider these users as strongly related. With this definition, we now have undirected weighted ego networks defined for each user.

3.2. Topological Feature Extraction

Our next task is to compare and classify the weighted ego networks of users as bot vs. human. For the comparison, we extract topological features of these ego networks using persistent homology. We compute the persistent homology using the weight rank clique filtration (explained in Section 2.1.4). At the beginning of the filtration, We add all vertices. Then, we add the edges in the order based on their weights. Finally, we add any n-simplex at the maximum value of its edge weights for $n > 1$. This construction allows us to stress the importance of the edge weights in the filtration. At the end of the filtration, we encode the topological features of weighted ego networks into persistent barcodes.

Example 5. *In Figure 5, we present examples of ego networks and their corresponding persistent barcodes for a bot user (a) and a human user (c). As we see from the figures, ego networks of a bot and a human user have different network structure and so their persistent barcodes are different.*

After computing persistent homology and creating persistence barcodes for each ego network, we calculate the bottleneck distance between barcodes for a given dimension to compare and compute the similarity of them. We create the similarity matrix of barcodes and also for ego networks of users using the bottleneck distance values between barcodes.

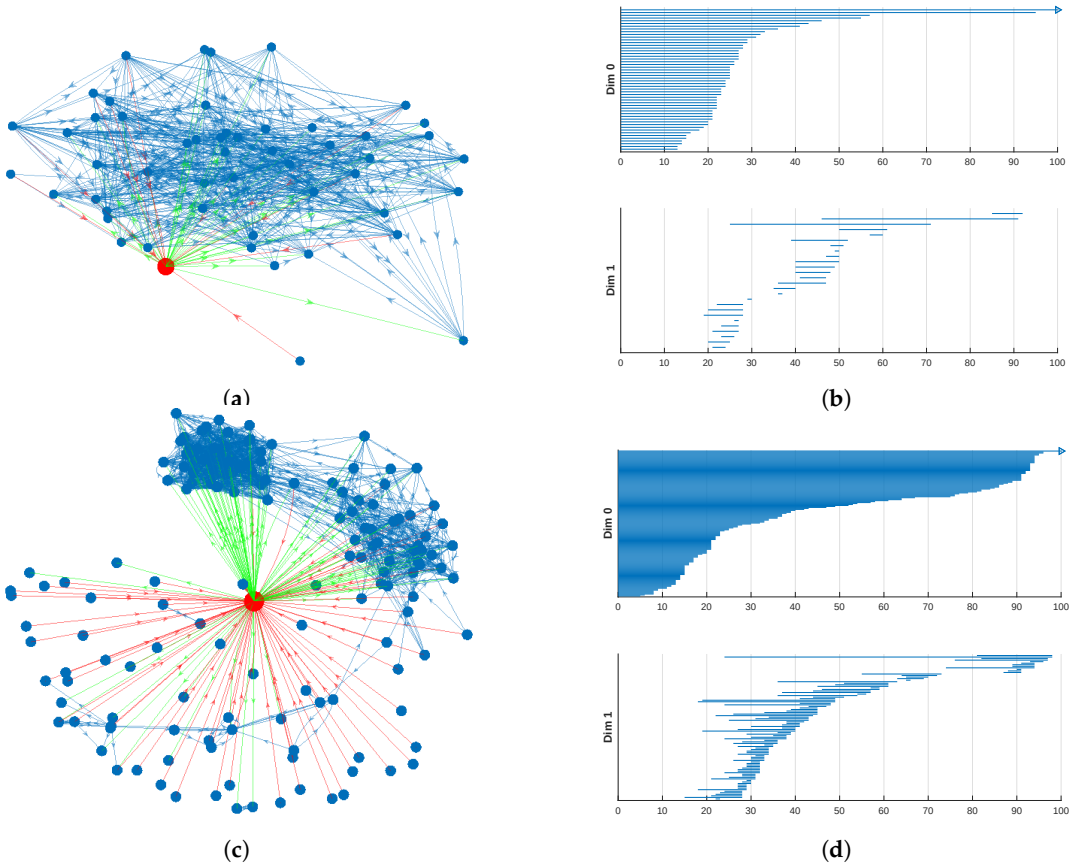


Figure 5. Ego network of a bot user (**a**), a human user (**c**), and their barcodes for dimension 0 and dimension 1, (**b**) and (**d**), respectively. In each ego network, the ego is colored red and its neighbors are colored blue. Also, the edges out-going from the ego are colored green, in-going to the ego are red and other edges between the neighbors are colored blue.

