Consensus Clustering and Supervised Classification for Profiling Phishing Emails in Internet Commerce Security

R. Dazeley[1], J.L. Yearwood[1], B.H. Kang[2], A.V. Kelarev[1]

[1]Centre for Informatics and Applied Optimization
Graduate School of ITMS, University of Ballarat
P.O. Box 663, Ballarat, Victoria 3353, Australia
{*r.dazeley,j.yearwood,a.kelarev*}*@ballarat.edu.au*

[2]School of Computing and Information Systems
University of Tasmania, Private Bag 100
Hobart, Tasmania 7001, Australia
BHKang@utas.edu.au

Abstract. This article investigates internet commerce security applications of a novel combined method, which uses unsupervised consensus clustering algorithms in combination with supervised classification methods. First, a variety of independent clustering algorithms are applied to a randomized sample of data. Second, several consensus functions and sophisticated algorithms are used to combine these independent clusterings into one final consensus clustering. Third, the consensus clustering of the randomized sample is used as a training set to train several fast supervised classification algorithms. Finally, these fast classification algorithms are used to classify the whole large data set. One of the advantages of this approach is in its ability to facilitate the inclusion of contributions from domain experts in order to adjust the training set created by consensus clustering. We apply this approach to profiling phishing emails selected from a very large data set supplied by the industry partners of the Centre for Informatics and Applied Optimization. Our experiments compare the performance of several classification algorithms incorporated in this scheme.

1 Introduction

The applications of clustering techniques to profiling phishing emails and web sites is an important problem in internet commerce security, which has been actively investigated recently. Various clustering algorithms have been used in this context by many authors. To illustrate, here we refer to just a few recent articles on this topic [7, 15, 16, 18].

This paper investigates a novel combined method, which at the same time uses unsupervised consensus clustering algorithms as well as supervised classification algorithms. First, a variety of independent clustering algorithms are

applied to a randomized sample of the data. Second, several consensus functions and sophisticated algorithms are used to combine these independent clusterings into one consensus clustering. Third, the consensus clustering of the randomized sample is used as a training set to train several fast classification algorithms on the consensus clustering of the initial sample. Finally, these fast supervised classification algorithms are used to classify the whole large data set in order to obtain final clustering.

This approach makes it possible to apply slow and most reliable clustering methods at the initial stages to improve the accuracy. It increases the speed of processing the whole large data set by incorporating fast algorithms in the final stages. It facilitates the inclusion of contributions from the domain experts to adjust the initial training set created by consensus clustering algorithms.

Our experimental investigation applies this approach to profiling phishing emails selected from a very large data set supplied by the industry partners of the Centre for Informatics and Applied Optimization. Our experimental results compare the efficiency of performance of several classification algorithms incorporated in this approach.

The number k of clusters is chosen and fixed as an input parameter for our algorithms. The question of how to vary this number and choose the most appropriate one for any given application is not considered in the present article.

The outcomes obtained show that this method can be used as a novel way of combining several clustering techniques in order to classify very large data sets of phishing emails for subsequent forensic analysis based on the resulting individual clusters.

The paper is organised as follows. Section 2 is devoted to the preprocessing of data and extraction of features for clustering algorithms. Section 3 outlines unsupervised clustering algorithms applied to obtain an initial clustering ensemble for a small randomized sample of the data set. Section 4 describes consensus functions and heuristics used to combine the ensemble into one final consensus clustering. Section 5 deals with the supervised classification algorithms trained on the consensus clustering. Section 6 summarizes the experimental results comparing the efficiencies of several classification algorithms in this setting.

2 Feature Extraction

Many authors have concentrated on the applications of machine learning algorithms for classification and clustering of phishing emails, since phishing represents one of the most rapidly growing and changing areas of internet commerce security. Phishing usually involves acts of social engineering attempting to extract confidential details by sending emails with false explanations urging users to provide private information that will be used for identity theft. The users may be requested to reply to the email, or visit a bogus web site, where they are asked to enter personal details, such as credit card numbers, tax file numbers, bank account numbers and passwords. More comprehensive information concerning

phishing is presented, for example, by the Anti-Phishing Working Group [2] and OECD Task Force on Spam [19].

