Similarity-Based Instance Transfer Learning for Botnet Detection

Basil Alothman
De Montfort University, Leicester, UK

Abstract

Botnet Detection has been an active research area over the last few decades. Researchers have been working hard to develop effective techniques to detect Botnets. From reviewing existing approaches, it can be noticed that many of them target specific Botnets and many others try to identify any Botnet activity by analysing network traffic. They achieve this by concatenating existing Botnet datasets to obtain larger datasets, building predictive models, and then employing these models to predict whether network traffic is safe or harmful. The problem with the first approaches is that data is usually scarce and costly to obtain. By using small amounts of data, the quality of predictive models will be questionable. On the other hand, the problem with the second approaches is that it is not always correct to concatenate datasets from different Botnets. Datasets can have different distributions which means they can downgrade the predictive performance of machine learning models. This paper introduces a transfer learning approach that utilises datasets from different but related domains. The idea is instead of concatenating datasets, transfer learning can be used to carefully decide what data to use. The hypothesis is that predictive performance can be improved by using transfer learning across datasets containing network traffic from different Botnets. The approach is compared to a classical open source transfer learning algorithm. Experiments show that the proposed method outperforms this approach and produces higher accuracy. Not only this, but it is also faster which gives it another advantage.

1. Introduction

The dangers of Botnets are becoming more widespread; an example of this is the WannaCry attack that caused many significant institutions in several countries to struggle to perform their services [1]. The research community has been actively trying to develop automatic techniques to identify Botnets in order to stop their harmful activities. This paper introduces a transfer learning method that can be used to enhance the performance of predictive models employed to identify Normal and Malicious traffic. The structure of this paper is as follows: the second section gives an overview of Botnets, section three introduces machine learning and transfer learning, section four is about existing Botnet detection approaches, section five provides an overview of the data that was used in this work, section six explains the methods used in this work, section seven provides detailed experimental setup and discussion of results. The paper then ends with conclusions and future work.

2. Botnet Overview

Before describing the work done in this paper, perhaps it is necessary to provide a brief overview of Botnets.

2.1. What is Botnet

Before delving into the details of the technique developed as part of this work, it is a good idea to provide an overview of Botnets. This section has several sub-sections on what Botnets are, key concepts that one should be familiar with to understand Botnets, how a Botnet attack is performed and Botnet topologies and architectures.

2.1.1. Botnet Definition

There are several definitions of Botnets in the literature. A general definition is:

"A Botnet is a group of networked devices that are used to carry out malicious attacks. Examples of such devices are desktop computers, laptops, smartphones and tablets. These devices, known as hosts, are normally under the remote control of another device known as the Botmaster" [2]

For malicious users, this configuration is appealing because the device(s) carrying out attacks, or malicious activities in general, is not theirs. This is because the communication needed to execute attacks is usually done through Internet Relay Chat (IRC) channels. Using such channels makes it difficult to trace attacks back because commands can be sent via an obfuscating proxy. Not only this, but attackers can also use tools to send commands via multiple hops to add more complexity [3].

It can be noticed that the word Botnet consists of two words; Bot and Net. The first word, Bot, refers to a computer program, a script or application, that executes tasks automatically. This means a Bot can be useful in cases when automation is required. An example is placing online bids such as on eBay. However, the Bot type this work is interested in is the one that is programmed to receive remote commands to perform dangerous actions. The word

Net means that several Bots (it can be thousands or even millions) are run on networked machines to perform tasks in large scale. These machines are identified as BotClients.

Botnets are considered by many as the core of cyber-crime [4]. They are used to gain control of, and use, a large number of connected devices to send commands to perform harmful tasks such as information stealth and industrial espionage [5]. In general, they are amongst the most popular threats that are difficult to mitigate and protect against [4].

One of the most well-known attacks on the internet is the Denial of Service (DoS) attacks. Although Botnets are used to carry out such attacks, they are mostly used for spam attacks to steal sensitive information like login credentials and credit card details. Not only this, but they are also used to execute fraudulent attacks on banks and other organisations [6].

The ways Botnets spread vary. According to [7] they can be injected into remote machines via using social engineering tricks on chatting applications and HTTP based communication tools in general. Another way is the use of advanced double fast-flux service networks and SQL injection attacks [8].

2.1.2. Key Concepts

To understand how Botnets work, it is essential to be familiar with the following concepts which are key in the configuration and functioning of Botnets:

- Network: the first concept to be familiar with is computer networking. Botnets are spread over a large number of devices that communicate with each other and with their Botmaster. This communication facilitates the ability to send/receive commands and updates.
- Machines are Compromised: Normally the BotClients are compromised devices. What this means is that these clients can be used to perform attacks unwillingly and unknowingly. In other words, these devices are exploited and used to participate in Botnets without the knowledge and permission of their real users or owners.
- Remote Control: the compromised devices mentioned in the previous point are customarily controlled remotely by a Botmaster. They receive commands and communicate in a Command and Control configuration (known as the C&C). This allows the exploiter to use some of, or all, the Bots in the Botnet as the attack they are trying to perform requires. This control can be in one of several structures (See section 1.1.4. for more detail).
- Remote Controller: The previous point talked about the compromised devices being remotely controlled. This process is performed by a malicious person (the exploiter). This person usually wishes to execute some illegal activities and harmful attacks. An example attack was the DDoS attack that was launched against Estonia as explained in [9].