3.3. Bot Detection

In the previous two sections, we first create the weighted ego network for each user and then compute the similarity between ego networks using the bottleneck distance between their persistent barcodes. Now, as the last step, we use this similarity matrix to extract features of ego networks in our detection algorithm. There are two main steps in our detection algorithm: feature extraction and classification.

For feature extraction, we use the multidimensional scaling (MDS) technique. MDS takes the similarity matrix of data objects as an input and gives k-dimensional points in Euclidean space as the output where the pairwise distance between objects is preserved. Hence, after obtaining the similarity matrix between ego networks, we use MDS to extract features.

For classification, we apply the leave-one-out method to n-dimensional linear discriminant analysis (LDA) projection of these k-dimensional scores with $n \leq k$. LDA is used as a tool for classification, dimension reduction, and data visualization. The basic idea in LDA is to maximize the variance of the data and the separation between multiple classes. In the leave-one-out method, one user is selected and left out from the corpus, and the remaining is used for LDA. Then, we project the left-out user's feature vector using this LDA projection. Our main motivation in using LDA is to use it in the leave-one-out method, not for the dimension reduction. However, we can choose n as 2 or 3 for the data visualization proposed. As the final step, we check the k-nearest neighbors in the LDA projection to decide the class of the left-out user.

4. Experiment

In this section, we first introduce the dataset we use in our experiments. Then, we share the detection results for different persistent barcode dimensions and metrics.

4.1. Dataset

We create our dataset using the dataset in [51] that includes labels of Twitter users as human or both. However, this dataset does not include network features of each user as well as other useful predictors such as the number of followers and friends. Also, it does not provide ego networks for each user as well. From this dataset, we select the users whose number of edges is less than 100 since downloading large ego networks is taking too much time on Twitter APIs. As a result, we download the ego networks of 56 users, 37 bots and 19 humans, along with their network features.

4.2. Results

In our experiments, we compute 0-dimensional and 1-dimensional persistent barcodes of the ego networks of users using the weight rank clique filtration. In the filtration, we add the edges using their weight. In the edge weight computation, we select the scale parameter $L = 100$, i.e., the maximum filtration value is 100. Actually, changing this parameter value will only scale the similarity matrix, hence will not change the classification results. For each dimension, we compute the similarity matrix using the bottleneck distance. We then project barcodes into an 8-dimensional Euclidean space by the MDS technique. The reason why we choose 8-dimensional MDS is that later in Section 4.3, we will compare our method with eight different graph-theoretic centrality measures. Furthermore, we create another set of feature vectors to see the performance of both dimensions when used together. We define another similarity matrix using the similarity matrices of 0 dimension and 1 dimension. For given two ego networks, we define this similarity as

$$Bd_{01} = \sqrt{Bd_0^2 + Bd_1^2}$$

where Bd_0 and Bd_1 are the bottleneck distance between their 0-dimensional and 1-dimensional persistent barcodes respectively.

After that, we apply the leave-one-out method to 3-dimensional LDA projection of these 8-dimensional scores. The reason we choose a 3-dimensional LDA projection is to be able to visualize the distribution of each user in 3-D space (Figure 6).

Lastly, we choose $k = 5$ in the nearest neighbor algorithm, i.e., we check the 5-nearest neighbors in the LDA projection. The validated leave-one-out experiment results for different dimensions and evaluation metrics are available in Table 1.

Table 1. Detection results for different metrics and dimensions. The cells for the best result in each column are colored gray.