We have undertaken experimental investigation of this novel approach to clustering, outlined in Section 1, for a sample of 3276 emails randomly selected from a very large data set of phishing messages supplied by the industry partners of the Centre for Informatics and Applied Optimization. A flexible preprocessing and feature extraction system has been implemented in Python for the purposes of this investigation. It has been used to extract features concerning the content and structure of the emails, and hyperlinks embedded in the text.

Following [12], we used the *term frequency–inverse document frequency* word weights, or TF-IDF weights, as features for the clustering. These weights are defined using the following concepts and notation. Suppose that we are extracting features from a data set E, which consists of $|E|$ messages. For a word w and a message m, let $\mathrm{N}(w, m)$ be the number of times w occurs in m. Suppose that a collection $T = \{t_1, \ldots, t_k\}$ of terms $t_1, \ldots, t_k$ is being looked at. The *term frequency* of a word $w \in T$ in a message m is denoted by $\mathrm{TF}(w, m)$ and is defined as the number of times w occurs in m, normalized over the number of occurrencies of all terms in m:

$$\mathrm{TF}(w, m) = \frac{\mathrm{N}(w, m)}{\sum_{i=1}^{k} N(t_i, m)} \tag{1}$$

The *document frequency* of the word w is denoted by $\mathrm{DF}(w)$ and is defined as the number of messages in the given data set where the word w occurs at least once. The *inverse document frequency* is used to measure the significance of each term. It is denoted by $\mathrm{IDF}(w)$ and is defined by the following formula

$$\mathrm{IDF}(w) = \log\left(\frac{|E|}{\mathrm{DF}(w)}\right). \tag{2}$$

The *term frequency–inverse document frequency* of a word w in message m, or TF-IDF weight of w in m is defined by

$$\text{TF-IDF}(w, m) = \mathrm{TF}(w, m) \times \mathrm{IDF}(w, m). \tag{3}$$

We collected a set of words with highest TF-IDF scores in all messages of the data set. For each message, the TF-IDF scores of these words in the message were determined. These weights and additional features were assembled in a vector. In order to determine the TF-IDF scores we used Gensim, a Python and NumPy package for vector space modelling of text documents.

In addition we used the following features reflecting the syntactic structure of the messages:

- number of html tags in the message;
- number of links in the message;
- number of mismatched links, where the visible link is different from the hyperlink reference;
- number of scripts included in the message;

- number of tables in the message;
- number of embedded images;
- number of attachments to the message.

These features were assembled in an algebraic vector space model representing the data set. A number of independent initial clusterings were then obtained for the feature vectors of the messages in the sample using the following clustering algorithms.

3 Independent Initial Clusterings

The standard k-means clustering algorithm is described, for example, in [10], Section 3.3.2, and [24], Section 4.8. This algorithm randomly chooses k messages as centroids of clusters at the initialization stage. Every other message is allocated to the cluster of its nearest centroid. After that each iteration finds new centroids of all current clusters as a mean of all members of the cluster. This is equivalent to finding the point such that the sum of all distances from the new centroid to all other sequences in the cluster is minimal. Then the algorithm reallocates all points to the clusters of the new centroids. It proceeds iteratively until the centroids stabilize. The outcomes of the algorithm often depend on the initial selection of the very first centroids. We used the following two more advanced algorithms, which overcome this dependence.

First, we used the well-known *Global k-Means* algorithm, GKM, introduced in [17]. It overcomes the dependence on the initial choice of the centroids. It starts with just one centroid, which is taken as the mean of all points in the data set. Then the algorithm proceeds inductively. Suppose that i centroids have been found, for some $i < k$. Each of the given points in the data set is then chosen in turn and used as an $(i + 1)$-st initial centroid. For each of these choices of the additional centroids, the standard k-means algorithm is then run to partition the data set into $i + 1$ clusters. After that, all of the resulting partitions are compared with each other.

