HOT ZONE IDENTIFICATION: ANALYZING EFFECTS OF DATA SAMPLING ON SPAM CLUSTERING

Rasib Khan
rasib@cis.uab.edu
Mainul Mizan
mainul@cis.uab.edu
Ragib Hasan
ragib@cis.uab.edu
Alan Sprague
sprague@cis.uab.edu
Department of Computer and Information Sciences
University of Alabama at Birmingham
115A Campbell Hall, 1300 University Boulevard
Birmingham, Alabama 35294-1170

ABSTRACT

Email is the most common and comparatively the most efficient means of exchanging information in today's world. However, given the widespread use of emails in all sectors, they have been the target of spammers since the beginning. Filtering spam emails has now led to critical actions such as forensic activities based on mining spam email. The data mine for spam emails at the University of Alabama at Birmingham is considered to be one of the most prominent resources for mining and identifying spam sources. It is a widely researched repository used by researchers from different global organizations. The usual process of mining the spam data involves going through every email in the data mine and clustering them based on their different attributes. However, given the size of the data mine, it takes an exceptionally long time to execute the clustering mechanism each time. In this paper, we have illustrated sampling as an efficient tool for data reduction, while preserving the information within the clusters, which would thus allow the spam forensic experts to quickly and effectively identify the 'hot zone' from the spam campaigns. We have provided detailed comparative analysis of the quality of the clusters after sampling, the overall distribution of clusters on the spam data, and timing measurements for our sampling approach. Additionally, we present different strategies which allowed us to optimize the sampling process using data-preprocessing and using the database engine's computational resources, and thus improving the performance of the clustering process.

Keywords: Clustering, Data mining, Monte-Carlo Sampler, Sampling, Spam, Step Sequence Sampler, Stepping Random Sampler, Hot Zone

1. INTRODUCTION

Advancement of the IT infrastructure significantly affects the way people communicate. Social interaction and information exchange are highly dependent on emails and other such forms of media. At the same time, such medium of communication has been the target of misuse since the beginning. Thus, the negative motives from spammers have been a serious issue, which have led to phishing, viruses, malware bots, and other such attacks.

Spam emails are mostly generated by malware bots on different computers across the Internet. However, malwares installed by the same spammer exhibit a specific pattern in the spam emails (Nhung and Phuong 2007; Ying et al., 2010). The content of the spam is usually generated using a common template. Therefore, the identification of the pattern in these spam emails is significantly important to IT forensic experts. The identified pattern can then help identify a specific spammer and follow through with proper investigations (Dagon et al., 2007; Ono et al., 2007). Mining spam emails helps discover and correlate useful patterns. Most of the mining techniques are text-based, given that such spam emails are mostly text-oriented. Once the emails are scrutinized for such patterns, different clustering techniques and algorithms can be applied over the email data to group the spams based on some similarity criteria. The speed of producing faster clusters from large datasets depends on efficient algorithms. However, in case of very large datasets, it might be required to reduce the size of the data prior to the clustering process.

In this paper, we focus on the evaluation of clustering performed on sampled spam emails. The data used is from the Spam Data Mine at the University of Alabama at Birmingham (UAB) (UAB-CIS, 2013). The UAB Spam Data Mine is a large and widely researched repository for spam emails, and is used as a helpful resource by researchers from different global organizations. Given the huge number of spam emails collected every day, the clustering of the spams take a long time. However, in this work, instead of focusing on algorithms to optimize the clustering process, we considered sampling the dataset prior to fetching it to clustering algorithms. Once we are able to prove sampling as an efficient and applicable solution for data reduction, we believe appropriate clustering algorithms can be applied accordingly. We have adopted the previous work done by Chun Wei et al., to create the clusters based on patterns in the subject header of the spam emails (Wei et al., 2009).

In this work, we have utilized four simple methods of sampling that we have applied on the spam data from the data mine. As a result, we aim in making the process of clustering more efficient and less time consuming. Furthermore, we provide the results to illustrate that the sampled data from the UAB Spam Data Mine preserves the information contained for forming clusters and highlight the 'hot zone'. In this context, we refer to 'hot zone' as the most prominent clusters with respect to spamming activities. We have presented the results in order to support our claim of using sampled spam data to allow investigators a faster and better opportunity to identify the 'hot zone' in spam clusters. We illustrated the resulting clusters from the sampled data, and performed extensive comparative analysis with the clusters formed using the whole data set. Our evaluation includes an analysis of the data distribution on the spam data, and also the time measurements for the different operations in the algorithm. The paper also includes a different approach to optimize the sampling process, utilizing the efficiency of the database engine, which allowed us to enhance the resulting performance of the required time.

Contributions: The contributions in this paper are as follows:

- We evaluate the sampling methods on actual spam emails from the UAB Spam Data Mine. The validation and effectiveness of sampling is based on the following: (a) quality of the clusters produced, (b) the data cover/distribution of spam emails within the data mine, and (c) the timing performance for the clustering operation. All the sampling models have been validated for varying sampling rates against the clusters created using the complete data set. Our results show that we are successfully able to highlight the 'hot zone' from the spam emails with a significant improvement in timing performance.
- We present techniques and strategies for the most efficient way to implement the sampling process and retrieve the huge number of spam emails from the data mine, which are then used to execute the clustering algorithm. The experimental measurements using our optimization strategies illustrate that there are further improvements in performance, compared to naïve SQL query based retrieval of sampled spam records from the UAB Spam Data Mine.

The rest of the paper is organized as follows. The motivation for the work is presented in Section 2. Section 3 describes the organization of the UAB Spam Data Mine, including the clustering algorithm from the work of Wei et al. (2009). The different sampling models are described in Section 4. The results and corresponding analysis are presented in Section 5. Section 6 includes the optimization strategies to improve the efficiency of the sampling process. Finally the related works and conclusion are presented in Section 7 and Section 8 respectively.

2. RESEARCH MOTIVATION

The increasing number of Internet users has attracted criminals to the field of online crimes. eCrimes have been significantly on the rise since the last few years. This section illustrates the issue of eCrimes on the Internet, and the research motivation behind the work on investigating spam clusters, and the importance of identifying the hot zone.

2.1 eCrimes on the Internet

Information security and economics have become interdependent in recent times. Corporations employ information security specialists, as well as economists and lawyers to deal with the rising concern of eCrimes. The network of criminal activities has become more organized with structured online black markets, where the criminals trade insider information. Data and information, such as credit card and PIN codes, are sold to online anonymous brokers in these underground eCrime markets. According to Moore et al. (2009), credit card information are sold at advertised prices of $0.40 to $20.00 per card, and bank account credentials at $10 to $100 per bank account. Social security numbers and other personal details are sold for $1 to $15 per person, while online auction credentials fetches around $1 to $8 per identity. Subsequently, the brokers sell the information to specific expert hackers, who perform the final act of money laundering.

The information collected in these online criminal activities incorporate specialized approaches. Usually, Internet users are driven to false websites with the help of advertising emails. These bulk emails are generally classified as spams, which are sent by spammers, using malicious software running on infected machines. The infected computers are used by the spammers to record keystrokes and send further spam emails.