Dimension	0	1	0&1
Accuracy	75.00	76.79	78.57
Bot-Recall	89.19	83.78	86.49
Bot-Precision	76.74	81.58	82.05
Human-Recall	47.37	63.16	63.16
Human-Precision	69.23	66.67	70.59

As we see in Table 1, the best results other than the bot-recall metric are obtained when we use 0-dimension and 1-dimension features together. The reason is that different dimensions capture different features of bot and human ego networks, hence utilizing these features together in the classifier provides better results. More precisely, in this combined case, our method could correctly detect 78.57% of the users. The recall rates for bots suggest that our method is very effective in detecting

bots. We also observe from the recall rates that while 0-dimension features are the distinct features for bots (89.19% for 0-dimension and 83.78% for 1-dimension), 1-dimension features are the distinct for humans (47.37% for 0-dimension and 63.16% for 1-dimension). Hence, having 1-dimensional loops is an important feature for humans. This is expected since humans tend to have more social interactions than bots, i.e., they are a part of more social circles than bots. Also, as we see from the recall rates, our method could detect bots better than detecting humans. In general, these results provide evidence that the proposed method can be used in the bot detecting problem.

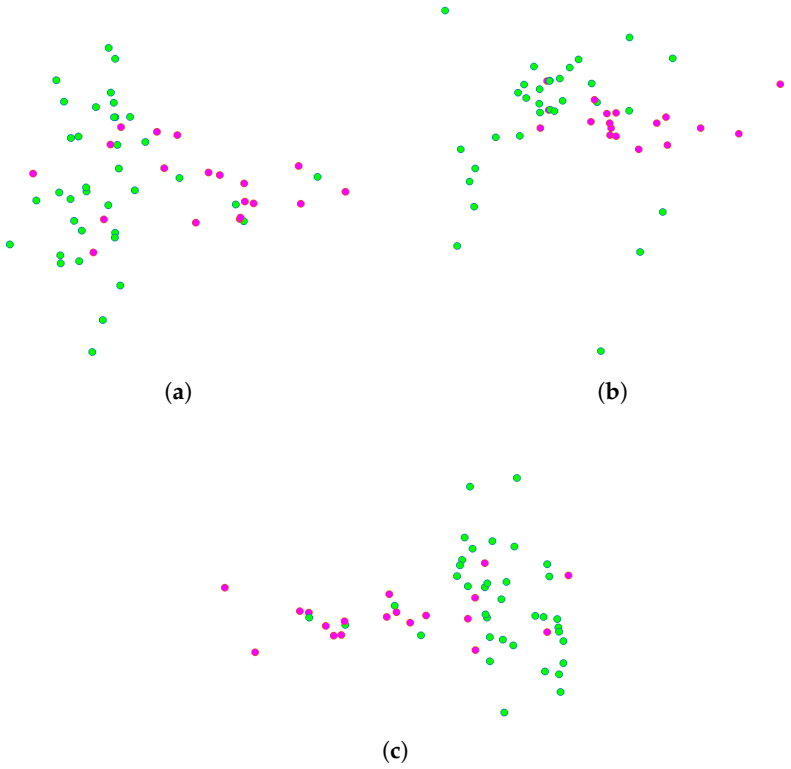


Figure 6. Three dimensional projection by linear discriminant analysis (LDA) for 0-dimension (**a**), 1-dimension (**b**) and 0- and 1-dimensions (**c**). Green points represent bots and pink points represent humans.

4.3. Comparison

In this section, we compare our method with the classical centrality measures defined in Section 2.1.2, namely in-degree, out-degree, betweenness, in-closeness, out-closeness, pagerank, hubs, and authorities centralities. For each ego network, we compute the centrality scores and create an 8-dimensional feature vector for the ego. For a fair comparison, we apply the same classification methods. We use a 3-dimensional LDA projection for the leave-one-out experiments and check the 5-nearest neighbors in the projection to decide the class of the left-out user. The comparison results are available in Figure 7.