In order to evaluate each partition C, the algorithm uses the sum of squares of the distances from all points to the centroids of their clusters in the clustering C. This sum is denoted by $\mathrm{GKM}(C)$. The sum is taken over all clusters of the partition. Each cluster contributes the summand equal to the sum of all squared distances from all elements of the cluster to the centroid of the cluster. If a clustering $C = C_1 \dot\cup \ldots \dot\cup C_{i+1}$ is a disjoint union of the clusters $C_1, \ldots, C_{i+1}$, where each cluster C_j has a centroid m_j, then

$$\mathrm{GKM}(C) = \sum_{j=1}^{i+1} \sum_{x \in C_j} ||x - m_j||^2. \tag{4}$$

The partition which minimizes (4) is chosen as the best clustering with $i + 1$ clusters. The global k-means algorithm continues this process iteratively until it finds a partition into k clusters. Notice that the Modified Global k-means

algorithm, MGKM, developed in [3], can be used to increase the efficiency of the GKM algorithm for large data sets.

Second, we used a modification of the *Multiple Start k-Means* algorithm, MSKM, considered in [3], [9] and [11]. The standard MSKM clustering algorithm selects many random sets of initial centroids, runs the k-means algorithm for each of them, and chooses the partition minimizing the sum-of-squares objective function (4). The purpose of our investigation, however, is to find an optimal consensus among versatile clusterings. We have tried to include various different independent clusterings in the scheme. Since we have already included the global k-means algorithm minimizing the sum-of-squares (4), adding other algorithms concentrating on this objective function could skew the resulting outcome.

Instead, we modified the MSKM algorithm and used the following *Consensus Multiple Start k-Means* algorithm, CMSKM. It makes 50 random selections of the initial centroids, runs the standard k-means algorithm, and then finds an aggregated consensus clustering of the resulting 50 k-means clusterings. To find the consensus clustering we used the simplest cluster-based similarity partitioning algorithm, CSPA, described in [9]. It places two messages in the same cluster if they belong together to one cluster in the majority of the clusterings of the ensemble.

Third, we used a version of hierarchical agglomerative clustering algorithm known as the *Nearest Neighbour* clustering, NN, see [10], Section 3.3.7, [11] and [24], Section 4.7. It never merges large clusters, and only amalgamates separate messages, i.e. singleton clusters, to other clusters at each step. Given the number k of clusters to be found, it chooses k random messages as representatives of the clusters. For every other message m in the data set, it considers all messages which have already been assigned to the clusters and finds the nearest neighbour of m among these messages. The message m is then allocated to the cluster of its nearest neighbour. This continues until all messages are allocated to clusters. The outcome of the algorithm strongly depends on the initial random choice of the very first representatives. This is why it is very seldom used in this form.

In order to overcome the dependence on the initial random selection, as recommended in [9], we made 50 uniformly distributed selections of the initial representatives, run the nearest neighbour clustering algorithm for each of them, and then found a common consensus clustering of all the resulting clusterings, using the CSPA consensus aggregation algorithm again. Here we call the resulting procedure the *Consensus Multiple Start Nearest Neighbour* clustering algorithm, CMSNN.

Fourth, we looked at the *k-Committees* clustering method considered in [25] for a data set of DNA sequences. For a very small positive integer $r = 2, 3, ...$, it finds a very special set of r elements in each class, called the *committee of r representatives* of the class, or simply the *committee* of the class. The committee of a cluster C is defined as a set of r points $x_1, \ldots, x_r$ defined as a solution to the following optimization problem:

$$\text{minimize} \quad \max_{y \in C} \left(\min_{i=1,\ldots,r} ||x_i - y|| \right) \quad \text{subject to } \; x_1, \ldots, x_r \in C, \qquad (5)$$

see [25], Section 5.