The monetizing channel for spam emails includes multiple organizations. It is illustrated by Levchenko et al. (2011), the spam value chain has multiple links between the money handling authorities and the spammers. Furthermore, according to an approximate consensus, 5% of online devices on the Internet are susceptible to being infected with malware. At least 10 million personal computers have been assumed to be infected with malware in 2008, the number for which should have had increased significantly over the last few years (Moore et al., 2009). Thus, these figures easily indicate that the network for criminal activities have outgrown the authorities dealing with eCrimes.

2.2 Spam Investigation

Spam emails are perceived as being analogous to junk mails. These emails are generally advertising emails, or with other forms of undesired content. However, spam emails are not as innocent as junk mails. They are sent to a large number of recipients, and usually have hidden motives along with the content of the email. They are considered as the primary channel for attackers to deploy Trojans, worms, viruses, spyware, and botnets on other machines across the Internet.

The email body of spams has hidden scripts, cookies, and other attached content to attract the recipient of the email. Once the user opens the email, the scripts may use the current information from the browser to expose the identity of the user to the attacker. This is the easiest and a very well-known approach, but still the most common scenario where users are victims of identity thefts on the Internet. This information can be used to remotely access the user's machine and install unwanted malwares as botnets. The malware can then operate from the infected machine using the identity of the user, and send further spam emails or perform other unwanted tasks.

When an attacker sends a spam, he generally uses a template to generate the content of the email. The format of the content is thus prevalent in all the spam emails those are being sent. However, the spammers replace some words or phrases to introduce variation and hence bypass the spam filters. Thus, it becomes a non-trivial task for such filtering services to detect all the spam. Data mining from spam emails is useful to detect and investigate these patterns. The spam emails are scrutinized and parsed into different text-based segments. Each email comprises of certain attributes, such as the sender email, subject header, and the mail body. These individual attributes can be investigated to match other spam emails, and thus grouping similar spam emails. Once a pattern is observed, they can be clustered and classified as a specific spam campaign (Caruana and Li 2008; Kyriakopoulou and Kalamboukis 2008; Sasaki and Shinnou 2005; UAB-CIS 2013; Wei et al., 2009; Ying et al., 2010). The individual clusters obtained from grouping spam emails allow the eCrime investigators to identify a particular spammer. The clustered spams are examined to classify the spammer and obtain further track-down information. eCrime investigators use these collected data to hunt down online criminals and take appropriate actions against the involved personnel.

The Spam Data Mine at UAB collects approximately 1 million spam emails each day (UAB-CIS, 2013). The spam emails can then be used to find the patterns and perform clustering on the collected data. The identified clusters are assumed to be individual spam campaigns by an attacker. The extracted patterns from the spam emails are dependent on the template used by the spammer to generate the spam. However, it should also be noted that an attacker generally uses a given spam template for a few days, after which he changes the format of the emails. This constant change in the format of the spams makes it difficult to identify a particular attacker. As a result, spam emails collected over a small duration of time exhibits the specific pattern, after which the extracted cluster information does not apply any more.

From the above scenario, we have observed the following requirements for investigating eCrimes using spam clusters. First, it is important that the identification of the spam campaigns should be done as early as possible. The multitude of financial loss resulting from eCrimes requires the investigation to proceed quickly. The sooner a particular spam campaign is taken down, the lesser is the financial loss. A quick action against a spam campaign would also mean that lesser people will fall as victims to the campaign on the Internet. However, given the huge amount of data, it requires a lot of time to execute the clustering operation. Thus, the inherent requirement to act quickly against such eCrimes is not fulfilled with the current approaches for clustering spam emails. Moreover, the quickly changing pattern of templates by the spammers makes it more difficult to extract the information from the spams and act on it accordingly.

Second, the 'hot zone' of the spam campaigns are the ones about which conclusive remarks can be made about an attacker. Here, we refer 'hot zone' as the group of largest clusters and the most prominent spam campaigns on the Internet. The largest spam clusters imply a large number of similar spam emails. As a result, the larger clusters incorporate more information for the eCrime investigators and law enforcement authorities to study the criminals. It is more important to identify the largest clusters rather than obtaining an extensive number of clusters for the huge amount of spam from the data mine. It might not be the same scenario when it comes to user privacy protection and spam filters on web browsers and email clients, where more fine-grained spam filtering is required to protect the users on the Internet. Therefore, when it comes to criminal investigations and law enforcement, the prominent clusters are the ones of interest, while the smaller ones can be classified as outliers.

3. CLUSTERING SPAM DATA

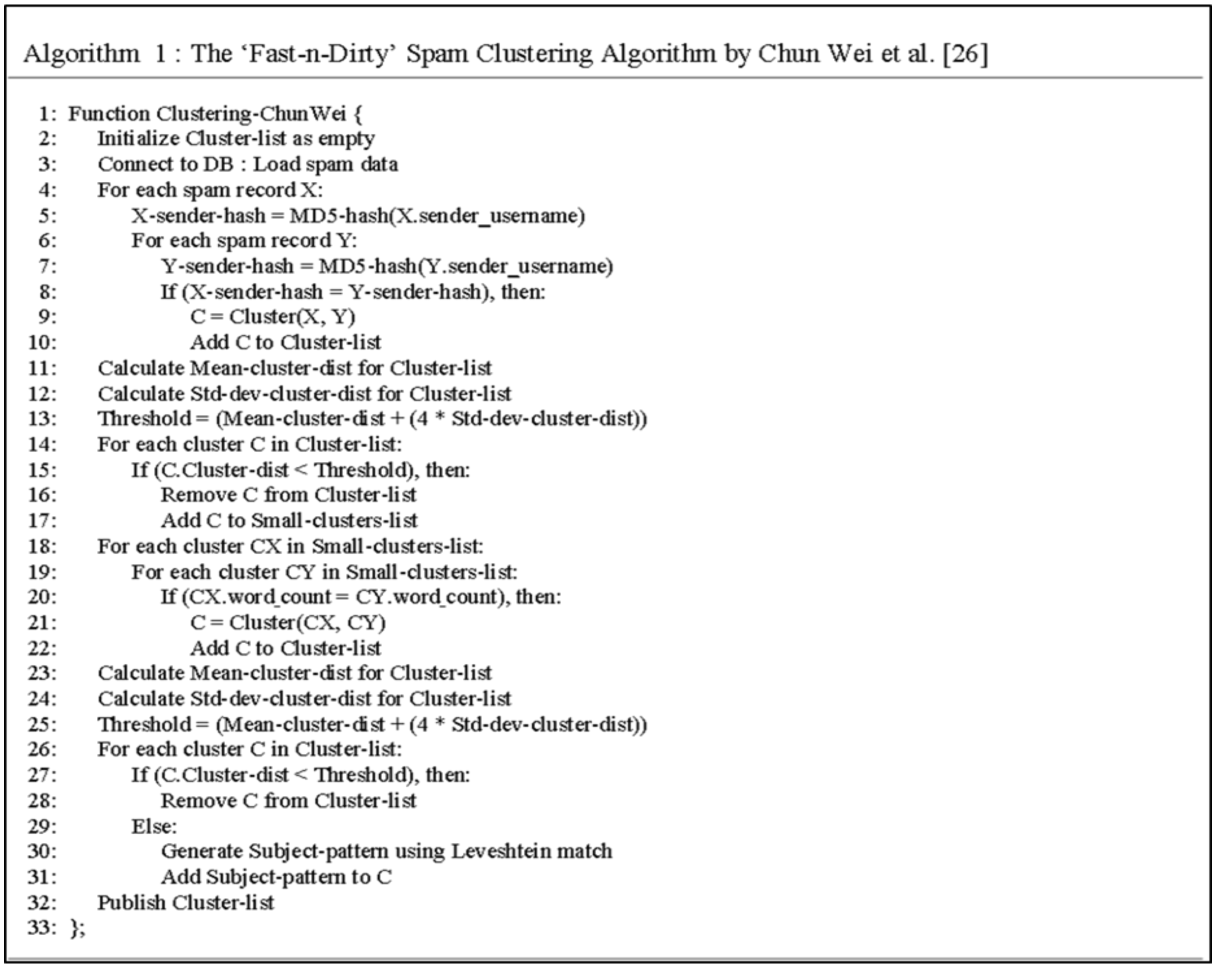
For our work in this paper, we have adopted an existing clustering algorithm proposed by Wei (2010) and Wei et al. (2009). The algorithm has been executed using data from the UAB Spam Data Mine (UAB-CIS, 2013). In this section, we discuss the background and the description of the data mine, including the clustering technique proposed by Chun Wei et al. (2009, 2010) on the spam data.