As we see in the figure, our method performs better than the baseline method for all different evaluation metrics. The main reason is that these classical centrality measures are usually local and mainly based on differences between either node or edge measurements, or correlations without considering the network topology. On the other hand, persistent homology extract the structural holes and higher-order interactions in networks that can give important information about network topology [1,2].

To check the significance of our accuracy results compared with the baseline method, we use the asymptotic McNemar test [52]. McNemar tests are hypothesis tests that compare two population proportions while addressing the issues resulting from two dependent, matched-pair samples.

Using this test, we statistically assess the accuracies of given two classification models. This test first compares their predicted labels against the true labels, and then it detects whether the difference between the misclassification rates is statistically significant. The test results suggest that the accuracy results are statistically significant for 1-dim, and 0-dim and 1-dim together with the p-values $p = 0.045$ and $p = 0.02$ respectively, and marginally significant for 0-dim with the p-value $p = 0.089$.

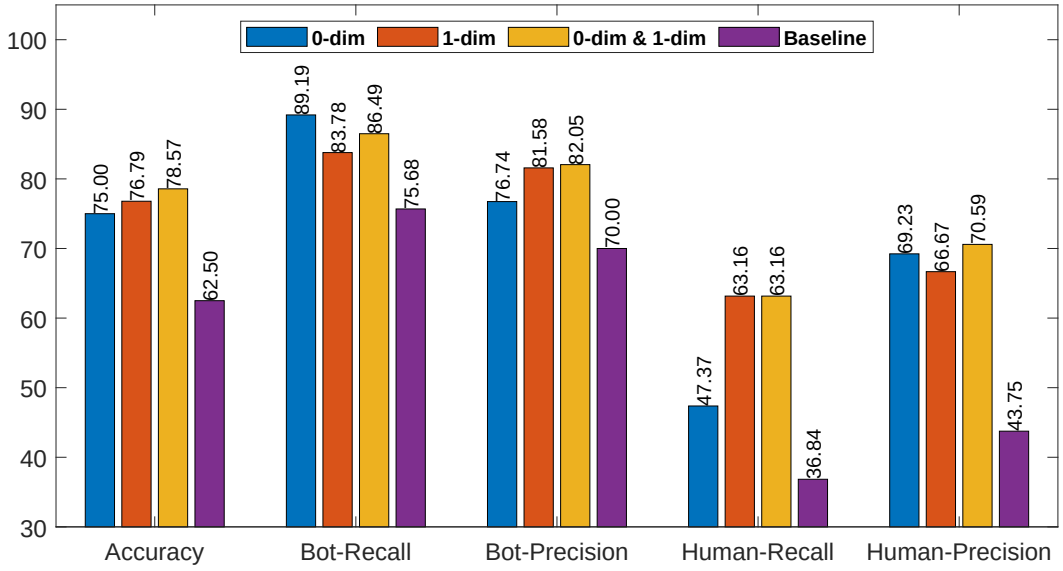


Figure 7. Detection results for different metrics and dimensions.

5. Conclusions

In this paper, we propose a novel method for bot detection in online social networks. We first represent each user with a weighted ego network. We then construct the persistent barcodes of ego networks to encode their higher dimensional topological features as the network features. Next, we compute the similarity matrix of the ego networks using the bottleneck distance between persistent barcodes and project them into an 8-dimensional Euclidean space by the MDS technique. After that, we use the leave-one-out on a 3-dimensional LDA projection of these 8-dimensional MDS scores, and use the k-nearest neighbor classifier to group each user with $k = 5$. The validated leave-one-out experiments show that the proposed method is very effective in bot detection. Furthermore, we also compare our method with the classical graph-theoretic centrality measures. We show that our method is more effective than these measures since it also takes the structural holes and the higher-order interactions in the network into consideration. This paper is the first study in the literature showing the potential of using persistent homology in the bot detection problem.