2.1.3. Anatomy of a Bot Attack

Botnets follow a systematic way to launch attacks. As shown in Figure 1 (which was obtained from our paper in [38]), the first step is for the Botmaster to infect a victim with a Bot. As mentioned previously, there are several ways to infect a victim. Observe that the number of infected victims can be considerable and, therefore, the attacker can have an army of Bots under his/her control. After the victims are infected, they connect to the C&C server and wait for instructions. This connection can be established using one of the known protocols such as HTTP or IRC. Then, the C&C server sends its commands to the victims which in turn execute the commands and report back the results.

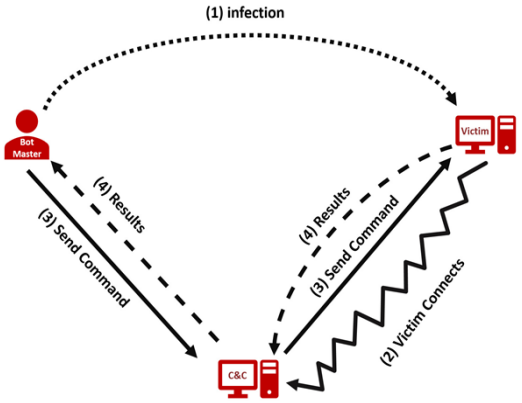

Figure 1: Anatomy of a Bot Attack

2.1.4. Botnet Topologies

Botnets have their own characteristics and components. They can typically be divided into smaller entities such as Botmasters, Bot clients, Bot servers and Bot victims. The Bot master is the attacker/controller of the Botnet. They are mostly the developers of the botnet (i.e. malware management or bot controller). The Bot client is the target device which the botmaster wants to control and use (this is sometimes referred to as the host or node). The Bot Server Command & Control (C&C) is the Command and Control server of the bots which receives commands from the Botmaster through some command and control (C&C) infrastructure to control and give orders to Bot clients. In general, there are three types of Bot server C&C: Centralised (one host), Peer2Peer (one to one) and distributed client (random). Nodes are the Bots which will be used to attack the victims. Botnet Victims, or the Botnet customers, receive the planned attack from the attacker. An example Bot attack is through sending or flooding the target system with a huge

amount of any data to disrupt the system. The Botnet topologies can be summarised as follows:

- Centralised Topology (Star & Distributed cluster): This configuration is like the usual client-server model where a client connects to a server, and the server sends commands and receives reports/results from the client. In such C&C architecture, all bots connect to the Botmaster (see Figure 2). It is worth mentioning here that the Botmaster itself can consist of more than one device (i.e. it can be a group of machines instead of a single machine). These machines are usually responsible for transmitting commands to Bots. This topology offers the advantage of reliable coordination between the Bots and their Botmaster. Also, it speeds up reaction time and it makes status monitoring easy for the Botmaster. On the other hand, the C&C server(s) in this architecture is always a single point of failure. This means once a Botnet is identified, cutting the communication between Bots and their Botmasters effectively means turning off the Botnet. This led to the development of the decentralised configuration discussed in the next point.

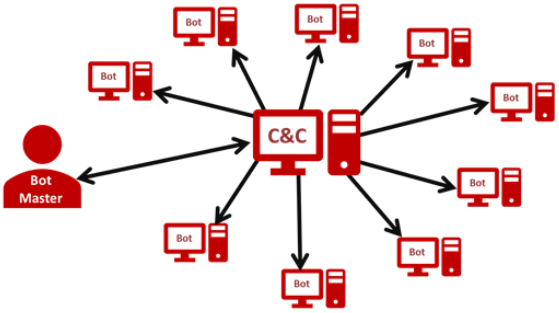

Figure 2. Centralised Topology

- DeCentralised Topology (Peer 2 Peer / Random / No C&C): The Peer-to-Peer topology, or P2P for short, is another Botnet configuration that exists (see Figure 3). In this architecture, as its name suggests, Bots can have a control role in addition to their usual role. This means that it is no longer necessary to communicate with the Botmaster which gives the advantage of being resilient to failure. Therefore, identifying Bots does not necessarily mean turning off the entire Botnet as the case with the centralised topology. Also, the P2P architecture offers more flexibility and robustness especially when the number of Bots is large.

There are a few techniques that are used to construct a P2P Botnet. The process usually has two steps. In the first step, peer candidates need to be selected. And in the second, actions need to be implemented so the selected candidates become members of the Botnet.