3.1 Background

The initial research issue for knowledge extraction or data mining is classifying data and creating representations of the feature space. Clustering is most commonly used for feature compression and extracting information (Kyriakopoulou and Kalamboukis, 2008). Specific features are compared and clustered into groups which represent a commonality among all of its data items. The task of measuring the similarity of data items can be performed in different ways. The most common methods for measuring similarity/dissimilarity are Jaccard and Levenshtein coefficients (Jaccard 1901; Levenshtein 1966). The distances can then be used in other clustering algorithms to create and evaluate clusters (Caruana and Li 2008; Kanungo et al., 2002; Hartigan and Wong 1979; Wei 2010; Ying et al., 2010). The clustering algorithms thus use the similarity or dissimilarity of individual data items based on the feature space, and group them into a common cluster based on preset threshold configurations.

3.2 The Spam Data Mine

We utilized the UAB Spam Data Mine (UAB-CIS, 2013) for the purpose of our research evaluation. The UAB Spam Data Mine is a research project under The Center for Information Assurance and Joint Forensics Research (CIS-JFR)[1]. The Center generates information about currently on-going campaigns by spammers. It archives spam emails received from numerous sources and honey-pots, and collects approximately 1 million spam emails each day.



Algorithm 1 : The 'Fast-n-Dirty' Spam Clustering Algorithm by Chun Wei et al. [26]

```
Function Clustering-ChunWei {
    Initialize Cluster-list as empty
    Connect to DB : Load spam data
    For each spam record X:
        X-sender-hash = MD5-hash(X.sender_username)
        For each spam record Y:
            Y-sender-hash = MD5-hash(Y.sender_username)
            If (X-sender-hash = Y-sender-hash), then:
                C = Cluster(X, Y)
                Add C to Cluster-list
    Calculate Mean-cluster-dist for Cluster-list
    Calculate Std-dev-cluster-dist for Cluster-list
    Threshold = (Mean-cluster-dist + (4 * Std-dev-cluster-dist))
    For each cluster C in Cluster-list:
        If (C.Cluster-dist < Threshold), then:
            Remove C from Cluster-list
            Add C to Small-clusters-list
    For each cluster CX in Small-clusters-list:
        For each cluster CY in Small-clusters-list:
            If (CX.word_count = CY.word_count), then:
                C = Cluster(CX, CY)
                Add C to Cluster-list
    Calculate Mean-cluster-dist for Cluster-list
    Calculate Std-dev-cluster-dist for Cluster-list
    Threshold = (Mean-cluster-dist + (4 * Std-dev-cluster-dist))
    For each cluster C in Cluster-list:
        If (C.Cluster-dist < Threshold), then:
            Remove C from Cluster-list
        Else:
            Generate Subject-pattern using Leveshtein match
            Add Subject-pattern to C
    Publish Cluster-list
};
```

The collection of spam emails from the sources is collected in a batch-wise operation. General users on the Internet, upon receiving a (suspected) spam email, marks the email as spam, and forwards it to the honey-pot email address for archiving. Additionally, numerous other honey-pots are placed at different points in the network which dedicatedly receive and archive spam emails. The archived spam emails are collected batch-wise at specific time intervals during the day. Thus, due to the manner these spam emails are stored and collected in the data mine, the records do not display a shuffled organization in their sequence.

Subsequently, the spam data mine stores the data regarding spam emails parsed into different attributes. The current database design holds the following attributes for each spam email: *message_id*, *subject*, *sender_name*, *sender_username*, *sender_domain*, *sender_ip*, *receiving_date*, *time_stamp*, *word_count*.

3.3 Algorithm for Clustering

The method employed by Wei et al. (2009) for clustering the spam data is specific to the data from the UAB Spam Data Mine (UAB-CIS, 2013). In this section, we present the clustering algorithm designed and implemented by Wei et al. (2009) and

[1] The Center (CIS-JFR), http://thecenter.uab.edu

also included as a part of the work in Wei (2010). For our purpose, we chose the rather 'fast-n-dirty' version of the clustering algorithm by Wei, which is shown in Algorithm 1. The clustering algorithm matched spam emails on exact similarity of sender email addresses. They are matched using the MD5 hash of the sender's email. Similar items were clustered into a common group. From within the clusters, some of them are set aside using a bounded threshold, which was set at a minimum of (*mean + (4*standard deviation)*).

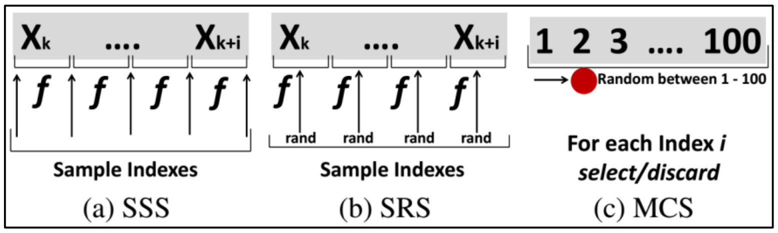


Figure 1 Sampling Methods: Step Sequence Sampler (SSS), Stepping Random Sampler (SRS), and Monte Carlo Sampler (MCS)

Next, the process was repeated for the *word_count* of the email body for all the small clusters, and further clusters were created. As a result, some of the clusters had both the *sender_name* and the *word_count* in the feature space, while some only had the *word_count* criteria. Finally, a Levenstein index is calculated to create a common pattern for the *subject* header for each of the clusters. The output patterns of subject headers for the spam emails are produced in the form '*__ similar __ word'*. Here, the blank spaces are the words which could be substituted for other words. The blank spaces together with the words '*similar*' and '*word*' define the basic template of the subject headers for each of the clusters of similar spam emails.

4. SPAM DATA SAMPLING

Sampling is a well-known technique for data reduction, given that it preserves the information from the original data set. In this section, we present our approaches to create the sampled data. We have presented four different schemes for creating the sampled data, which have been discussed in the following sections. For each of the models, we invoke the sampling method with the begin index, end index, and sampling rate parameters.