Every new message is then allocated to the class of its nearest committee member, see [25]. The training stage of the k-committees algorithm is not scalable, and this is why it has not been used in practice. However, after the completion of the training, the algorithm runs fast. Our scheme makes it possible to apply this algorithm, since it has to be trained with a fairly small initial sample only.

In order to overcome the dependence of this algorithm on the initial choice of starting committees, we run it for 50 randomized selections of these representatives, and for small values of r from 2 to 6 representatives in each committee. Then we applied the CSPA consensus aggregation algorithm as above to combine the resulting ensemble into one clustering. In the present article this procedure is called the *Consensus Multiple Start k-Committees* algorithm, CMSKC.

4 Consensus Functions for Ensemble Clusterings

The process of finding the combined consensus clustering has also been divided into two substages. During the first substage several independent initial clusterings were ensembled using various consensus functions. This has produced a number of very similar consensus clusterings. During the second substage a fairly simple and fast consensus heuristic was used to combine them all into one common final consensus clustering.

During the first substage, given an ensemble of several independent clusterings on one and the same data set, consensus functions were applied to form new common consensus clusterings. Here we use the methods described, for example, in [5, 20, 23]. Let us denote the data set being investigated by

$$D = \{d_1, d_2, ..., d_n\}. \tag{6}$$

The clustering ensemble on this data set will be denoted by

$$C = \{C^{(1)}, C^{(2)}, \ldots, C^{(k)}\}, \tag{7}$$

where, for each clustering $C^{(i)}$, the whole set D is a disjoint union of the classes in this clustering so that

$$C^{(i)} = \{C_1^{(i)}, C_2^{(i)}, \ldots, C_{k_i}^{(i)}\}, \tag{8}$$

$$D = C_1^{(i)} \dot{\cup} C_2^{(i)} \dot{\cup} \ldots \dot{\cup} C_{k_i}^{(i)} \tag{9}$$

for all $i = 1, \ldots, k$.

Looking at the features described in Section 2, we applied several different clustering algorithms outlined in Section 3 to obtain initial clusterings. The following consensus functions and algorithms were invoked to combine the resulting

cluster ensemble into one consensus clustering:

ALCH – Average Link Consensus Heuristic,
CBGF – Cluster-Based Graph Formulation,
CCPH – Consensus Clustering Pivot Heuristic,
HBGF – Hybrid Bipartite Graph Formulation,
IBGF – Instance-Based Graph Formulation,
KMCF – k-Means Consensus Function.

All these consensus clustering algorithms have been compared for numerous data sets in [5, 6, 20]. Here we include only a brief summary of these methods, and refer to [1], [5] and [9] for more details.

Average Link Consensus Heuristic, ALCH, is an agglomerative clustering algorithm described in [9]. It starts off with a partition where every element belongs to its own separate singleton cluster. For each pair of elements i, j, the proportion p_{ij} of the initial consensus clusterings which cluster i and j in different clusters is determined. Then the algorithm finds two clusters with the smallest average distance and merges them together into one new cluster. This is repeated until the two closest clusters have average distance greater than the set threshold $\tau = 1/4$.

Cluster-Based Graph Formulation, CBGF, is a graph-based consensus function. It defines a complete weighted undirected graph on the set of vertices consisting of all the given clusters. The weight of each edge of this graph is determined by a measure of similarity of the clusters corresponding to the vertices. Namely, for two clusters C' and C'' the weight of the edge (C', C'') can be set equal to

$$w((C', C'')) = \frac{|C' \cap C''|}{|C' \cup C''|}, \tag{10}$$

known as the *Jaccard index* or *Jaccard similarity coefficient*, see [21], Chapter 2. In order to ensure that clusters which have a lot of elements in common are grouped together, the edges with lowest weights are then eliminated by applying a graph partitioning algorithm. Each element is then allocated to the new final cluster where it occurs most frequently.

Consensus Clustering Pivot Heuristic, CCPH, is an agglomerative clustering algorithm described in [1]. It chooses a pivot element i uniformly at random from the unclustered elements. It finds all elements j such that the proportion p_{ij} of the given initial clusterings in the ensemble, which cluster i and j in different clusters, does not exceed the threshold value $\tau = 1/2$, and places all of these elements j in the same cluster with i. This continues until all elements are clustered.