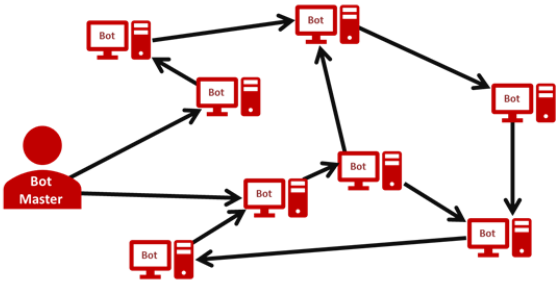

Figure 3. Decentralised Topology

- The Hybrid Model C&C: In this architecture, functionalities from both centralised and decentralised botnets are used. In general, Bots that are part of a hybrid P2P Botnet can be either server Bots or client Bots. The server Bots exhibit the behaviour of both clients and servers whereas the client Bots are configured to act as clients only.
- Random model C&C: In this architecture the Bot does not continuously communicate with the Botmaster or other Bots; rather it waits for the Botmaster to make connection attempts. For an attack to be performed, the Botmaster tries to connect to idle Bots, if it finds any, it sends the commands to execute attacks. This model is not difficult to implement, and it is not easy to identify and interrupt because the Bot does not initiate the communication between the Bot and Botmaster. On the other hand, coordination issues can arise if the number of Bots is large as the Botmaster has to go through an extensive list of Bots. There are not any real Botnets that use this model, it is only theoretical as it was suggested by Cooke et al. [10].

3. Machine Learning and Transfer Learning

This paper presents a machine learning approach for the automatic detection of Botnets. Specifically, the technique is a transfer learning technique that attempts to boost small Botnet datasets using existing larger datasets for data that belongs to other Botnets. In this section a brief overview of machine learning and transfer learning is given.

3.1. What is Machine Learning

Machine learning is mainly about developing and applying algorithms and techniques to automatically explain the past and predict the future through data analysis. This field combines several other fields that include statistics, data science, artificial intelligence, and database technologies. Hence, it is a multidisciplinary field. It is worth mentioning here that data is a key component. Without data nothing practical can be done and only theoretical concepts and ideas can be developed at most.

On the one hand, explaining the past is done via data analysis and exploration. Here statistical and visualisation techniques are usually used to inspect the data. This is performed to highlight important relationships, patterns, trends, or aspects that exist in the data so further analysis can be performed. On the other hand, predicting the future is performed via modelling. In predictive modelling, a model is created from the data to make a prediction. The prediction can be made for one or more outcomes. In this work the focus is on using only a single outcome. If such an outcome is categorical, then the process is called classification. If it is numerical, then the process is called regression. If the predictive process is about grouping similar instances of the data into groups of similar instances, then it is called clustering. There are more processes in machine learning and data mining; observe that this work is focused on classification.

Making automatic predictions is highly regarded nowadays. The historical data that is already stored, and the large amounts of data that is generated daily, can help businesses (via machine learning and data mining) derive valuable insights and knowledge. This extracted knowledge can assist in decision making and future planning to improve efficiency and maximise gains and profits.

3.2. Example Machine Learning Algorithms

The following is a high-level overview of some of the most common traditional machine learning algorithms. The reader is encouraged to review the cited sources for a more detailed explanation:

- NaiveBayes [11]: The Naive Bayes classifier is based on Bayes theorem with independence assumptions between input variables (predictors). Suppose x was the input variables and c was the class, Bayes theorem introduces a method to calculate the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. This classifier assumes that the effect of the value of an input variable (x) on a given class (c) is not dependent on the values of other input variables (i.e. they are independent). This assumption is known as class conditional independence.
- K-Nearest Neighbours (KNN) [12]: K-Nearest Neighbours (KNN) is an algorithm that stores all available examples (i.e. instances) and classifies new examples based on a similarity measure (e.g. distance functions). In more detail, a majority vote of neighbours is used to classify new examples. This is achieved by the choosing the most common class amongst the K-Nearest neighbours.
- Decision Trees [13]: This is a variation of the well-known C4.5 algorithm. The way it works is by building classification or regression models in the form of a tree structure. This is performed by breaking down a dataset into smaller and smaller subsets recursively, while incrementally developing an associated decision tree at the same time. This finally results in a tree that contains decision and leaf nodes. Decision nodes have branches, whereas Leaf nodes contain decisions (e.g. classification).
- RandomForest [14]: This algorithm works by building many decision trees at training time and using them to vote for the class of a new example. To construct each tree, the training data is obtained by randomly sampling both the examples and input variables (with replacement).
- Support Vector Machine (SVM) [15]: This algorithm carries out classification by finding the hyperplane that sees the maximum possible margin between two classes.

3.3. What is Transfer Learning

We as human beings can utilise past learning experiences when we are faced with new tasks. Our level of mastering the new task depends on how much it is related to the past task. In machine learning, the sub-field that attempts to apply this experience, or knowledge, transfer is known as Transfer Learning.

As explained in [16], in traditional machine learning algorithms, tasks are dealt with individually, meaning if there are several tasks then each one is learnt separately. On the other hand, transfer learning attempts to learn one or more tasks (known as source tasks) and use the knowledge learnt to enhance learning in another task (known as the target task). The target and source tasks are usually related.

Transfer learning is typically employed when there is little, or limited, amounts of labelled data in one task (the target task), and plenty of data in one or more other related tasks (the source tasks). The idea here is that using only the target data can lead to obtaining models with poor performance since there is not sufficient data. Whereas, by transferring knowledge from the source tasks the model quality can be improved.