4.1 Simple Random Sampler

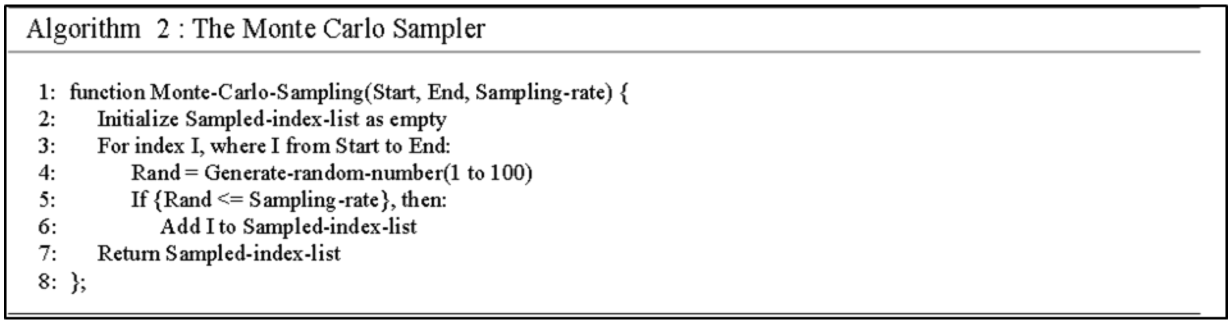
The simple random sampler is implemented using the Java Random class[2]. The Java Random class initializes using a 48-bit long random seed. Subsequently, it is modified using a linear congruential formula to generate a stream of pseudo-random numbers (Knuth, 2006). Alternatively, Mersenne Twister is another method for polynomial calculations over two-element fields to generate uniform pseudo-random numbers (Matsumoto and Nishimura 1998). However, our random generator uses the linear congruential formula due to the simplicity of the model, and serves the purpose of our work.

The simple random sampler takes in a range of values within a begin/end index for *message_id*s. Subsequently, it generates the random indexes within the given range, according to the desired sampling rate. However, the generated random indexes may or may not be evenly distributed across the range of values for the *message_id*s.

4.2 Step Sequence Sampler

The step sequence sampler is another method of sampling which we utilized for our spam data. As shown in Figure 1a, given the sampling rate *r*, we initially calculated the step frequency *f*. The range of values for the *message_id*s is then divided into *f*-segments, and the boundary index values are returned as the sampled indexes. As a result, the obtained sampled data is evenly distributed, and sequentially selected from the data set.

[2] Java Random class, http://docs.oracle.com/javase/7/docs/api/java/util/Random.html



Algorithm 2 : The Monte Carlo Sampler

```
function Monte-Carlo-Sampling(Start, End, Sampling-rate) {
    Initialize Sampled-index-list as empty
    For index I, where I from Start to End:
        Rand = Generate-random-number(1 to 100)
        If {Rand <= Sampling-rate}, then:
            Add I to Sampled-index-list
    Return Sampled-index-list
};
```

4.3 Stepping Random Sampler

The stepping random sampler is an extension of the step sequence sampler, as shown in Figure 1b. As before, we calculated the step frequency *f* for the given range of *message_id*s based on the sampling rate. After that, we utilized the Java Random class to randomly select an index from within each block. Thus, the sampled index values for the *message_id*s are evenly distributed with the frequency *f*, and randomized within each blocked segment, thus ensuring unbiased results.

4.4 Monte Carlo Sampler

Monte Carlo methods refer to computational algorithms which are based on repeated random sampling to obtain a desired goal. It is a process of calculating heuristic probability for a given scenario which is defined by the specific validation of a success or fail event (Hammersley et al., 1965). In our case, we designed a simple Monte Carlo sampler to probabilistically generate some random indexes for choosing the sampled *message_id*s, as illustrated in Figure 1c, and presented in Algorithm 2.

In the Monte Carlo sampler, for each index *i*, where *i* is between begin and end, we 'roll' between 0 -100. If the random 'roll' is less than or equal to the sampling rate r, we select the specific index *i*. Thus, the sampled indexes are sequentially selected or discarded from within the range of begin and end indexes for *message_id*s. However, the number of index values that we receive from the Monte Carlo sampler is not exact, but probabilistically close to match the sampling rate *r*. The success or fail events in Monte Carlo models are usually executed for a large number of events. Therefore, according to the model, the larger the range of *message_id*s, the closer we get to the desired value for the number of sampled items (Hammersley et al., 1965).

4.5 Comparison of Sampling Methods

Table 1 Comparison of properties for the Random Sampler (RS), Step Sequence Sampler (SSS), Stepping Random Sampler (SRS), and the Monte Carlo Sampler (MCS)

	RS	**SSS**	**SRS**	**MCS**
Randomness	good	bad	med	good
Sequential	no	yes	yes	yes
Repetition	maybe	no	no	no
Data cover	maybe	yes	yes	maybe
Number of samples	n*r	n*r	n*r	≈ n*r

The properties of the different sampling methods are summarized in Table 1. In this context, we define the following properties for the different sampling methods.

i. Randomness in the sampling process implies the probability of a particular index being chosen in the sample.
ii. Sequential sampling refers to the criteria of the chosen indexes being in order once the sampling process has completed.
iii. Repetition in sampling means the possibility of an index being chosen more than once.
iv. Data cover represents the feature of the chosen sampled indexes being evenly distributed over the range of values from the original data set.
v. Number of samples refers to the number of indexes chosen, given the total number of indexes n, and the sampling rate r.

As shown in Table 1, the simple random sampler provides good randomness, as it depends on a simple linear congruential formula to generate the pseudo-random number stream. However, it is not sequential, as the chosen index samples are generated at random, and does not preserve order. Additionally, the simple random sampler does not guarantee uniqueness, as the same number can be generated more than once. Therefore, the already mentioned properties can be utilized to state that the simple random sampler does not provide a guaranteed data cover either. The step sequence sampler does not provide any randomness and is purely sequential. However, we are able to ensure no repetition and full data cover. Using the stepping random sampler allows mediocre randomness, but contains sequence, ensures uniqueness, and also provides a full data cover. Finally, the Monte Carlo method provides good randomness and ensures sequentiality with no repetition. However, it has a probabilistic sample size of approximately (*n*r*), where n is the data size and r is the sampling rate. The probability of the sample size will get closer to (*n*r*) with a greater range of values for the indexes.

5. RESULTS AND ANALYSIS

In this section, we present the results obtained from the different sampling methods presented previously. The sampled data were mined and used to create clusters, based on the algorithm of Wei et al. (2010) (Ying et al., 2010). We also provide an analysis of the results and comparison of each of the sampling methods against clustering performed on the full data set. The results presented have been generated using two days' spam data. As mentioned earlier, the data mine collects a huge number of spam emails, and there were a total of approximately 1.8 million spam emails in these two days.

5.1 Clustering Quality

Initially, we performed the clustering on the whole spam data for a range of two days. With the clusters formed, we selected the ten largest clusters and analyzed their statistics. We recorded the number of data points, pattern of the subject within the cluster, and the percentage of data that each of the clusters has with respect to the data size. We refer to clustering factor as the value between 0 and 1, which represents the size of the cluster in terms of the size of the data. The rightmost bar on Figure 2 shows the distribution of the clusters which were created from complete data set for the given range of days. It can be seen that the ten largest clusters actually represent almost 25% of the whole data set, with three largest clusters representing approximately 9%, 8%, and 3% respectively.

