Class Balanced Similarity-Based Instance Transfer Learning for Botnet Family Classification

Basil Alothman[1], Helge Janicke[2], and Suleiman Y. Yerima[3]

De Montfort University, Leicester LE1 9BH, UK.
[1] P14029266@my365.dmu.ac.uk
[2] heljanic@dmu.ac.uk
[3] syerima@dmu.ac.uk
http://www.dmu.ac.uk/technology
Faculty of Technology, De Montfort University, Leicester, UK

Abstract. The use of Transfer Learning algorithms for enhancing the performance of machine learning algorithms has gained attention over the last decade. In this paper we introduce an extension and evaluation of our novel approach Similarity Based Instance Transfer Learning (SBIT). The extended version is denoted Class Balanced SBIT (or CB-SBIT for short) because it ensures the dataset resulting after instance transfer does not contain class imbalance. We compare the performance of CB-SBIT against the original SBIT algorithm. In addition, we compare its performance against that of the classical Synthetic Minority Over-sampling Technique (SMOTE) using network traffic data. We also compare the performance of CB-SBIT against the performance of the open source transfer learning algorithm TransferBoost using text data. Our results show that CB-SBIT outperforms the original SBIT and SMOTE using varying sizes of network traffic data but falls short when compared to TransferBoost using text data.

Keywords: Similarity-Based Transfer Learning · Botnet Detection · SMOTE · TransferBoost

1 Introduction

Transfer learning is an active research area in machine learning [14]. Common machine learning algorithms deal with tasks individually [17], meaning several tasks can only be learnt separately. Transfer learning attempts to learn from 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 must be related in one way or another.

Transfer learning is typically employed when there is limited amount of labelled data in one task (the target task), and sufficient data in another related task (the source task). The idea here is that using only the target data can lead to obtaining models with poor performance since there is insufficient data.

Whereas, by transferring knowledge from the source task(s), model quality can be improved.

Transfer learning in network traffic classification was introduced in [19] where feature transfer learning was used, as opposed to our method which is based on 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 model building and making predictions. The technique attempts to preserve the distribution of the data. Although the reported results seem to be reasonably good, there is no freely available tool or code to use for comparison. As this technique is iterative, it can be computationally heavy. The approach we propose in this work is more efficient in terms of speed as it performs instance transfer by performing only one pass over the target as well as source data.

A recent work that applies transfer learning for classification of network traffic can be found in [16]. This work does not propose a new transfer learning method, rather, it only evaluates the performance of an existing open source transfer learning algorithm called TrAdaBoost [5]. Although the results show performance improvement when compared against the base classifier without transfer (referred to as NoTL in the publication), it is noteworthy to mention that TrAdaBoost was extended and enhanced by the introduction of TransferBoost [7] - which is the algorithm that we compare our results against as explained in more detail in [2].

Instance transfer learning has been applied in multiple areas. For example, the recent work in [13] reports an attempt that employs Multiple Instance Learning (MIL) in text classification. This is a two stage method where, in the first stage, the algorithm decides whether the source and target tasks are similar enough to perform transfer which leads to the second stage where transfer is performed.

In this paper, we extend our novel algorithm Similarity Based Instance Transfer Learning (SBIT) and evaluate the performance of the extended version. More detailed explanation of how SBIT works and an evaluation of its performance can be found in our previous work in [2]. We will refer to the extension presented in this work as Class Balanced SBIT (or CB-SBIT) because it ensures that the dataset resulting after instance transfer is class balanced (see Section 2 for more details).

The main contributions of this paper are as follows: 1) We introduce an extension to our previous Similarity Based Instance Transfer approach [2] that guarantess class balance in the resulting dataset (avoiding over-fitting) 2) We compare the performance of the extended version of our algorithm against the original version 3) We compare the performance of the extended version of our algorithm against two classical and well known algorithms 4) We show where our algorithm works well and where it does not.

The remainder of this paper is organised as follows: Section 2 introduces the SBIT algorithm, highlights one of its current shortcomings, explains the class imbalance problem and provides an overview of the new algorithm CB-SBIT. Section 3 provides a detailed explanation of experimental setups and results for comparing the performance of CB-SBIT against SBIT, SMOTE and TransferBoost using two different types of data. The paper then ends with the conclusions and future work in Section 4.

2 Similarity Based Instance Transfer

This section provides a short overview of the original SBIT algorithm [2], class imbalance problem and then introduces the extended version of SBIT (i.e. the CB-SBIT).