There are three key research issues in transfer learning [17]. The first issue is what to transfer, which is concerned with the parts of knowledge that can be transferred between tasks. This is because not all knowledge is common between different tasks (i.e. some knowledge can be specific for specific tasks). The second issue is how to transfer. This is related to whether to transfer the knowledge as is or to apply some form of modification such as assigning weights. The third issue is when to transfer. This is an important aspect of transfer learning as in may cases the knowledge transfer ends in a negative transfer. That is, instead of improving learning in the target task, the knowledge transferred deteriorates the process.

Transfer learning techniques can be categorised into three sub-settings. Inductive transfer learning,

transductive transfer learning and unsupervised transfer learning. This work focuses on Inductive transfer learning, the reader is referred to the survey in [17] for more details on the other two categories. This work is focused on Inductive transfer learning because the problem it is trying to address involves data that comes from the same domain (i.e. network traffic) and all datasets that have been obtained have the same feature space (i.e. they have the same features).

3.4. Inductive transfer learning

In transfer learning in general, we can have two different tasks (i.e. the source and target tasks are not the same) coming either from the same domain or two different domains. In inductive transfer learning, all we care about is having different source and target tasks, it is not critical if the source and target domains are different or the same. Inductive transfer learning can be performed in more than one way:

- The instance-transfer approach: This approach focuses on transferring instances, or examples, from source tasks to the target task. Its main idea is not to transfer all data from the source task directly, but rather, it is to try to use relevant instances from the source task, along with data in the target task, in building a model for the target task. An example approach is TransferBoost that can be found in [18]. TransferBoost is based on the classical AdaBoost algorithm. It works when the source and target tasks have the same set of features, but different data distributions. Also, TransferBoost assumes that some of the data in the source tasks can be useful (i.e. leads to positive transfer) and some can be harmful (i.e. leads to negative transfer). The idea is to assign weights to data from the source task in such a way that useful data can have more effect than harmful data. The developer has made the java implementation of this approach publicly available.
- Feature representation transfer: In this approach, attempts are made to find feature representations that reduce classification error. This task is also known as common feature learning [19]. Methods for feature representation transfer can be supervised [20], [21] or unsupervised [22].
- Parameter-transfer: In this approach, it is assumed that models created for individual related tasks should have some parameters in common. Some of the existing approaches transfer parameters of Support Vector Machines [23] and priors Gaussian Processes [24].
- Relational-knowledge transfer: Methods are falling under this category focus on transfer learning in relational domains. Approaches employ techniques from statistical relational learning to transfer relationships from the source to target domains. An example method is reported in [25] where relational knowledge is transferred across relational domains by using Markov Logic Networks [26].

4. Existing Transfer Learning techniques for Botnet Detection

The idea of Bots and Botnets started with using computer programs to provide support in Internet Relay Chat (IRC) rooms. This dates back to the early 1990's. Therefore, although it was possible, harmful behaviour was not primarily intended. However, after the successful utilisation of such programs to carry out activities such as offering simple games to IRC users and interpreting or executing simple commands, the first malicious IRC Bot was Eggdrop which came to the scene in 1993 [27]. Since then until this time, several Botnets with various degrees of harm have appeared.

There is a large number of efforts reported in the literature to automatically detect and identify Botnets. The reader is referred to the survey in [28]. Although none of the reported methods in this survey uses transfer learning, many of them use traditional machine learning techniques such SVM, NiaveBayes and others. According to the survey, existing Botnet detection techniques can be categorised in multiple ways. For example, some methods are *host-based*; this means these techniques analyse the behaviour of the host machine (the Bot) as it is where the Bot is running. If a Bot is running, it will be carrying out its predefined activities which are not the same as activities performed by normal processes. The advantage of using host-based detection methods is their effectiveness because the analysis is done on every machine. On the other hand, their disadvantage is that they become costly and time-consuming if the number of machines to be analysed is large.

Some other methods can be considered *network-based* because they attempt to identify malicious behaviour by analysing network traffic. They monitor and scrutinise several aspects such as the patterns, behaviour and response time. It is worth mentioning here that although many network-based techniques try to be generic and involve multiple protocols and architectures, some were created for specific protocols. According to the survey in [28], these monitoring techniques can be active or passive.

Botnet detection is an ideal area for transfer learning. It is not surprising that for many Botnets, there is plenty of data and for other Botnets, data is scarce and costly to obtain. Models built with very small datasets are usually poor and one way to improve performance (i.e. prediction accuracy) is by employing Transfer Learning. For example, one of the most recent Botnets is WannaCry. How do we obtain enough data to build a highly accurate model? The idea here is that if there is little WannaCry data, data from other Botnets can be exploited to build a better model.

Although several machine learning techniques were used for Botnet detection, transfer learning is still unexplored well. This is despite its obvious suitability. To the best of the author's knowledge, this work is only the second attempt to try Transfer Learning for Botnet Detection. The first attempt was introduced in [29] where feature transfer learning was used, as opposed to the method proposed in this work which is instance transfer. The technique is based on projecting the source and target data into a common latent shared feature space and then using this new feature space for making predictions. The technique works in such a way that it attempts to preserve the distribution of the data. Although the results reported by the author seem to be reasonable, there is no freely available tool or code to use for comparison. As this technique is iterative, it is computationally heavy. The approach proposed in this work is different as it performs instance transfer by doing only one pass over the target data.