Next, we executed the clustering algorithm on sampled data with each of our samplers. The sampling was performed at varying rates of 1%, 2%, 3%, 5%, and 8% respectively. For each of the cases, we analyzed the clusters created with the sampled data. To visualize the clustering quality with better understanding, we normalized each of the sampled clusters using the size of the sample to calculate the clustering factor for each. Using a normalized view for the sampled clusters thus makes it easier to evaluate the quality of the clustering with respect to the clusters formed using the full data set. The clustering factor for each of the sampling methods at varying sampling rates is illustrated in Figure 2.

From the results, it can be seen that random sampling, step sequence, and stepping random create the clusters with a similar clustering factor as that of the full data set. Thus, the more similar the clustering factors and distributions are, the better they can be claimed to have performed. It should also be noted that all the three sampling methods perform in a stable manner with their varying sampling rates. Additionally, we verified that each of the ten largest clusters from the sampled data actually coincides with at least eight of the largest clusters from the full dataset. However, they might sometimes be slightly out of order in the sampled cluster sizes. Moreover, the top three to five clusters as shown in Figure 2 is always the same clusters in all the cases, which verifies that the sampling effectively allows us to identify the ‘hot zone’ of spam campaigns. Table 2 describes the patterns of subject headers for each of the top ten clusters created in order of their sizes. It can be seen that most of the clusters created from the 2% step sequence sampling are exactly in the same order if compared to the clusters created using the full data set. However, there are minor interchanges in the position of the clusters in their ordering. Nonetheless, they are not the top clusters, and are usually of similar sizes and hence tend to swap places with minor changes in the order.

Table 2 Subject Header Patterns of Ten Largest Clusters Compared using Full Dataset Vs. 2% Sampled Data

No.	Clustering on full data set	Clustering using 2% Step Sequence
1	Canadian Pharmacy: BUY NOW VIAGRA & CIALIS !	Canadian Pharmacy: BUY NOW VIAGRA & CIALIS !
2	New prices	New prices
3	Lowest prices	Lowest prices
4	_ Vigara Now ___	_ Vigara _ = ____
5	_ Vigara _______	_ Vigara Now ___
6	Corporate eFax message - _ pages	Corporate eFax message - _ pages
7	_ Vigara _ SALE!	United Parcel Service notification ___
8	United Parcel Service notification ___	_ Vigara _______
9	Vigara Now ____	_ Vigara = ______
10	_ Vigara _ Off!	Purchase your Levitra from one of our drugstores today. Levitra/Viagr/Cialis from $1.25 _____

However, with the Monte Carlo sampler, it can be seen that the sampled data had some skewness towards the clustering data points. This can be claimed as both positive and negative. Given that the results tend to have a greater clustering factor for the larger clusters and represent almost 45% of the sampled data, it can be argued that Monte Carlo sampling makes it easier to focus on the largest clusters. However, they tend to distort the actual distribution of clusters and misrepresent the clustering factor for each of the clusters compared to the full data. An interesting convergence towards the desired clustering factor distribution can be seen as the sampling rate is increased.

Therefore, from the clusters created and the clustering factors, we are able to infer the effect of the different sampling methods. It can be seen that random, step sequence, and stepping random sampling tends to preserve the distribution of the original data set of spams. Therefore, we can say that the sampling models for the above three are *representative* sampling. On the other hand, Monte Carlo seems to perform well in highlighting larger clusters and removing noise from smaller clusters. Hence, we call it *noise suppressive* sampling. Given the context and the requirement, each of the sampling methods can be utilized accordingly.

5.2 Data Cover

We utilized the clusters created from our experiments to analyze the distribution of the data in the spam data mine. We are interested to visualize how the spam emails have been archived in the data mine, with respect to the cluster each spam email belongs to. In this context, data cover refers to the distribution of the spam emails in the data set.

Figure 3 illustrates the graph to help visualize the distribution for the complete dataset. The x-axis corresponds to the total number of *message_id*s for the given date. The y-axis specifies the number of spam emails in the cluster to which the corresponding *message_id* belongs to. The colored lines are formed by very closely placed data points, and each of the colors represents a different cluster.

We also present the data cover graphs generated from the clusters created using the four different sampling methods, shown in Figures 4, 5, 6, and 7 respectively. The sampled graphs have been produced only for a sampling rate of 2%, which is sufficient to prove the effectiveness of sampling. It can be seen that each of the sampling methods have been equally capable to successfully identify the same top clusters which have been created by the complete data set. Additionally, it can be seen that most items which belong to the same cluster reside closely in the data set. This observation is useful in asserting the fact that sampling the data which preserves the sequentiality is also able to preserve the representation of the dataset.

An interesting observation is the comparison of tailing or sparse data from Figure 3 compared to any of the other Figures 4, 5, 6, and 7. All the sampling methods have nicely cleaned the scattered data points.

However, the sampled data for step sequence sampler and Monte Carlo sampler (Figure 5 and 7) still shows some minor traces of the existence of the scattered data in comparison to the original data. In all the cases, the leveling clusters at the bottom are cluttered together. However, these are the smaller clusters and do not play any interesting role in the identification of the ‘hot zone’.

Thus, Figures 3, 4, 5, 6, and 7 illustrates the way the data set is organized. This can lead us to generalize a pattern of arrivals of spam emails into the archive. Additionally, such a pattern of data arrival strengthens ours claim of sampling being sufficient and effective to preserve the characteristics of the dataset and the largest clusters from the spam emails in the data mine.