2.1 The SBIT Algorithm and its Class Imbalance Problem

The SBIT algorithm is an instance transfer algorithm that scans source datasets one at a time and tries to find similar instances in these datasets to instances in the target dataset. If any similar instance is found, it is transferred to the target dataset which is used later to build a learning model. The pseudo-code of SBIT is provided in Algorithm 1. It is important to bear in mind that SBIT assumes the input target dataset is class balanced (i.e. it contains approximately an equal percentage of classes).

Algorithm 1: The Proposed Transfer Learning Method Algorithm: Similarity-Based Instance Transfer (SBIT)

```
Input  : Source Datasets $S_1, S_2, \dots S_n$
Input  : Target Dataset $T$
Input  : Selected = [ ]
Input  : $thr_1, thr_2, \dots thr_k$
Output: New Dataset that is the result of $Concatenate(T, Selected)$
1  for $S \in [S_1, S_2 \dots S_n]$: do
2      for $I_s \in S$: do
3          for $I_T \in T$: do
4              $Sim_1 = ComputeSimilarity1(I_s, I_T)$;
5              $Sim_2 = ComputeSimilarity2(I_s, I_T)$;
6              $\dots$;
7              $Sim_k = ComputeSimilarity_k(I_s, I_T)$;
8              if $Sim_1 > thr_1 \& Sim_2 > thr_2 \dots \& Sim_k > thr_k$ then
9                  Add $I_s$ to Selected ;
10 $T_{NEW} = Concatenate(T, Selected)$;
11 Return $T_{NEW}$;
```

Careful inspection of Algorithm 1 reveals that SBIT copies an instance from the source data to the target data as soon as it satisfies the similarity criteria (lines 8 and 9). It performs this step without paying attention to the class of that instance. This means it is very possible for instances transferred by SBIT to belong to one class only (or at least for the majority of them to belong to the same class) which leads to creating a new target dataset that is class imbalanced.

2.2 The Class Imbalance Problem

One of the main reasons that cause overfitting is class imbalance [10]. Class imbalance refers to the problem when a classification dataset contains more than one class and number of instances in each class is not approximately the same. For example, there might be a two-class classification dataset that contains 100 instances where the number of instances for one of the classes is 90 and for the other is 10. This dataset is said to be *imbalanced* as the ratio of first class to second class instances is 90:10 (or 9:1). One might train a model that yields 90% accuracy but in reality it could be that the model is predicting the same class for the vast majority of testing data. It is worth mentioning here that evaluation methods such as f1-score, area under the curve, or precision/recall rates provide provide better insight regarding classifier performance when using imbalanced datasets. However, our work focuses on ensuring class balance so an easy to interpret metric such as accuracy can be used.

There are several ways to combat class imbalance [3]. One of these methods is to down sample the majority class (this is sometimes referred to as under sampling). In other words, to randomly select a subset of the instances that belong to the majority class so that the number of instances in each class in the resulting dataset is approximately the same. Another method is to over sample the minority class; which means to randomly duplicate instances from the minority class so the dataset becomes class balanced.

One common technique that falls under this category is the SMOTE algorithm (or the Synthetic Minority Over-sampling Technique [4]) which generates synthetic instances that belong to the minority class rather than generating duplicates.

2.3 The Class Balanced SBIT Algorithm (CB-SBIT)

To avoid class imbalance, the SBIT [2] algorithm discussed in Section 2.1 can be modified to ensure the resulting dataset is class balanced.

Recall SBIT assumes that the target dataset is class balanced, the modified version of SBIT makes sure that the new dataset (resulting after selecting instances from source datasets) remains class balanced by using a strict criterion as illustrated in Algorithm 2. This can be achieved in more than one way. For example, it can be done on the fly by keeping track of the ratio of classes of instances transferred from the source datasets and ensuring that whenever an instance is added, the ratio remains almost the same. In other words, it guarantees that approximately the same number of instances from different classes is transferred to the target dataset. Another method is to perform a post-processing step and sub-sample the instances selected for transfer in such a way that the classes are balanced. In our implementation we have both methods although we elected to include the latter in Algorithm 2 (lines 10 and 11).