As future tasks, we plan to apply our method not only to detect bots vs. humans, but also detect different bot types, such as automated, fake pages, trolls, and also human types. Furthermore, we also plan to use other networks on Twitter, such as the retweet network and hashtag network, to check the effectiveness of our method. Also, we would like to use other filtrations on the ego network, especially the ones that are sensitive to edge directions, such as the Dowker filtration [47], for topological feature extraction.

Author Contributions: Conceptualization, M.A. and E.A.; methodology, M.N., M.A. and E.A; software, M.N.; validation, M.N., M.A. and E.A.; data curation, M.N.; writing—original draft preparation, M.N., M.A. and E.A.; writing—review and editing, M.N., M.A. and E.A.; visualization, M.N. and M.A.; supervision, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We would like to thank the anonymous reviewers and the editor for their helpful and constructive comments that greatly contributed to improving the final version of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDS	Multidimensional scaling
LDA	Linear discriminant analysis
dim	Dimension

References

1. Xu, H.; Zhang, J.; Yang, J.; Lun, L. Measurement of Nodes Importance for Complex Networks Structural-Holes-Oriented. In Proceedings of the International Conference of Pioneering Computer Scientists, Engineers and Educators, Harbin, China, 22–28 August 2016; pp. 458–469.
2. Xu, H.; Zhang, J.; Yang, J.; Lun, L. Assessing nodes' importance in complex networks using structural holes. *Int. J. High Perform. Comput. Netw.* **2018**, *12*, 314–323. [CrossRef]
3. Battiston, F.; Cencetti, G.; Iacopini, I.; Latora, V.; Lucas, M.; Patania, A.; Young, J.G.; Petri, G. Networks beyond pairwise interactions: Structure and dynamics. *Phys. Rep.* **2020**, *47*, 777–780. [CrossRef]
4. Pozzana, I.; Ferrara, E. Measuring bot and human behavioral dynamics. *Front. Phys.* **2020**, *8*, 125. [CrossRef]
5. Ghrist, R. Barcodes: The persistent topology of data. *Bull. Am. Math. Soc.* **2008**, *45*, 61–75. [CrossRef]
6. Li, M.; Duncan, K.; Topp, C.N.; Chitwood, D.H. Persistent homology and the branching topologies of plants. *Am. J. Bot.* **2017**, *104*, 349–353. [CrossRef]
7. Adcock, A.; Carlsson, E.; Carlsson, G. The ring of algebraic functions on persistence bar codes. Available online: https://arxiv.org/pdf/1304.0530.pdf (accessed on 2 September 2020).
8. Keil, W.; Aktas, M. Topological Data Analysis of Attribute Networks using Diffusion Frechet Function with Ego-Networks. In Proceedings of the Complex Networks and Their Applications Conference, Cambridge, UK, 11–13 December 2018; pp. 194–196.
9. Erden, F.; Cetin, A.E. Period estimation of an almost periodic signal using persistent homology with application to respiratory rate measurement. *IEEE Signal Process. Lett.* **2017**, *24*, 958–962. [CrossRef]
10. Aggarwal, C.C.; Wang, H. *Managing and Mining Graph Data*; Springer: Berlin, Germany, 2010.
11. Cook, D.J.; Holder, L.B. *Mining Graph Data*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
12. Akbas, E.; Zhao, P. Attributed Graph Clustering: An Attribute-aware Graph Embedding Approach. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, Sydney, Australia, 31 July–3 August 2017; pp. 305–308.
13. Akbas, E.; Zhao, P. Truss-based community search: A truss-equivalence based indexing approach. *Proc. VLDB Endow.* **2017**, *10*, 1298–1309. [CrossRef]
14. Tanner, W.; Akbas, E.; Hasan, M. Paper Recommendation Based on Citation Relation. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3053–3059.
15. Newman, M.J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef]
16. Przulj, N. *Graph Theory Approaches to Protein Interaction Data Analysis*; University of Toronto: Toronto, ON, Canada, 2004.
17. Freeman, L.C. Centrality in social networks conceptual clarification. *Soc. Netw.* **1978**, *1*, 215–239. [CrossRef]
18. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The PageRank Citation Ranking: Bringing Order To The Web*; Stanford InfoLab: Stanford, CA, USA, 1998.
19. Kleinberg, J.M. Authoritative sources in a hyperlinked environment. *J. ACM* **1999**, *46*, 604–632. [CrossRef]
20. Hatcher, A. *Algebraic Topology*; Cambridge University Press: Cambridge, UK, 2002.
21. Aktas, M.E.; Akbas, E.; El Fatmaoui, A. Persistence homology of networks: Methods and applications. *Appl. Netw. Sci.* **2019**, *4*, 61. [CrossRef]
22. Petri, G.; Scolamiero, M.; Donato, I.; Vaccarino, F. Topological strata of weighted complex networks. *PLoS ONE* **2013**, *8*, e66506. [CrossRef] [PubMed]