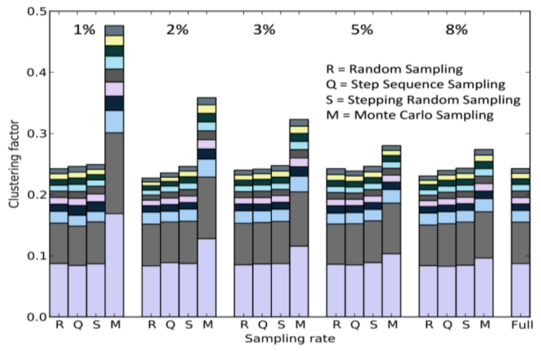


Figure 2 Clustering Factor for Ten Largest Clusters

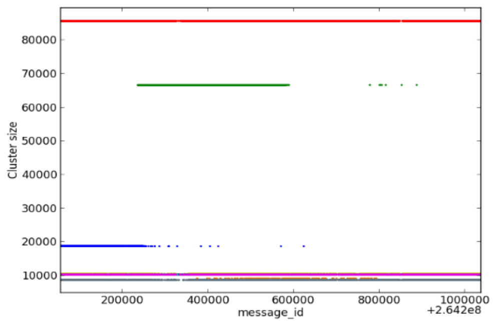


Figure 3 Spam Distribution based on Clusters for Complete Dataset

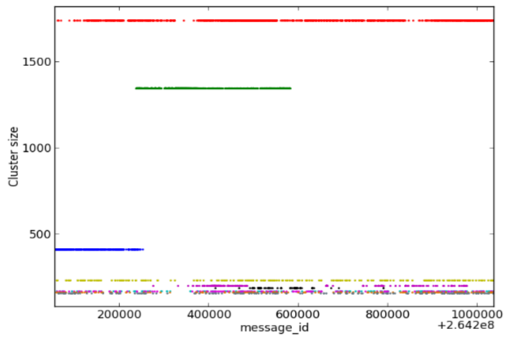


Figure 4 Spam Distribution based on Clusters for Simple Random 2% Sampling

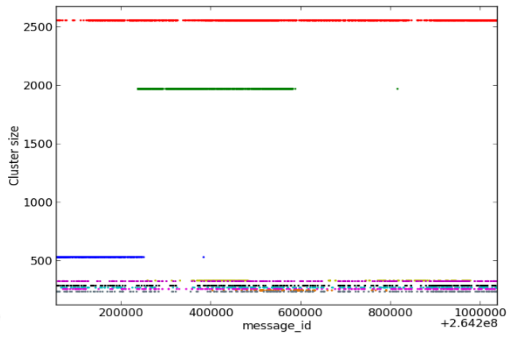


Figure 5 Spam Distribution based on Clusters for Step Sequence 2% Sampling

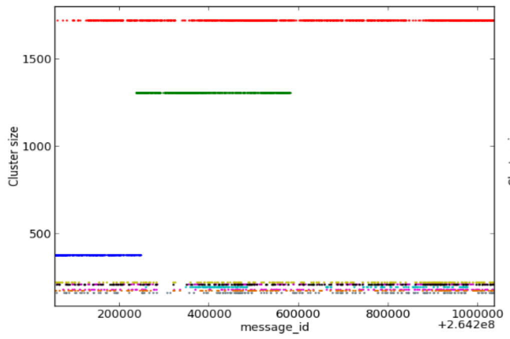


Figure 6 Spam Distribution based on Clusters for Stepping Random 2% Sampling

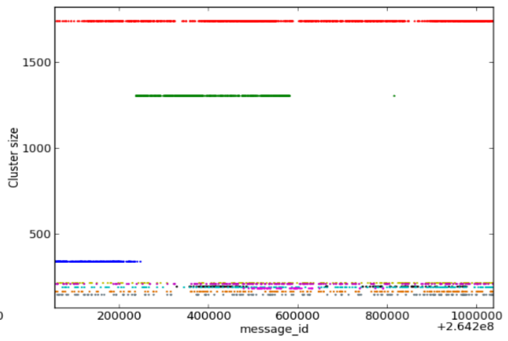


Figure 7 Spam Distribution based on Clusters for Monte Carlo 2% Sampling

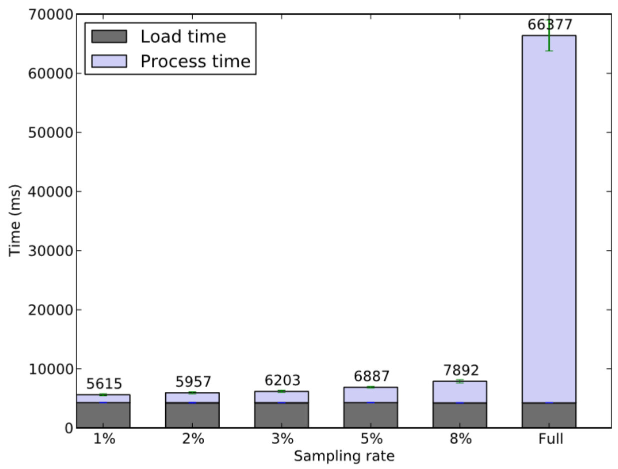


Figure 8 Timing Performance for Application Level Filtering

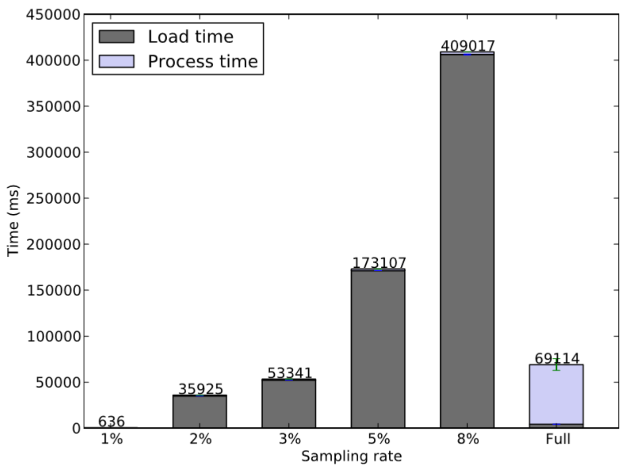


Figure 9 Timing Performance for Database Filtering using Naive SQL Query

5.3 Timing Performance

Here, we present the timing performance enhancement from mining and clustering the sampled data compared to using the whole dataset. The database was deployed on a x86 64-bit machine, using Intel 2.4 Ghz processor, with 6 processing cores and 12 GB RAM. Additionally, we executed the Java program to perform the clustering on the same machine. Hence, all timing measurements have been recorded based on the corresponding execution times. Figure 8 illustrates the timing measurements from the different sampling rates, including the timing for the complete data set.

The mean time required for loading the data from the database is 4261 milliseconds, and is depicted by the lower block in the timing bars in Figure 8. The loading time of the data is almost constant for all cases. This is because the query executed on the database from the application requests for the complete dataset for the specified day(s). Once the data is received, the application then performs an application level filtering of the data, by either selecting or discarding the item, based on the sampled indexes generated separately. Thus, given that the machine executing the program had sufficient main memory, the task of on-memory filtering of the data was performed within a very short time.

The interesting measurement to be noticed is the upper segment in Figure 8, which corresponds to the processing time required for each of the cases of reduced data size using varying sampling rates. Once the data have been loaded and sampled, the clustering algorithm (Wei 2010; Ying et al., 2010) creates the clusters based on the given data. It can be distinctively seen that the time required for the whole data set is very high, compared to the sampled data clustering. Additionally, the algorithm adapted from Chun Wei et. al.'s work is the simple and faster version, which still is significantly high compared to the measurements obtained for the sampled data. The increase in time required with increasing sampling rate is not exactly linear, but not quadratic either. Thus, the reduction in the amount of time to perform a whole data set clustering can be reduced by a factor greater than linear if a sampled data set is used.

6. SAMPLING OPTIMIZATION

For further research, we explored some strategies to optimize the process of sampling. In our opinion, the timing performance of sampling can be improved if we are able to perform the operation on the database engine. The following sections illustrate our process of investigation and the methods we adopted to fulfill the requirements.

6.1 Data Preprocessing

Given the huge number of spam emails gathered every day, reading the data items from the database required a significant amount of time. In the clustering implementation by Chun Wei et. al. (Wei et al., 2009), they performed a read operation on the whole data for a specific date. As a result, this incurred to a huge number of read operations on the database server.

We performed some initial data preprocessing to reduce the number of read operations while retrieving the data items from the database. We created a new table, namely ***daily_index***, with fields *receiving_date* and *message_id*. The table was populated using the minimum values for the *message_id* for each date from the spam table. With the ***daily_index*** table created, we can now easily retrieve the range of values for *message_id* for the given dates for which we will perform the clustering. For each sampling method, we initially provide the *message_id* range, get the sampled indexes, and subsequently, retrieve only the required data items from the database based on the desired sampling rate *r*. As a result of this operation, we are able to save (*n-(n*r/100)*) read operations from the database; where *n* is the total number of records for the given date.