5. The Data

To run experiments and evaluate the method introduced in this work, the data that was used in [30] was downloaded. The data is in Packet Capture (PCAP) format. Details of this data can found in [31]. Experiments were only run using the Testing Dataset because it contains more Botnets than the Training Dataset. Because the data is in PCAP format, it needed to be transformed it into a format that machine learning platforms such as WEKA [32] or SciKit-Learn [33] understand (i.e. into Character Separated Values, or CSV, format). Therefore, FlowMeter [34] was downloaded and used. It reads in a directory that contains one or more PCAP files and transforms them into CSV files. It generates several attributes (features) such as Source Port, Destination Port, Protocol, Flow Duration, Flow Bytes per second and Flow Packets per second. The original number of features generated by FlowMeter is 26 and the total number of instances obtained is ~309000. After this step several preprocessing steps were carried out. A label was assigned to each instance in the dataset according to the guidelines provided in [31] and missing values were imputed. Then one hot encoding was applied to the fields: Source Port, Destination Port and Protocol to transform them from categorical to numerical. Also, highly correlated features were removed. This was followed by outlier detection and removal, splitting the large dataset into smaller sub-datasets according to Label. This means a separate dataset was created for each Botnet and one for the Normal traffic. Furthermore, non-overlapping random samples were chosen from the Normal data and appended to Botnet datasets so that there are positive and negative examples in each dataset. The author of this paper has written a paper that provides a full description of these preprocessing steps and submitted to the Cyber Incident 2018 conference (still awaiting decision).

It was mentioned previously that transfer learning is going to be used for Botnet detection, before explaining the approaches and experiments, it may be a good idea to inspect the different Botnet datasets. To perform data exploration, a common technique is Partial Least Squares [35] (PLS). PLS is a supervised technique that uses both the data X and the class variable Y. It is used to model the covariance structures in these two matrices (i.e. X and Y) and discover the fundamental relations between them. The idea behind it is to explain the highest multidimensional variance in Y by finding the corresponding multidimensional direction in X. It is noteworthy that before applying techniques such as PLS, two important steps that are usually applied are data scaling and normalisation.

Figure 4 shows the result of using PLS with data from Botnets TBot, Zero access and Zeus. It can be seen in the figure that there is a clear separation between instances that belong to different Botnet types. Instances belonging to the same Botnet type tend to form a cluster and this solidifies the argument of this work.

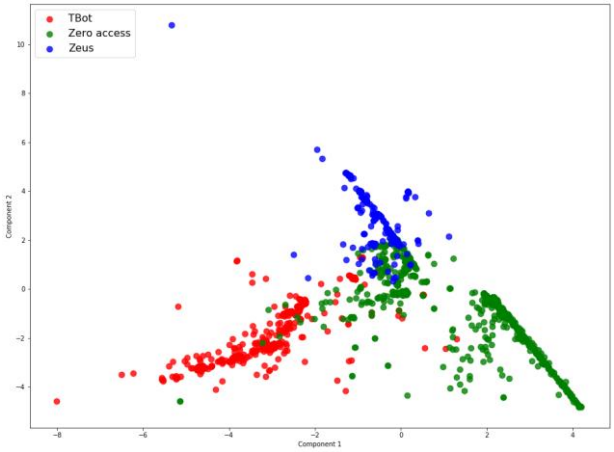

Figure 4. PLS

6. Methods

In the following subsections provide an overview of the open source transfer learning algorithm that was used. They also explain the proposed transfer learning method in detail.

6.1. The TransferBoost Algorithm

The TransferBoost algorithm [18] is a transfer learning algorithm that is based on the classical AdaBoost algorithm. It is an instance transfer learning algorithm and it works by trying to boost target data by transferring instances from the source data and assigning weights to these instances. It examines the transferability of instances by checking

the change in performance on the target task when, and when not, transferring instances. The weight assignment in TransferBoost is done in such a way that higher weights are assigned to instances that show positive transferability and lower weights are assigned to instances that show negative transferability. It iteratively updates weights so that, when it finishes training, instances that exhibit positive transfer can have high weights, and therefore they have more influence, and instances that exhibit negative transfer can have very low, or zero, weights, and therefore they have little to no influence.

The TransferBoost algorithm developer has made the implementation publicly available. The implementation is in Java and it is based on WEKA. It was downloaded and used in the experiments as explained in more detail in Section 7.

6.2. The Similarity-Based Instance Transfer Algorithm

In this section the proposed transfer learning method is going to be explained. As it was mentioned previously, the method is based on instance transfer. As illustrated in the pseudo-code shown in Algorithm 1, the algorithm receives as input one target dataset and one or more source datasets. It loops through the instances of each source dataset and checks how similar these instances are to the instances of the target dataset. As the data is numerical, several similarity methods are used to check how similar the instances are. An instance is selected for transfer from the source dataset to the target dataset if it satisfies the conditions. These conditions are based on using empirically determined threshold values for each type of similarity that was used. The idea is, because the source datasets are related to the target dataset, they are likely to contain similar instances.