Hybrid Bipartite Graph Formulation, HBGF, is a consensus function based on a bipartite graph. It has two sets of vertices: clusters and elements of the data set. A cluster C and an element d are connected by an edge in this bipartite graph if

and only if d belongs to C. (The weights associated to these edges may have to be chosen as very large constants if the particular graph partitioning algorithm does not allow zero weights and can handle only complete graphs.) An appropriate graph partitioning algorithm is then applied to the whole bipartite graph, and the final clustering is determined by the way it partitions all elements of the data set. We used METIS graph partitioning software described in [14].

Instance-Based Graph Formulation, IBGF, is also a consensus function based on a complete undirected weighted graph. Vertices of the graph are all elements of the data set. The edge (d', d'') has weight given by the formula

$$w((d', d'')) = \sum_{i=1,\ldots,k;\ C_i(d')=C_i(d'')} 1/k,$$

where $C_i(x)$ stands for the cluster containing x in the i-th clustering. This means that $w((d', d''))$ is the proportion of clusterings where the clusters of d' and d'' coincide. Then IBGF applies an appropriate graph partitioning algorithm to divide the graph into classes. These classes determine clusters of the final consensus clustering.

k-Means Consensus Function, KMCF, relies on the standard k-means algorithm to produce final clustering. A complete explanation of this method is given in [23]. KMCF uses the set of all clusters in all clusterings of the ensemble as features for its feature vectors. For each element $d \in D$ and each cluster $C_j^{(i)}$, the $C_j^{(i)}$-th component of the feature vector of d is set to 1 if d belongs to $C_j^{(i)}$, and it is set to 0 otherwise. The standard k-means clustering algorithm is then used to cluster this set of feature vectors in order to find the consensus clustering.

During the second substage all the resulting consensus clusterings described above have been combined into one common consensus clustering using a very simple Majority Rule heuristic described in [9]. It is also known as the quote rule, see [8]. The Majority Rule is an agglomerative clustering algorithm, which starts with a partition where every element belongs to a separate singleton cluster. For each pair of elements i and j it computes the proportion p_{ij} of the initial consensus clusterings which cluster i and j in different clusters. If p_{ij} is less than a threshold value τ, then the current clusters containing i and j are combined together into one cluster. In our problem we used τ equal to the half of the total number of the consensus clusterings being combined.

5 Supervised Classification Algorithms

The resulting consensus clustering described in Section 4 was used to train the following fast supervised classification algorithms.

First, we used the k-means classification algorithm considered, for example, in [25] for a data set of DNA sequences. It finds the centroids of all clusters in the training set, and then allocates every new message to the cluster of its nearest

centroid (see [4], Chapter 4 and Section 10.4.3, [24], Chapter 4, and [22]). We have incorporated this method into our scheme, because it is very fast.

Second, we used the simplest and fastest nearest neighbour classification algorithm. It utilizes the clusters of the training set, called the *prototypes* or *exemplars*, see [4], Chaper 4, and [24], Section 6.4. The algorithm allocates every new message to the class of its nearest exemplar.

Finally, we used the k-committees classification algorithm considered in [25] for a data set of DNA sequences, see also [13]. For a very small positive integer r, it also finds a *committee* of the class. The committee of C is again defined as a set of points $x_1, \ldots, x_r$, which are found as a solution to the optimization problem (5). The committees have to be found only for the small training set created by the consensus clustering. Training stage of the k-committees algorithm is not scalable. In our scheme the algorithm only has to be trained on the relatively small training set, and this is why it can be incorporated in the scheme. In order to classify new messages after the training, the algorithm allocates every new message to the class of its nearest committee member, and can be executed very fast, since the number of the representatives in each committee is a very small and fixed positive integer.