Algorithm 2: Class Balanced Similarity-Based Instance Transfer (CB-SBIT)

```
Input  : Source Datasets $S_1, S_2, \dots S_n$
Input  : Target Dataset $T$
Input  : Selected = [ ]
Input  : $thr_1, thr_2, \dots thr_k$
Output: New Dataset that is the result of $Concatenate(T, Selected)$
1  for $S \in [S_1, S_2 \dots S_n]$: do
2      for $I_s \in S$: do
3          for $I_T \in T$: do
4              $Sim_1 = ComputeSimilarity1(I_s, I_T)$;
5              $Sim_2 = ComputeSimilarity2(I_s, I_T)$;
6              $\dots$;
7              $Sim_k = ComputeSimilarity_k(I_s, I_T)$;
8              if $Sim_1 > thr_1 \& Sim_2 > thr_2 \dots \& Sim_k > thr_k$ then
9                  Add $I_s$ to Selected ;
10 $ClassBalancedSelected = SubSample(Selected)$;
11 $T_{NEW} = Concatenate(T, ClassBalancedSelected)$;
12 Return $T_{NEW}$;
```

The *SubSample* function in Algorithm 2 counts the number of instances in each class in the input dataset and randomly removes instances from the majority class(s) until the dataset is class balanced.

3 Experiments and Discussion

In this section we provide a detailed explanation of our experimental setups and discuss the results. We are going to evaluate the performance of some commonly used classifiers on Botnet network traffic data, compare CB-SBIT against the original SBIT and then against two algorithms using data from two different fields.

3.1 Evaluation of Classical Classifiers on Network Traffic Data

In this section we evaluate the performance of several classical classifiers on botnet network traffic data (we use data for the following five botnets: RBot, Smoke_bot, Sogou, TBot and Zeus. In the plot in Figure 1 these are shown in the x-axis as numbers from one to five. The y-axis in Figure 1 is the Accuracy. The main purpose of these experiments is to select the best performing algorithm so it can be used for comparison and as the base classifier for SBIT and CB-SBIT.

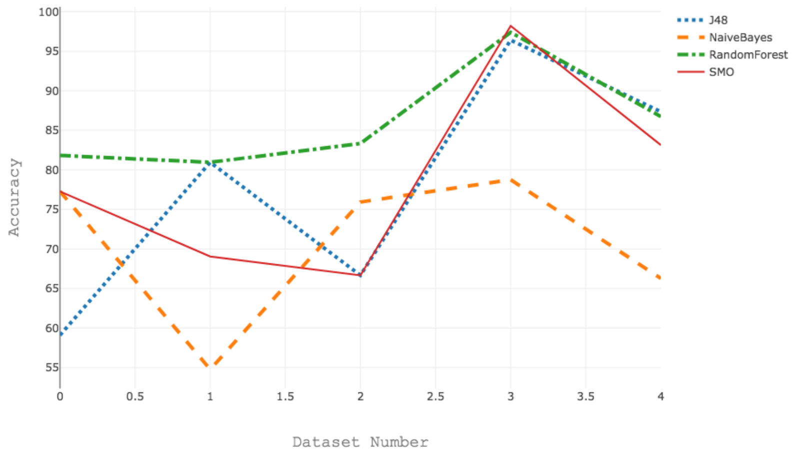
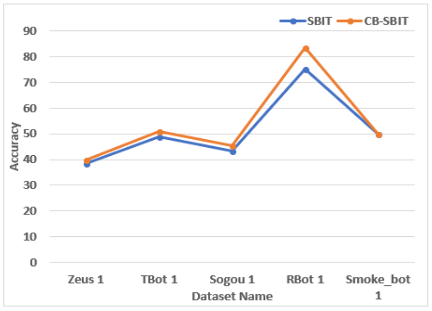


Fig. 1. Performance of Classical Classifiers on Network Traffic Data

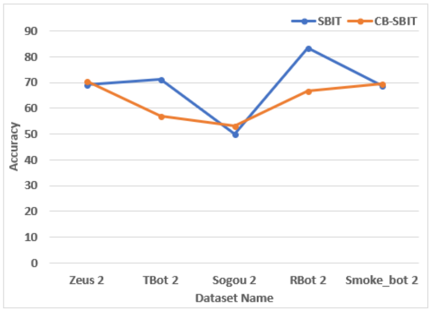
Figure 1 shows the average accuracy after running a ten-fold cross validation using WEKA's Decision Tree (J48), NaiveBayes, RanfomForest and SMO. It can be noticed that RandomForest scored the highest accuracy in more datasets than any other classifier. After performing the previous experiments, it becomes clear that RandomForest should be selected as the base classifier for the transfer learning algorithm developed as part of this work. This is because it performs better than other classifiers on network traffic data.