6.2 Naïve SQL Query

The initial time measurements were taken based on an application level filtering for the sampling process. On the contrary, with the data pre-processing and the ***daily_index*** table created, we initially generated indexes for the sampled *message_id*s. Subsequently, we queried the database with a long matching clause of the sampled *message_id*s to retrieve the required rows. However, in this form of queries, we failed to improve the timing requirement. The size of the query was itself very large, and the database took a very long time to select and load the sampled records. The measurements from the naïve SQL query are illustrated in Figure 9. It can be seen clearly that even though the processing time is reduced, the sampling queries take an exceptionally long time to load the sampled data. Thus, as we failed to improve the performance using the naïve SQL query, we investigated further options to optimize the sampling process.

6.3 Cross-Product with Temporary Table

Next, we considered executing the query in a different fashion. In this approach, similar to the previous, we performed the sampling selection using the ***daily_index*** table. However, the next operation included creating a temporary table with only the selected *message_id*s. A query was then executed on the database to return the cross-product of the temporary table and the spam table. The execution of cross-product operation is optimized by the database itself, and therefore, the database is able to return the resulting records in split seconds. The timing measurements from using a temporary table and cross-product operation are shown in Figure 10.

It can be seen that the total time required for the sampled data is much lesser than the time required for the complete data set. As it was seen previously in Figure 9, the load times for the sampled records were significantly high compared to the full data retrieval. However, in this case, it can be seen from Figure 10 that the load times for sampled *message_id*s are around a few hundred milliseconds, which are much lesser compared to the full data. The maximum load time was required when we reached a sampling rate of 8%, which was still equal to the load time for the whole data set. If we compare our results from the initial timing measurements presented in Figure 8, it can be seen that the times for sampling rates 1%, 2%, 3%, and 5% are all much lesser in our optimized sampling operation. In the case of 8%, it is still lesser, but maybe comparable to the previously recorded measurements.

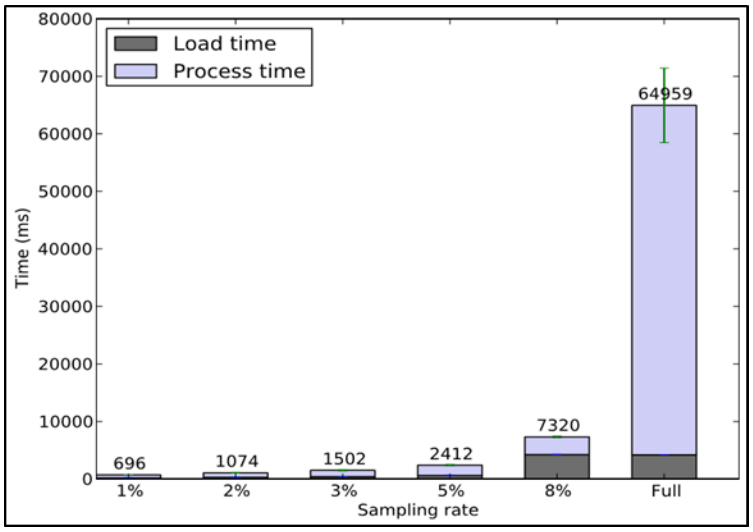


Figure 10 Timing Performance for Database Filtering using Temporary Table

Therefore, with the given results, we can argue that the proposed approach is significantly better than the original application layer filtering. We have successfully illustrated that the processing time for the sampled clustering using a temporary table is much better for reasonable sampling rates. Additionally, sampled clustering using this strategy reduces a lot of task load on the machine which executes the clustering algorithm. Even though we had both the program and the database on the same machine, it can be surely assumed that the database server is usually a separate machine with more processing power. Therefore, the described method of optimizing the process of sampling takes advantage of the processing power of the database engine, and keeps the machine running the clustering algorithm much lighter in its operation.

7. RELATED WORKS

Researchers have been working on interaction with large databases for a long time. Data mining and knowledge extraction technologies have been a rather new addition to the list of research works on large data sets. The clustering algorithm used here has been the ‘fast-n-dirty’ version of Wei's work (Wei 2010; Wei et al., 2009). The focus of this paper was to illustrate the efficiency which can be reached prior to the process of clustering, leading to a faster identification of the ‘hot zone’. Therefore, the algorithm for clustering is separate from the sampling process. As a result, any underlying algorithm for the sampling models will provide more efficient results with respect to time and space.

The performance of the clustering process and the quality of the resultant clusters depends on the corresponding clustering algorithms. In this paper, we have successfully illustrated that we are able to identify the prominent spam clusters from the sampled data, with radical improvements in timing performance for clustering algorithms. There are multiple clustering algorithms which explore the text-based patterns in spam emails (Kyriakopoulou and Kalamboukis 2008; Ramachandran et al., 2007; Sasaki and Shinnou 2005; Wei 2010; Wei et al., 2009), including clustering algorithms specifically applicable for large datasets (Ganti et al., 1999). Halkidi et al., proposed further techniques, which can be used to validate the clustering quality (2001). Therefore, given that we have proved sampling to be an effective data reduction process, our following research will focus on optimizing the clustering algorithms.

We have explored different strategies and related works on clustering mechanisms. The oldest centroid based clustering method is the k-means algorithm (Hartigan and Wong, 1979). Later, many optimized and efficient versions of the k-means algorithm have been proposed (Kanungo et al., 2002). One of the earliest works on modern clustering techniques was proposed by Koontz et al. (1975). They proposed a branch and bound clustering algorithm based on global combinatorial optimization. DBSCAN is a well-known density-based clustering algorithm. Arlia et al., proposed a method of parallelizing DBSCAN, which is suitable for high-dimensional data, and thus can be useful in implementing a suitable clustering algorithm for the huge number of spam emails (Arlia and Coppola, 2001). ST-DBSCAN is a different variation of DBSCAN, proposed by Birant et al. (2007), which performs the clustering based on identifying core objects, noise objects, and adjacent clusters. Ying et al., has already presented in (Ying et al., 2010) a variation of DBSCAN to successfully identify spam clusters. The proposed research aims for faster clustering results from spam emails. Henceforth, it can be suitably stated that, given the organization of the spam data mine, we will be able to preserve the results from these clustering algorithms, when compared to clustering based on sampled data.

There has been significant research on sampling methodologies so far. The random sampling with reservoir, proposed by Vitter (Vitter 1985), uses a non-replacing one pass sampler, requires constant space, and runs in O(n(1 + log(N/n))) time. These sampling models aim to introduce randomness in the sampled items. However, we are interested in identifying the most prominent clusters. The purpose is fulfilled using the proposed models and are shown to be effective in determining the 'hot zone' appropriately. Nagwani et al. (2010) proposed a weighted matching technique of attributes to measure attribute similarity of email content. The weights of the attributes are custom assigned and are then used to create the spam clusters. An algorithm for text clustering based on vector space is presented by Sasaki et al., in (Sasaki and Shinnou, 2005). The proposed algorithm creates disjoint clusters with the underlying spherical k-means algorithm to obtain centroid vectors of the spam clusters.