6 Experimental Results

We have carried out experimental investigation of the novel approach to clustering for a sample of 3276 emails randomly selected from a very large data set of emails supplied by the industry partners of the Centre for Informatics and Applied Optimization. All of these emails have already been classified as phishing messages by the information security group of our industry partners. Many of these emails contain both text and hyperlinks and include HTML script, tables and images.

A flexible preprocessing and feature extraction system has been implemented in Python for the purposes of this investigation. It has been used to extract features concerning the content and structure of the emails, and hyperlinks embedded in the text as described in Section 2.

First, we found combined consensus clustering for the whole data set, following the procedure described in Section 4. Second, this clustering was used as a benchmark to determine the accuracy of the performance of several classification algorithms incorporated into the scheme. We used ten times tenfold cross validation to evaluate the accuracy of our multistage scheme. Each of the ten times, the data set was divides into ten equal parts, nine parts were used as a training set, and one part was used as a testing set. We run the combined consensus clustering procedure on the training set to prepare input for training as described in Section 4. After that the supervised classification algorithms described in Section 5 were trained on the training set obtained. Our experimental results compare the efficiency of the performance of these classification algorithms presented in Section 5 after their training on the initial consensus clustering data.

We used ten times tenfold cross validation to evaluate the accuracy of these algorithms. The accuracy of their performance was then evaluated in comparison with the combined overall total consensus clustering obtained previously. It is defined as the percentage of the messages in the test set which are classified correctly. This was repeated ten times, and the average accuracy was then calculated for each algorithm. The results of our experiments are summarized in Table 1.

Accuracy of classification algorithms w.r.t. consensus clustering				
Algorithm	Number of clusters			
	5	10	15	20
k-means	66.50	60.07	57.69	52.28
k-committees with $r = 2$	76.11	72.56	68.12	61.12
$r = 3$	83.23	77.44	73.24	70.58
$r = 4$	88.05	83.66	80.05	74.84
$r = 5$	91.35	87.86	80.07	78.49
$r = 6$	93.55	86.21	84.98	78.44
Nearest Neighbour	88.75	81.47	77.25	73.45

Table 1. Ten times tenfold cross validation.

7 Acknowledgements

The authors are grateful to three referees for comments and corrections that have helped to improve the text of this article.

8 Conclusion

This article looked at a novel method for profiling phishing emails. First, a multitude of independent clustering algorithms were used to a randomized sample of the messages. Second, several consensus functions and sophisticated algorithms were applied to combine these independent clusterings into one final consensus clustering. Third, several fast supervised classification algorithms were trained on the consensus clustering of the randomized sample. Finally, these fast classification algorithms classified the whole data set. This approach facilitates the inclusion of contributions from domain experts via adjusting the training set created by consensus clustering. We applied this approach to a set of phishing emails provided by the industry partners of the Centre for Informatics and Applied Optimization. The experimental results show that the nearest neighbour and the k-committees algorithms achieve much better accuracy in this scheme,

compared with the k-means algorithm, and the k-committees algorithm outperforms the nearest neighbour on the average. It has also been demonstrated that the scheme can be used in practice. If required, then it has the potential to facilitate the inclusion of contributions form domain experts for adjusting the training set produced by consensus clustering algorithms. Another advantage of our approach is in its ability to combine highly accurate consensus clustering techniques with fast and simple classification algorithms in one scheme.