3.2 CB-SBIT vs SBIT (using Network Traffic Data)

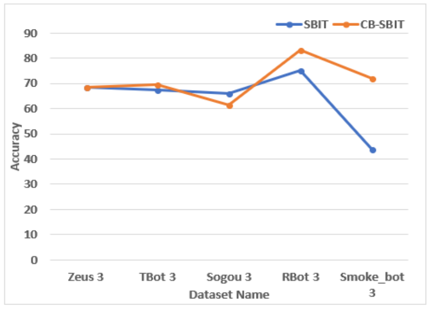
As explained in Section 2, SBIT and its extension CB-SBIT work by selecting instances from source datasets and transferring those instances to the target dataset. Currently the difference between the two algorithms is that CB-SBIT makes sure the new target dataset contains equal percentage of classes. In order to compare the two algorithms against each other, we have created varying sizes of small network traffic datasets. The reason we selected to work on small datasets is that transfer learning is normally applied when data is scarce. These datasets are the same datasets used in [2] (i.e. network traffic data that belong to the following five botnets: *Zeus*, *TBot*, *Sogou*, *RBot* and *Smoke bot*). As explained in detail in [2], each of these botnets has a target dataset and a testing dataset. Datasets that contain network traffic from *Menti*, *Murlo* and *Neris* botnets were used as source datasets.



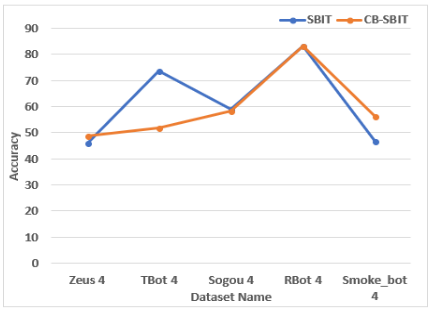
(a) Dataset 1×1



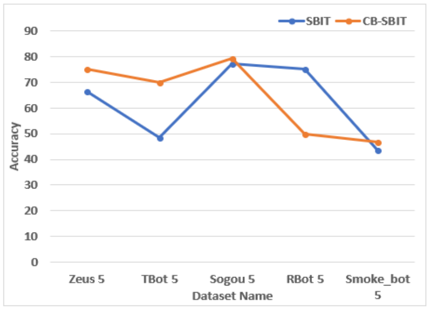
(b) Dataset 2×2



(c) Dataset 3×3



(d) Dataset 4×4



(e) Dataset 5×5

Fig. 2. Accuracy Values for CB-SBIT and SBIT

The contents of these datasets are derived from the freely available raw Botnet network traffic data which can be found in [15]. As this dataset is in raw format, we used FlowMeter [6] to generate several features that include statistical values as well as information such as Source Port, Destination Port and Protocol. Several steps were performed to transform this data into a suitable format for machine learning. The data is in packet capture (PCAP) format and contains traffic data for multiple Botnets as well as Normal traffic. we used FlowMeter to transform it into CSV format. We then followed guidelines provided by the data publisher to assign labels to instances and replaced missing values in each feature by the median of that feature. After this step we used one-hot encoding to represent source port, destination port and protocol fields in binary format, removed highly correlated features and detected and removed Outliers. After the pre-processing steps were completed, we split the data into smaller datasets according to label (each Botnet has a separate dataset) and used these datasets in our experiments. All of these steps are explained in detail in [1].

To perform experiments, we varied the size of each target dataset in such a way that each time the target dataset contains two, four, six, eight and ten instances (we made sure each dataset contains the same number of botnet and normal traffic to guarantee class balance). Then we ran SBIT and CB-SBIT on each of these datasets and evaluated their performance by computing the accuracy using the corresponding test dataset for each botnet. The accuracy values are illustrated in Figure 2. A description of the target datasets is provided in the first column in Table 1 in Section 3.3.

It is important to observe that although there are several metrics that can be used to evaluate the performance of classifiers [11], we have only used the accuracy (accuracy is the percentage of predictions that a model gets right). The reason is that our test datasets are class balanced.

Figure 2 illustrates the results of comparing the performance of CB-SBIT against that of SBIT using the experiment's datasets. It shows that CB-SBIT performs better than SBIT in general. Out of the 25 target datasets we used, CB-SBIT outperforms SBIT in 16 of them. However, SBIT still outperformed CB-SBIT in 6 datasets and they performed equally on three datasets.