23. Carlsson, G.; Zomorodian, A.; Collins, A.; Guibas, L.J. Persistence barcodes for shapes. *Int. J. Shap. Model.* **2005**, *11*, 149–187. [CrossRef]
24. Cohen-Steiner, D.; Edelsbrunner, H.; Harer, J. Stability of persistence diagrams. *Discret. Comput. Geom.* **2007**, *37*, 103–120. [CrossRef]
25. Edelsbrunner, H.; Letscher, D.; Zomorodian, A. Topological Tersistence and Simplification. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, USA, 12–14 November 2000.
26. Alothali, E.; Zaki, N.; Mohamed, E.A.; Alashwal, H. Detecting Social Bots on Twitter: A Literature Review. In Proceedings of the 2018 International Conference On Innovations in Information Technology (IIT), Al Ain, UAE, 18–19 November 2018; pp. 175–180.
27. Varol, O.; Ferrara, E.; Davis, C.A.; Menczer, F.; Flammini, A. Online human-bot interactions: Detection, estimation, and characterization. In Proceedings of the Eleventh International AAAI Conference on Web and Social Media, Montreal, QC, Canada, 15–18 May 2017.
28. Yang, K.C.; Varol, O.; Davis, C.A.; Ferrara, E.; Flammini, A.; Menczer, F. Arming the public with artificial intelligence to counter social bots. *Hum. Behav. Emerg. Technol.* **2019**, *1*, 48–61. [CrossRef]
29. Kudugunta, S.; Ferrara, E. Deep neural networks for bot detection. *Inf. Sci.* **2018**, *467*, 312–322. [CrossRef]
30. Wang, A.H. Detecting spam bots in online social networking sites: A machine learning approach. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy, Rome, Italy, 21–23 June 2010.
31. Stringhini, G.; Kruegel, C.; Vigna, G. Detecting spammers on social networks. In Proceedings of the 26th Annual Computer Security Applications Conference, Austin, TX, USA, 6–10 December 2010.
32. Beskow, D.M.; Carley, K.M. Bot conversations are different: Leveraging network metrics for bot detection in twitter. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, 28–31 August 2018.
33. Almaatouq, A.; Shmueli, E.; Nouh, M.; Alabdulkareem, A.; Singh, V.K.; Alsaleh, M.; Alarifi, A.; Alfaris, A.; Pentland, A.S. If it looks like a spammer and behaves like a spammer, it must be a spammer: Analysis and detection of microblogging spam accounts. *Int. J. Inf. Sec.* **2016**, *15*, 475–491. [CrossRef]
34. Bhat, S.Y.; Abulaish, M. Community-based features for identifying spammers in online social networks. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), Niagara, NY, Canada, 25–28 August 2013; pp. 100–107.
35. Davis, C.A.; Varol, O.; Ferrara, E.; Flammini, A.; Menczer, F. BotOrNot: A System to Evaluate Social Bots. In Proceedings of the 25th International Conference Companion on World Wide Web, Montréal, QC, Canada, 11–15 April 2016; pp. 273–274. [CrossRef]
36. Mazza, M.; Cresci, S.; Avvenuti, M.; Quattrociocchi, W.; Tesconi, M. RTbust: Exploiting Temporal Patterns for Botnet Detection on Twitter. In Proceedings of the 10th ACM Conference on Web Science, Association for Computing Machinery, Boston, MA, USA, 30 June–3 July 2019; pp. 183–192. [CrossRef]
37. Echeverrï£¡a, J.; De Cristofaro, E.; Kourtellis, N.; Leontiadis, I.; Stringhini, G.; Zhou, S. LOBO: Evaluation of Generalization Deficiencies in Twitter Bot Classifiers. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 137–146. [CrossRef]
38. Yang, K.C.; Varol, O.; Hui, P.M.; Menczer, F. Scalable and generalizable social bot detection through data selection. In Proceedings of the AAAI Conference on Artificial Intelligence, Hilton, NY, USA, 7–12 February 2020; pp. 1096–1103.
39. Minnich, A.; Chavoshi, N.; Koutra, D.; Mueen, A. BotWalk: Efficient Adaptive Exploration of Twitter Bot Networks. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, Sydney, Australia, 31 July–3 August 2017; pp. 467–474. [CrossRef]
40. Karataş, A.; Şahin, S. A review on social bot detection techniques and research directions. In Proceedings of the Interrnational Security and Cryptology Conference, Ankara, Turkey, 20–21 October 2017; pp. 156–161.
41. Chung, M.K.; Hanson, J.L.; Lee, H.; Adluru, N.; Alexander, A.L.; Davidson, R.J.; Pollak, S.D. Persistent homological sparse network approach to detecting white matter abnormality in maltreated children: MRI and DTI multimodal study. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Nagoya, Japan, 22–26 September 2013; pp. 300–307.
42. Chung, M.K.; Hanson, J.L.; Ye, J.; Davidson, R.J.; Pollak, S.D. Persistent homology in sparse regression and its application to brain morphometry. *IEEE Trans. Med. Imaging* **2015**, *34*, 1928–1939. [CrossRef]