References

1. Ailon, N., Charikar, M. and Newman, A. (2005) *Aggregating inconsistent information: ranking and clustering*. Proc. 37th annual ACM Symposium on Theory of Computing, 684–693.
2. Anti-Phishing Working Group (2009). Retrieved April 2010, http://apwg.org/
3. Bagirov, A.M. (2008) *Modified global k-means algorithm for minimum sum-of-squares clustering problems*, Pattern Recognition 41(2008), 3192–3199.
4. Duda, R.O., Hart, P.E. and Stork, D.G. (2001). "Pattern Classification", 2nd Edition, Wiley-Interscience.
5. Fern, X.Z. and Brodley, C.E. (2004). *Cluster ensembles for high dimensional clustering: an empirical study*. J. Machine Learning Research.
6. Fern, X.Z. and Brodley, C.E. (2004). *Solving cluster ensemble problems by bipartite graph partitioning*. Proc. 21st Internat. Conference on Machine Learning (Banff, Alberta, Canada, July 04 - 08, 2004). ICML'04, Vol. 69. ACM, New York, NY, 36.
7. Fette, I., Sadeh, N. and Tomasic, A. (2007). *Learning to detect phishing emails*. Proc. 16th Internat. Conf. on the World Wide Web, WWW'07, ACM Press, New York, NY, USA, 649–656.
8. Filkov, V. and Skiena, S. (2004). *Heterogeneous data integration with the consensus clustering formalism*. Proc. of Data Integration in the Life Sciences, pp. 110–123.
9. Goder, A. and Filkov, V. (2008). *Consensus clustering algorithms: comparison and refinement*. Proc. Tenth SIAM Workshop on Algorithm Engineering and Experiments, ALENEX 2008, 109–117.
10. Jain, A.K. and Dubes, R.C. (1988) "Algorithms for Clustering Data", Prentice Hall.
11. Jain, A.K., Murty, M.N. and Flynn, P.J. (1999). *Data clustering: a review*. ACM Computing Surveys, 31(3), 264–323.
12. Joachims, T. (1997). *A probabilistic analysis of the Rocchio algorithm with TF-IDF for text categorization*. Proc. 14th Internat. Conf. Machine Learning, 143–151.
13. Kang, B.H., Kelarev, A.V., Sale, A.H.J. and Williams, R.N. (2006). *A new model for classifying DNA code inspired by neural networks and FSA*. Pacific Knowledge Acquisition Workshop, PKAW2006. (Guilin, China, 7-8 August 2006). Lect. Notes Computer Science 4303, 187–198.
14. Karypis, G. and Kumar, V. (1998). *METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices*, Technical Report, University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Centre, Minneapolis.
15. Layton, R. and Watters, P. (2009). *Determining provenance in phishing websites using automated conceptual analysis*. Proc. 4th annual APWG eCrime Researchers Summit, Tacoma, WA.

16. Layton, R., Brown, S. and Watters, P. (2009). *Using differencing to increase distinctiveness for phishing website clustering*. Proc. Cybercrime and Trustworthy Computing Workshop, CTC-2009, Brisbane, Australia.
17. Likas, A., Vlassis, N. and Verbeek, J.J. (2003). *The global k-means clustering algorithm*. Pattern Recognition 36, 451–461.
18. Ma, L., Yearwood, J. and Watters, P.A. (2009). Establishing phishing provenance using orthographic features, APWG E-crime Research Summit.
19. OECD Task Force on Spam (2009). OECD Anti-Spam Toolkit and its Annexes. Retrieved April 2010, http://www.oecd-antispam.org/
20. Strehl, A. and Ghosh, J. (2002). *Cluster ensembles - a knowledge reuse framework for combining multiple partitions*. J. Machine Learning Research 3, 583–617.
21. Tan, P.N., Steinbach, M. and Kumar, V. (2005). "Introduction to Data Mining", Addison Wesley.
22. Theodoridis, S. and Koutroumbas, K. (2008). "Pattern Recognition", 4th Edition, Academic Press.
23. Topchy, A., Jain, A.K. and Punch, W. (2003). *Combining multiple weak clusterings*. Proc. IEEE Internat. Conf. on Data Mining, pp. 331–338.
24. Witten, I.H. and Frank, E. (2005). "Data Mining: Practical Machine Learning Tools and Techniques", Elsevier/Morgan Kaufman, Amsterdam.
25. Yearwood, J.L., Kang, B.H. and Kelarev, A.V. (2008) *Experimental investigation of classification algorithms for ITS dataset*. PKAW 2008, Pacific Rim Knowledge Acquisition Workshop, (Hanoi, Vietnam, 15 - 16 December 2008), 262–272.