3.3 CB-SBIT vs SMOTE (using Network Traffic Data)

The way SBIT and CB-SBIT work means new real data is being added to the target dataset. By real data we mean the data is not synthetically generated but rather it is collected from its original source. A common algorithm that is used to generate synthetic data is the SMOTE algorithm (or the Synthetic Minority Over-sampling Technique [4]) which generates synthetic instances for a particular class in a dataset. This section compares and evaluates the performance of CB-SBIT and SMOTE. The datasets in Section 3.2 were used in this evaluation and their full description is provided in Table 1.

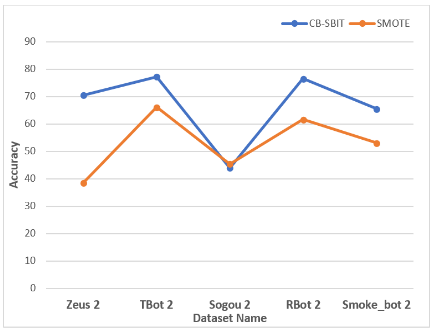
We varied the size of each target dataset so that each time the target dataset contains two, four, six, eight and ten instances - we ensured that each dataset contains the same number of botnet and normal traffic to guarantee class balance. Then we ran CB-SBIT on each of these datasets and saved the resulting target dataset - which now contains the original instances and instances added from source datasets. Using the number of instances of each class in all the resulting datasets, we ran SMOTE to generate new datasets of similar sizes using the original target datasets as the base datasets.

Dataset Name (size)	Size of Dataset generated by CB-SBIT	Size of Dataset generated by SMOTE
Zeus 1 (1×1)	32×32	-
Zeus 2 (2×2)	106×106	106×106
Zeus 3 (3×3)	108×108	108×108
Zeus 4 (4×4)	138×138	138×138
Zeus 5 (5×5)	156×156	156×156
TBot 1 (1×1)	42×42	-
TBot 2 (2×2)	161×161	161×161
TBot 3 (3×3)	211×211	211×211
TBot 4 (4×4)	274×274	274×274
TBot 5 (5×5)	360×360	360×360
Sogou 1 (1×1)	44×44	-
Sogou 2 (2×2)	67×67	67×67
Sogou 3 (3×3)	147×147	147×147
Sogou 4 (4×4)	170×170	170×170
Sogou 5 (5×5)	252×252	252×252
RBot 1 (1×1)	17×17	-
RBot 2 (2×2)	34×34	34×34
RBot 3 (3×3)	38×38	38×38
RBot 4 (4×4)	186×186	186×186
RBot 5 (5×5)	212×212	212×212
Smoke bot 1 (1×1)	1×1	-
Smoke bot 2 (2×2)	52×52	52×52
Smoke bot 3 (3×3)	58×58	58×58
Smoke bot 4 (4×4)	77×77	77×77
Smoke bot 5 (5×5)	96×96	96×96

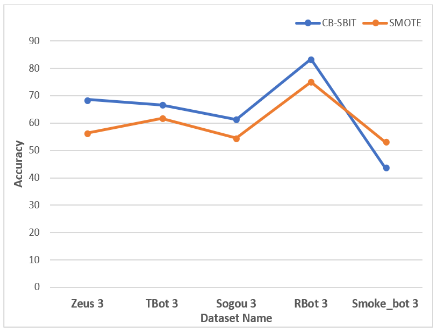
Table 1. Datasets Resulting after CB-SBIT and SMOTE

The first column of Table 1 shows the botnet name and the size of the baseline target dataset used (the 1×1 means this dataset contains only two instances, one botnet and one normal, the same concept applies for other sizes). The second column contains the size of the dataset after applying CB-SBIT using each target dataset as explained above (*number of botnet instances* $\times$ *number of normal instances*). The third column contains the size of the dataset after applying SMOTE using each target dataset. Observe that the cells corresponding to target dataset of size 1×1 is empty. This is because SMOTE requires at least two instances of each class to work. Therefore, because SBIT (and CB-SBIT) works normally even when the target dataset contains only one instance of one or more classes, we believe it is fair to conclude that CB-SBIT has a clear advantage over SMOTE when this is the case. In real life there may be cases where only one instance is present for a botnet family - especially when a botnet family is newly discovered.