43. Benzekry, S.; Tuszynski, J.A.; Rietman, E.A.; Klement, G.L. Design principles for cancer therapy guided by changes in complexity of protein-protein interaction networks. *Biol. Direct* **2015**, *10*, 32. [CrossRef]
44. Huang, W.; Ribeiro, A. Persistent homology lower bounds on high-order network distances. *IEEE Trans. Signal Process.* **2017**, *65*, 319–334. [CrossRef]
45. Zhao, Q.; Wang, Y. Learning metrics for persistence-based summaries and applications for graph classification. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 9855–9866.
46. Chowdhury, S.; Mémoli, F. Persistent homology of directed networks. In Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 6–9 November 2016; pp. 77–81.
47. Chowdhury, S.; Mémoli, F. A functorial Dowker theorem and persistent homology of asymmetric networks. *J. Appl. Comput. Topol.* **2018**, *2*, 115–175. [CrossRef]
48. Horak, D.; Maletić, S.; Rajković, M. Persistent homology of complex networks. *J. Stat. Mech. Theory Exp.* **2009**, *2009*, P03034. [CrossRef]
49. Dey, T.K.; Shi, D.; Wang, Y. Comparing graphs via persistence distortion. Available online: https://arxiv.org/pdf/1503.07414.pdf (accessed on 2 September 2020).
50. Hajij, M.; Wang, B.; Scheidegger, C.E.; Rosen, P. Visual Detection of Structural Changes in Time-Varying Graphs Using Persistent Homology. In Proceedings of the Pacific Visualization Symposium (PacificVis), Kobe, Japan, 1 April 2018; pp. 125–134.
51. Mazza, M.; Cresci, S.; Avvenuti, M.; Quattrociocchi, W.; Tesconi, M. Rtbust: Exploiting temporal patterns for botnet detection on twitter. In Proceedings of the 10th ACM Conference on Web Science, Amsterdam, The Netherlands, 27–30 May 2018; pp. 183–192.
52. McNemar, Q. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* **1947**, *12*, 153–157. [CrossRef] [PubMed]