There are other works related to email filtering which can be related to analyzing the content of spam emails. An interesting approach for filtering spam emails based on behavioral blacklisting has been proposed by Ramachandran et al. (2007). The proposed method overcomes the problem of varying sender IP addresses by classifying sending patterns and behaviors of spammers, and subsequently enforcing blacklisting decisions. Thomas et al., presents an interesting approach for spam detection, which includes real-time web crawling of URLs, based on blacklists and whitelists (Thomas et al., 2011). All the approaches for clustering spam emails are suitable and will have varying results. These algorithms are typically applicable for spam filters, usually on web browsers and email clients. However, given the size of the dataset of the UAB Spam Data Mine (UAB-CIS, 2013), we suggest that the purpose of identifying the 'hot zone' by eCrime investigators and law enforcement authorities is better served by avoiding such fine-grained spam detection algorithms.

8. CONCLUSION

Spam campaigns and emails create a lot of hassle in today's world. A lot of people fall victims to such scams every day. Most spams are sent using malware bots, which are installed on affected PCs and spread around like a virus. The UAB Spam Data Mine collects such spam emails, and provides reports on ongoing spam campaigns. Clustering the spam data to categorize and identify the spammer has been implemented using the full dataset. In this paper, we presented different models for sampling the spam data, to be used as a tool for data reduction. Subsequently, the sampled data were utilized to create the clusters.

Our obtained results substantially prove that sampling the data and creating the clusters allow the investigators to interpret the same conclusions, as opposed to using the whole data set. As a result, we claim that it is much faster and efficient to perform the clusters after sampling the data, and thus identify the 'hot zone' within a significantly shorter period of time. We have provided extensive experimental results using actual spam data and investigated the distribution of spam in the data mine, which reinforced our claims of sampling being more effective given its purpose. Furthermore, we also presented an optimization strategy which utilizes the computational power of database engines to perform the sampling operation more efficiently, and thus promises faster results in terms of the time required.

ACKNOWLEDGEMENT

This research was supported by a Google Faculty Research Award, the Office of Naval Research Grant #N000141210217, the Department of Homeland Security Grant #FA8750-12-2- 0254, and by the National Science Foundation under Grant \#0937060 to the Computing Research Association for the CIFellows Project. We would like to thank Jason Britt and Gary Warner for providing the support for the UAB Spam Data Mine.

REFERENCES

1. Arlia, D. & Coppola, M. (2001). Experiments in parallel clustering with dbscan. Euro-Par 2001 Parallel Processing. Lecture Notes in Computer Science, 2150. Springer Berlin Heidelberg, 326-331.
2. Birant, D. & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, *60*(1), 208-221.
3. Caruana, G. & Li, M. (2008). A survey of emerging approaches to spam filtering. *ACM Computing Surveys*, *44*(2), 9:1-9:27.
4. Dagon, D., Gu, G., Lee, C., & Lee, W. (2007). A taxonomy of botnet structures. Proceedings of the 23rd Annual Computer Security Applications Conference. ACSAC 2007, 325-339.
5. Ganti, V., Ramakrishnan, R., Gehrke, J., & Powell, A. (1999). Clustering large datasets in arbitrary metric spaces. Proceedings of the 15th International Conference on Data Engineering (*ICDE 1999). IEEE Computer Society*, Washington, DC, USA.
6. Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, *17*, December, 2-3, 107-145.
7. Hammersley, J. M., Handscomb, D. C., & Weiss, G. (1965). Monte Carlo methods. *Physics Today*, *18*, 55.
8. Hartigan, J. A. & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society*. Series C (Applied Statistics) *28*(1), 100-108.
9. Jaccard, P. (1901). Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, *37*, 241-272.
10. Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., & Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 881-892.
11. Knuth, D. E. (2006). The art of computer programming. 4, fascicle 4, 1. print.. Generating all trees. Addison-Wesley.
12. Koontz, W. L. G., Narendra, P. M., & Fukunaga, K. (1975). A Branch and Bound Clustering Algorithm. *IEEE Transactions on Computers*, *24*(9), 908-915.
13. Kyriakopoulou, A. & Kalamboukis, T. (2008). Combining clustering with classification for spam detection in social bookmarking systems. Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Discovery Challenge, (ECML/PKDD RSDC 2008), 47-54.
14. Levchenko, K., Pitsillidis, A., Chachra, N., Enright, B., Halvorson, T., Kanich, C…Savage, S. (2011). Click trajectories: End-to-end analysis of the spam value chain. Proceedings of The IEEE Symposium on Security & Privacy, 431-446.
15. Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*. *10*(8 Feb), 707-710.
16. Matsumoto, M. & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS) - Special issue on uniform random number generation*, *8*(1 Jan), 3-30.
17. Moore, T., Clayton, R., & Anderson, R. (2009). The economics of online crime. *The Journal of Economic Perspectives*, *23*(3), 3-20.
18. Nagwani, N. K. & Bhansali, A. (2010). An Email Clustering Model Using Weighted Similarities between Emails Attributes. *International Journal of Research and Reviews in Computer Science* (IJRRCS), *1*, 2.
19. Nhung, N. P. & Phuong, T. M. (2007). An efficient method for filtering image-based spam e-mail. Proceedings of The 12th international conference on Computer analysis of images and patterns, (CAIP'07). Springer-Verlag, Berlin, Heidelberg, 945-953.
20. Ono, K., Kawaishi, I., & Kamon, T. (2007). Trend of Botnet Activities. Proceedings of the 41st Annual IEEE International Carnahan Conference on Security Technology, (ICCST) '07, 243-249.
21. Ramachandran, A., Feamster, N., & Vempala, S. (2007). Filtering spam with behavioral blacklisting. Proceedings of the 14th ACM Conference on Computer and Communications Security, (CCS) 2007. ACM, New York, NY, USA, 342-351.

22. Sasaki, M. & Shinnou, H. (2005). Spam detection using text clustering. Proceedings of the International Conference on Cyberworlds, *4*(4), 319.
23. Thomas, K., Grier, C., Ma , J., Paxson , V., & Song, D. (2011). Design and evaluation of a real-time url spam filtering service. Proceedings of the 2011 IEEE Symposium on Security and Privacy, (S&P 2011), IEEE, 447-462.
24. UAB-CIS. (2013). Department of CIS, University of Alabama at Birmingham, UAB Spam Data Mine. Retrieved from http://www.cis.uab.edu/UABSpamDataMine.
25. Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software* (TOMS), *11*(1 Mar), 37-57.
26. Wei, C. (2010). Clustering Spam Domains and Hosts: Anti-Spam Forensics with Data Mining. Doctoral thesis, University of Alabama at Birmingham.
27. Wei, C., Sprague, A., & Warner, G. (2009). Clustering malware-generated spam emails with a novel fuzzy string matching algorithm. Proceedings of the 2009 ACM symposium on Applied Computing, (SAC 2009), ACM, New York, NY, USA, 889-890.
28. Ying, W., Kai, Y., & Zhong, Jian Z. (2010). Using DBSCAN clustering algorithm in spam identifying. Proceedings of the 2nd International Conference on Education Technology and Computer. (ICETC) 2010, *1*, 398-402.