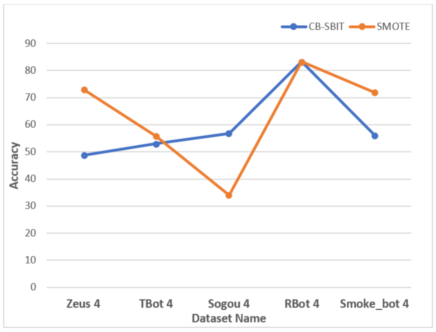
We have evaluated the performance of RandomForest using each one of them. We have run RandomForest on each dataset and computed the accuracy using the corresponding test dataset for each botnet. The accuracy values are illustrated in Figure 3.



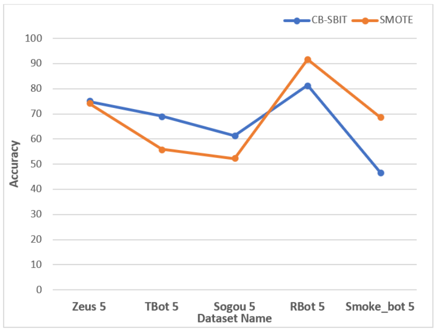
(a) Dataset 2 × 2



(b) Dataset 3 × 3



(c) Dataset 4 × 4



(d) Dataset 5 × 5

Fig. 3. Accuracy Values for CB-SBIT and SMOTE

Inspecting Figure 3 reveals interesting results. Because SMOTE does not work when the number of instances for any of the classes in the data is less than two, CB-SBIT has a clear advantage in this case. Figure 3a shows a similar behaviour that CB-SBIT performs better when the dataset size is small but greater than two. When the dataset size is increased gradually, the performance of SMOTE improves and it can be said that it performs equally to CB-SBIT. After using the 25 datasets described in Table 1, CB-SBIT performs better than SMOTE in 17 cases, SMOTE performs better than CB-SBIT in 7 cases and the two of them perform equally in one case. Recall that CB-SBIT (and SBIT) are proposed specifically to address the problem of scarcity of instances in the datasets. Clearly in this scenario CB-SBIT is a better choice than the classical SMOTE.

3.4 CB-SBIT vs TransferBoost (using Text Data)

For this comparison the popular 20 news groups dataset [12] was used to compare the performance of CB-SBIT against TransferBoost [7] and RandomForest. This dataset consists of 20,000 messages from 20 different netnews newgroups where 1000 messages were collected from each newsgroup. According to the guidelines provided in [12] the 20 groups can be generally categorised into the following six high level categories: computer (contains five sub-categories), miscellaneous (contains only one sub-category), recordings (contains four sub-categories), science (contains four sub-categories), talk (contains three sub-categories) and religion (contains three sub-categories). In order to perform our experiments we have chosen the following six datasets (one from each category): *misc.forsale*, *comp.graphics*, *alt.atheism*, *sci.electronics*, *rec.autos* and *talk.politics.misc*.

In order to obtain data suitable for machine learning, we used techniques popular in text mining [8]. Text mining involves using several techniques to process (usually unstructured) textual information and generate structured data which can be used to create predictive models and/or to gain some insight into the original textual information. The structured data is usually extracted by analysing the words in the documents and deriving numerical summaries about them.

To be able to use the text documents belonging to the six categories, we created a dataset that has two columns: the first column is the text contained in each document and the second column is the class of that document (which is one of the six categories). After that, we applied the TextToWordVector filter in WEKA [9] with Term Frequency and Inverse Document Frequency [18] (TF-IDF). TF-IDF is a widely used transformation in text mining where terms (or words) in a document are given importance scores based on the frequency of their appearance across documents. A word is important and is assigned a high score if it appears multiple times in a document. However, it is assigned a low score (meaning it is less important) if it appears in several documents.

We used WEKA's default parameters for this filter except for the *number of words to keep*. This parameter is 1000 by default, and we changed it to 10000. In addition to the TextToWordVector, we also used WEKA's NGramTokenizer (with NGramMinSize and NGramMaxSize set to two and three respectively). Not only this, but we also removed Stop Words using a freely available set of stop words. The resulting dataset contained as many as 10530 features and several thousand instances (belonging to the six classes).