Algorithm 1: The Proposed Transfer Learning Method

Algorithm Similarity-Based Instance Transfer (SBIT):

Input: Source Datasets $S_1, S_2, \ldots S_n$
Target Dataset T
Selected = [ ]
$thr_1, thr_2, \ldots thr_k$

1. For S in $[S_1, S_2, \ldots, S_n]$:
2. For I_s in S:
3. For I_T in T:
4. $Sim_1 = ComputeSimilarity_1(I_s, I_T)$
$Sim_2 = ComputeSimilarity_2(I_s, I_T)$
...
$Sim_k = ComputeSimilarity_k(I_s, I_T)$
5. If $Sim_1 > thr_1$ & $Sim_2 > thr_2 \ldots$ & $Sim_k > thr_k$:
6. Add I_s to Selected
7. T_{NEW} = Merge(T,Selected)

Output: T_{NEW}

In more detail, the source datasets are scanned one by one and an attempt is made to find any similar instances in these datasets to any of the instances of the target dataset. In the remaining parts of this paper, we are going to refer to the developed method as SBIT (for Similarity-Based Instance Transfer).

6.3. Similarities

Before discussing the types of similarities that were used, let us explain why instance similarity is used. The idea is that similar instances tend to belong to similar classes (i.e. they have similar behaviour). The author believes that this is a reasonable rule-of-thumb in the absence of more detailed knowledge. Also, this is going to be examined experimentally and its worthiness will be investigated.

6.3.1 What is Similarity

The definition of similarity can be subjective; therefore, it is essential to have a quantitative approach for estimating the degree of resemblance.

Let us assume that the similarity of two entities is measured as a real number S. Therefore, it is common to make sure that the value of S is:

$$0 <= S <= 1$$

This is interpreted as a value of S = 0 means there is no similarity at all between the two entities, whereas a value of S = 1 means the two entities are identical (i.e. They are the same). The degree of similarity increases as S approaches 1, and decreases as S approaches 0.

6.3.2 How to Measure the Similarity of Instances?

To measure the similarity of two instances, one approach is to consider the feature values of each instance as a vector (only feature values without the class label). Fortunately, the feature values in the network traffic data used in this work are all numeric.

For example, a feature vector representing one instance should look like:

f_1	f_2	….	f_n

Where f_1, f_2, …, f_n are the feature values for the first feature, second feature and so on.

There are many different formulae for computing the similarity between two numerical vectors of the same length. An excellent resource is a book that can be found in [36].

6.3.3 The Similarity Types used in this Work

Now let us assume there are two real-value vectors *X* and *Y* such that:

$X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_n]$

To compute similarities between instances, one only needs to plug these vectors in a suitable similarity formula. The similarity types used in this work are listed in Table 1.

Table 1. Similarity Measure and their Formulae

Similarity	**Formula**
Tanimoto Similarity (X,Y)	$sum(x_i,y_i)/(sum(x_i * x_i) + sum(y_i * y_i) + sum(x_i,y_i))$
Ellenberg Similarity (X,Y)	$Sum((x_i, y_i) * 1 \text{ (when } x_i*y_i != 0)) / Sum((x_i+y_i) * (1 + 1\ x_i*y_i=0))$
Gleason Similarity (X,Y)	$Sum((x_i, y_i) * 1 \text{ (when } x_i*y_i != 0)) / Sum(x_i+y_i)$
Ruzicka Similarity (X,Y)	$Sum(min(x_i, y_i)) / Sum(max(x_i+y_i))$
BrayCurtis Similarity (X,Y)	$(2/(n * (Avg(X)+Avg(Y)))) * Sum(min(x_i,y_i))$

7. Experiments and Discussion of Results

To test the SBIT method and compare it to existing methods, the Botnet dataset mentioned in Section 5 was used. It was split into smaller Sub-datasets according to the class label which resulted in one separate dataset for each Botnet as well as one dataset that contains the Normal traffic. Non-overlapping samples from the Normal dataset were taken and added to the Botnet datasets so that there are positive and negative examples in each dataset (i.e. each dataset now contains Botnet and Normal data). It was ensured that the number of positive and negative examples in each dataset is approximately the same to avoid class imbalance.

As explained in Section 3.3, transfer learning requires Target and Source datasets. To carry out experiments, datasets which are relatively small were selected and used as Target datasets. The number of Target datasets is five, Table 3 shows details of each source and target datasets. Also, three datasets were selected to be used as Source datasets (this means source datasets are the same across all experiments). To evaluate performance, the Target datasets were split into two datasets: Target and Test.

Table 2. Dataset Details

Dataset Name	Usage	No of Instances
Menti	Source Dataset	489
Mulro	Source Dataset	526
Neris	Source Dataset	501
Sogou-Target	Target Dataset	10
Sogou-Test	Evaluation Dataset	44
TBot-Target	Target Dataset	10
TBot-Test	Evaluation Dataset	231
RBot-Target	Target Dataset	10
RBot-Test	Evaluation Dataset	13
Zeus-Target	Target Dataset	10
Zeus-Test	Evaluation Dataset	165
Smoke_bot-Target	Target Dataset	10
Smoke_bot-Test	Evaluation Dataset	32

With the previous setup, RandomForest, TransferBoost and the SBIT method were. For RandomForest, it was trained using only the Target datasets one at a time. For TransferBoost and the SBIT method, source and target datasets are required to perform training. The source datasets were fixed for both as mentioned previously. The Target dataset was changed using the Target datasets that were selected. To evaluate, the accuracy of each model was computed using the corresponding test dataset. The results are illustrated in Table3.