The next step was to make sure datasets contained positive and negative examples. We have achieved this by choosing one of the six categories to be our negative class (we randomly chose misc.forsale data). After this, we split the large dataset into smaller datasets according to class and randomly selected a subset of 194 instances from each dataset (except the misc.forsale dataset). Then we randomly selected (without replacement) samples from the misc.forsale dataset and appended them to the other datasets. This was done to ensure that each dataset contains positive and negative instances. At the end of this step we had five datasets as follows: comp.graphics, alt.atheism, sci.electronics, rec.autos and talk.politics.misc (to clarify, the comp.graphics dataset now contains 388 instances, 194 of which are of the comp.graphics class and the remaining 194 are of the misc.forsale class, the same concept applies for the other four datasets).

Since transfer learning requires source and target datasets, we have randomly selected two of the five datasets to be our source datasets (these were the rec.autos and sci.electronics datasets). The remaining three datasets (comp.graphics, alt.atheism and talk.politics.misc) were our target datasets. We have randomly split each of these three datasets into smaller datasets (a target and testing datasets). Each target dataset contained 10 instances (five positive and five negative) and the remaining data was used as our testing datasets. Observe that we made sure we randomly select non-overlapping subsets in all previous steps. Details of these datasets are provided in Table 2.

Dataset Name	No of Instances	Dataset Usage
rec.autos	388 (194×194)	Source dataset
sci.elecronics	388 (194×194)	Source dataset
alt.atheism_Target	10 (5×5)	Target dataset
alt.atheism_Test	378 (189×189)	Test dataset
comp.graphics_Target	10 (5×5)	Target dataset
comp.graphics_Test	378 (189×189)	Test dataset
talk.politics.misc_Target	10 (5×5)	Target dataset
talk.politics.misc_Test	378 (189×189)	Test dataset

Table 2. Text Dataset Details

With this setup we have run experiments using RandomForest, TransferBoost and CB-SBIT. When using RandomForest, we have trained it using only the target datasets one at a time. This is because RandomForest only requires one dataset as its input. TransferBoost and CB-SBIT require one Target dataset and one or more Source Datasets, therefore we fixed the source datasets as shown in Table 2 and changed the Target dataset using the Target datasets we have selected. To evaluate, we computed the accuracy of each model using the corresponding test dataset. Our results are illustrated in Table 3.

Dataset Name	CB-SBIT	TransferBoost	RandomForest
alt.atheism	51.06%	89.68%	50.53%
comp.graphics	50.00%	78.84%	50.00%
talk.politics.misc	50.26%	87.56%	52.12%

Table 3. Results using Text Dataset

It is clear from Table 3 that when using textual data, TransferBoost outperforms RandomForest and CB-SBIT. This could be attributed to the nature of the data and how each algorithm works. It can be noticed that the performance of CB-SBIT and RandomForest are almost identical. This is because CB-SBIT uses RandomForest as its base learner and the fact that similarity values between instances in source and target datasets were found to be too small (when compared to the similarity values obtained when using network traffic data).

Table 4 shows computed percentage of similarity values that are greater than 0.5 for two example text and network traffic datasets. The first column of the table shows the two pairs used, while columns two to six show the percentage of similarity results that are greater than 0.5 for the five different types of similarity computation techniques we have used in our work: Tanimoto, Ellenberg, Gleason, Ruzicka and BrayCurtis. Note that the total number of similarity values is the product of the sizes of the pair of families/categories used. Further details on the similarity computation techniques can be found in [2].

Similarity between	Tanimoto	Ellenberg	Gleason	Ruzicka	BrayCurtis
Graphics - Autos	0.0093%	0.0093%	0.0193%	0.0093%	0.0193%
Politics - Electronics	0.0086%	0.0080%	0.0173%	0.0080%	0.0173%
Zeus - Sogou	12.6311%	91.2733%	97.3485%	7.9254%	14.1463%
TBot - Menti	2.9381%	85.6801%	99.8750%	2.0438%	3.0313%

Table 4. Percentage of Similairty Values that are > 0.5 using Text and Network Traffic Data

It is evident that there is much higher similarity in network traffic data than in text data. This means that CB-SBIT could hardly find any instances to transfer from the source to any of the target datasets when using text data. This is an interesting observation especially when it is compared to how CB-SBIT was able to transfer several instances when used with the network traffic data.

4 Conclusions and Future Work

This paper has introduced an extension to a novel transfer learning algorithm that is based on the similarity between instances from the target and source datasets (the SBIT algorithm). The extended version of the algorithm is aware of the percentage of classes in the resulting dataset (resulting after instance transfer) in the sense that it makes sure the classes are balanced. This helps in avoiding several problems such as overfitting and misinterpretation. The paper also included experimental evaluation of the new algorithm (i.e. the CB-SBIT algorithm) against the original SBIT algorithm as well as against two open source commonly used algorithms; the SMOTE and TransferBoost algorithm.