Table 3. The accuracy of each Method using Different Target Datasets

Target Dataset	RandomForest	TransferBoost	SBIT
RBot	83.33	83.33	**86.58**
Smoke bot	46.87	50.00	**63.75**
Sogou	52.27	59.10	**77.27**
TBot	63.62	**71.42**	69.51
Zeus	69.87	70.51	**76.66**

Because now there are five different Target datasets (and their corresponding Test datasets), it can be seen from Table 3 that the SBIT method outperforms RandomForest and TransferBoost in 80% of the cases (four out of five). When experimenting with the Smoke bot data, the SBIT method produces > 10% increase in accuracy. Also, when using Sogou data, it can be noticed that the SBIT method's accuracy is > 17% higher than that of TransferBoost. However, TransferBoost performs better than the SBIT method when using TBot data. In addition to this, it can be seen that RandomForest was the worst performer in all experiments. It is also worth mentioning that the performance of traditional learners such as RandomForest, NaiveBayes, SVM and others was evaluated in a separate experiment. RandomForest was in general the most accurate amongst them and therefore it was selected to be used it in the experiments.

Another interesting comparison aspect is the run time of the SBIT algorithm against TransferBoost. Since the SBIT algorithm makes a single iteration over the Source data (as opposed to the iterative nature of TransferBoost), it is expected that the SBIT method is going to be faster. This was confirmed when run times of the previous experiments were computed as illustrated in Figure 5.

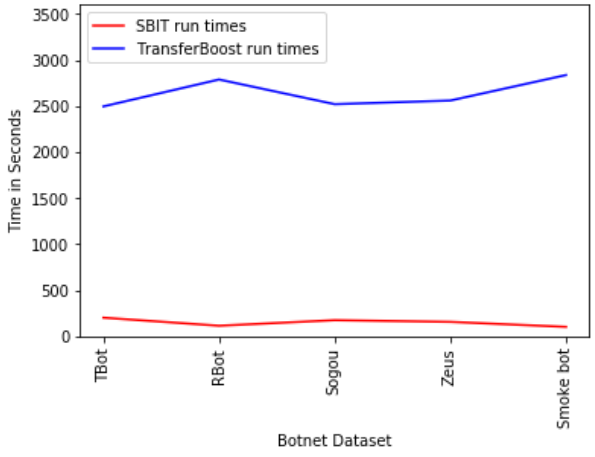

Figure 5. Run Times of the Two Algorithms

8. Conclusions and Future Work

This paper has introduced a simple yet powerful method for transfer learning. This method is an instance transfer method that is based on the similarity between instances in the Source data and instances in the Target data. The method computes more than one similarity measure to make sure as much information is captured as possible. Experimental results show that this method outperforms, in general, a classical instance transfer learning algorithm, namely the TransferBoost algorithm. Not only this, but this method is also much faster which gives it another advantage.

There is already a plan to extend this algorithm in more than one direction in the near future. For example, it is intended to use an optimisation approach, such as Genetic Algorithms, to find the optimal threshold used for similarity. Another idea that is planned to be investigated is to assign weights to the instances transferred from the Source data to the Target data. In addition, it is planned to use the SBIT method in other domains to check how useful it is. For example, it would be interesting to use it to perform transfer learning in the text mining domain as well as in bioinformatics. This will depend on the amount of available data, although enough freely available text mining data exists such as the one available at [37].

9. References

[1] C. K. Cara McGoogan, James Titcomb, "What is WannaCry and how does ransomware work?," 2017. [Online]. Available: https://www.telegraph.co.uk/technology/0/ransomware-does-work/. [Accessed: 27-Feb-2018].

[2] Fariba Haddadi, Jillian Morgan, Eduardo Gomes Filho, A. Nur Zincir-Heywood, (2014). Botnet Behaviour Analysis using IP Flows With HTTP filters using classifiers. In 28th International Conference on Advanced Information Networking and Applications Workshops. Victoria, BC , 13-16 May 2014 . : IEEE. 7-12

[3] Craig Schiller and Jim Binkley. 2007. Botnets: The Killer Web Applications. Syngress Publishing.

[4] Cao Lijun , Wu Xianghua , (2012). Analysis and Design of Botnet Detection System. In International Conference on Computer Science and Service System: IEEE, Pages:947-950.

[5] LI Ting, WANG Huai-bin, WANG Chun-dong, 2009. Botnet Detection Based on Analysis of Mail Flow, IEEE.

[6] Oleg Savenko, Sergii Lysenko, Andrii Kryshchuk, Yuriy Klots, (2014). Botnet Detection Technique for Corporate Area Network. In The 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. Berlin, Germany, 12-14 September 2013. Berlin, Germany: IEEE. 363-368.

[7] Borgaonkar, Ravishankar, 2010. An Analysis of the Asprox Botnet. Fourth International Conference on Emerging Security Information, Systems and Technologies, IEEE, , 148-153.