Experimental results show that CB-SBIT outperforms SBIT in majority of the tests; which means CB-SBIT is an improvement over SBIT. When comparing CB-SBIT against SMOTE, several network traffic datasets of various sizes were used and it was evident that CB-SBIT outperforms SMOTE in small datasets (CB-SBIT seems to perform better than SMOTE as the dataset gets smaller). An interesting case was when the dataset contains only one instance of one or more classes. SMOTE does not work in this case whereas CB-SBIT functions normally. On the other hand, text data from the publicly available 20 news groups dataset was used to compare the performance of CB-SBIT against TransferBoost. It was interesting to discover that, despite the fact that SBIT outperforms TransferBoost when using network traffic data as it was shown in the original SBIT paper, TransferBoost performs much better than CB-SBIT on text data. This could be due to the nature of the data and the transformations performed in pre-processing it. One interesting observation was made by CB-SBIT is that the similarity values between instances from different topics was very small. This accounts for the poorer perfornace of CB-SBIT on the text data. Similarity values were observed to be much higher in the network data where CB-SBIT performed very well.

References

1. Alothman, B.: Raw network traffic data preprocessing and preparation for automatic analysis. International Conference On Cyber Incident Response, Coordination, Containment & Control (Cyber Incident) - 2018 (Jun 2018)
2. Alothman, B.: Similarity based instance transfer learning for botnet detection. International Journal of Intelligent Computing Research (IJICR) **9**, 880–889 (Mar 2018)
3. Chawla, N.V.: Data Mining for Imbalanced Datasets: An Overview, pp. 875–886. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-0-387-09823-4_45, https://doi.org/10.1007/978-0-387-09823-4_45
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (Jun 2002), http://dl.acm.org/citation.cfm?id=1622407.1622416
5. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning. pp. 193–200. ICML '07, ACM, New York, NY, USA (2007). https://doi.org/10.1145/1273496.1273521, http://doi.acm.org/10.1145/1273496.1273521
6. Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., A., A.: Characterization of encrypted and vpn traffic using time-related features. In: ICISSP (2016)
7. Eaton, E., desJardins, M.: Selective transfer between learning tasks using task-based boosting. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11). pp. 337–342. AAAI Press (August 7–11 2011)
8. Feldman, R., Sanger, J.: Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, New York, NY, USA (2006)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. Newsl. **11**(1), 10–18 (Nov 2009). https://doi.org/10.1145/1656274.1656278, http://doi.acm.org/10.1145/1656274.1656278
10. He, H., Ma, Y.: Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley-IEEE Press, 1st edn. (2013)

11. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, New York, NY, USA (2011)
12. Lang, K.: 20 newsgroups data set, http://www.ai.mit.edu/people/jrennie /20Newsgroups/
13. Liu, B., Xiao, Y., Hao, Z.: A selective multiple instance transfer learning method for text categorization problems. Knowledge-Based Systems **141**, 178 – 187 (2018). https://doi.org/https://doi.org/10.1016/j.knosys.2017.11.019, http://www.sciencedirect.com/science/article/pii/S0950705117305415
14. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. **22**(10), 1345–1359 (Oct 2010). https://doi.org/10.1109/TKDE.2009.191, http://dx.doi.org/10.1109/TKDE.2009.191
15. Samani, E.B.B., Jazi, H.H., Stakhanova, N., Ghorbani, A.A.: Towards effective feature selection in machine learning-based botnet detection approaches. 2014 IEEE Conference on Communications and Network Security pp. 247–255 (2014)
16. Sun, G., Liang, L., Chen, T., Xiao, F., Lang, F.: Network traffic classification based on transfer learning. Computers & Electrical Engineering (2018). https://doi.org/https://doi.org/10.1016/j.compeleceng.2018.03.005, http://www.sciencedirect.com/science/article/pii/S004579061732829X
17. Torrey, L., Shavlik, J.: Transfer learning. Handbook of Research on Machine Learning Applications. IGI Global **3**, 17–35 (2009)
18. Weiss, S., Indurkhya, N., Zhang, T., Damerau, F.: Text Mining: Predictive Methods for Analyzing Unstructured Information. SpringerVerlag (2004)
19. Zhao, J., Shetty, S., Pan, J.W.: Feature-based transfer learning for network security. In: MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM) (Oct 2017)