[8] Aditya K. Sood, Sherali Zeadally, Richard J. Enbody, "An Empirical Study of HTTP-based Financial Botnets", IEEE Transactions on Dependable and Secure Computing vol. 13 no. 2, p. 236-251, , 2016

[9] Grant, R. (2007). Victory in cyberspace, an Air Force Association special report. Retrieved from http://www.afa.org

[10] E. Cooke, F. Jahanian, D. McPherson, The zombie roundup: understanding, detecting, and disrupting botnets, in: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop, USENIX Association, Berkeley, CA, USA, 2005, p. 6.

[11] WIKIPEDIA, S.. Classification Algorithms: Artificial Neural Network, Naive Bayes Classifier, Support Vector MacHine, Boosting, Linear Classifier, Case-Based Reasonin. University-Press Org, 2013.
[12] Larose, Daniel T. , 2014. Discovering Knowledge in Data: An Introduction to Data Mining. 2nd ed.: WILEY.
[13] Lior Rokach, 2014. Data Mining With Decision Trees : Theory and Applications (2nd Edition) (Series in Machine Perception and Artificial Intelligence) (Series in Machine Perception and Artificial Intelligence). 2 Edition. World Scientific Publishing Company.
[14] Sumeet Dua, 2011. Data Mining and Machine Learning in Cybersecurity. 1 Edition. Auerbach Publications.
[15] Ingo Steinwart, 2008. Support Vector Machines (Information Science and Statistics). 2008 Edition. Springer.
[16] L. Torrey and J. Shavlik, "Transfer Learning," Handb. Res. Mach. Learn. Appl. IGI Glob., 2009.
[17] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, 2010.
[18] Eaton, Eric, 2011. Selective Transfer Between Learning Tasks Using Task-Based Boosting. Proceedings of the National Conference on Artificial Intelligence, desJardins, Marie
[19] A. Argyriou and M. Pontil, "Multi-Task Feature Learning."
[20] A. Argyriou, C. Micchelli, M. Pontil, and Y. Ying, "A Spectral Regularization Framework for Multi-Task Structure Learning," Adv. Neural …, pp. 1–8, 2008.
[21] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller, "Learning a meta-level prior for feature relevance from multiple related tasks," Proc. 24th Int. Conf. Mach. Learn. - ICML '07, pp. 489–496, 2007.
[22] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught Learning : Transfer Learning from Unlabeled Data," Proc. 24th Int. Conf. Mach. Learn., pp. 759–766, 2007.
[23] T. Evgeniou and M. Pontil, "Regularized Multi – Task Learning," Proc. tenth ACM SIGKDD …, pp. 109–117, 2004.
[24] N. D. Lawrence and J. C. Platt, "Learning to learn with the informative vector machine," Twenty-first Int. Conf. Mach. Learn. - ICML '04, p. 65, 2004.
[25] L. Mihalkova, T. N. Huynh, and R. J. Mooney, "Mapping and Revising Markov Logic Networks for Transfer Learning," Proc. Twenty-Second Conf. Artif. Intell., no. July, pp. 608–614, 2007.
[26] M. Richardson and P. Domingos, "Markov logic networks," Mach. Learn., vol. 62, no. 1–2, pp. 107–136, 2006.
[27] EggHeads, EggHeads.org - Eggdrop Development, 1993 <www.eggheads.org>
[28] SéRgio S. C. Silva, Rodrigo M. P. Silva, Raquel C. G. Pinto, and Ronaldo M. Salles. 2013. Botnets: A survey. Comput. Netw. 57, 2 (February 2013), 378-403. DOI=http://dx.doi.org/10.1016/j.comnet.2012.07.021
[29] Sachin Shetty 2017. Transfer Learning for Malware Detection. Online Presentation Available at [https://www.odu.edu/~sshetty]. Accessed on 20 Feb 2018
[30] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, Ali A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security, Volume 31, Issue 3, May 2012, Pages 357-374, ISSN 0167-4048, 10.1016/j.cose.2011.12.012.
[31] The ISCX Botnet Dataset. Available at [http://www.unb.ca/cic/datasets/ids.html]. Accessed on 20 Feb 2018
[32] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
[33] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[34] Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy(ICISSP 2016) , pages 407-414, Rome , Italy
[35] Boulesteix, Anne-Laure and Strimmer, Korbinian 2007. Partial least squares: a versatile tool for the analysis of high-dimensional genomic data. Brief Bioinform. pages 32-44, Volume 8
[36] Michel Marie Deza, Elena Deza. 2014. Book title: Encyclopedia of Distances Hardcover. ISBN-10: 3662443414. Publisher: Springer; 3rd ed. 2014 edition
[37] Multi-Domain Sentiment Dataset. 2018. Multi-Domain Sentiment Dataset. [ONLINE] Available at: http://www.cs.jhu.edu/~mdredze/datasets/sentiment/. [Accessed 27 February 2018].
[38] B. Alothman, P. Rattadilok, 2017. Android Botnet Detection: An Integrated Source Code Mining Approach. In the 12th International Conference for Internet Technology and Secured Transactions (ICITST-2017). University of Cambridge, UK, lnfonomics Society. 111-115.

