EURASIP Journal on Information Security
a SpringerOpen Journal

RESEARCH Open Access

phishGILLNET—phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training

Venkatesh Ramanathan* and Harry Wechsler

Abstract

Identity theft is one of the most profitable crimes committed by felons. In the cyber space, this is commonly achieved using phishing. We propose here robust server side methodology to detect phishing attacks, called phishGILLNET, which incorporates the power of natural language processing and machine learning techniques. phishGILLNET is a multi-layered approach to detect phishing attacks. The first layer (phishGILLNET1) employs Probabilistic Latent Semantic Analysis (PLSA) to build a topic model. The topic model handles synonym (multiple words with similar meaning), polysemy (words with multiple meanings), and other linguistic variations found in phishing. Intentional misspelled words found in phishing are handled using Levenshtein editing and Google APIs for correction. Based on term document frequency matrix as input PLSA finds phishing and non-phishing topics using tempered expectation maximization. The performance of phishGILLNET1 is evaluated using PLSA fold in technique and the classification is achieved using Fisher similarity. The second layer of phishGILLNET (phishGILLNET2) employs AdaBoost to build a robust classifier. Using probability distributions of the best PLSA topics as features the classifier is built using AdaBoost. The third layer (phishGILLNET3) further expands phishGILLNET2 by building a classifier from labeled and unlabeled examples by employing Co-Training. Experiments were conducted using one of the largest public corpus of email data containing 400,000 emails. Results show that phishGILLNET3 outperforms state of the art phishing detection methods and achieves *F*-measure of 100%. Moreover, phishGILLNET3 requires only a small percentage (10%) of data be annotated thus saving significant time, labor, and avoiding errors incurred in human annotation.

Keywords: identity theft, machine learning, natural language processing, phishing, probabilistic latent semantic analysis, boosting, co-training

1 Introduction

Stealing a person's identity is one of the most profitable crimes committed by criminals. Among 1.3 million complaints received by the Federal Trade Commission in 2009, identity theft ranked first and accounted for 21% of the complaints costing consumers over 1.7 billion US dollars [1]. Identity theft has been around for many years while the means of committing it has changed with technology. The traditional way criminals steal a person's identity is by killing the individual. Another way to steal identity is using phone scams, where in, criminals inform the person that they have won a sweepstake, and convince the user to reveal some personal information to claim the money. The more popular method of identity theft that is prevalent even today is called Dumpster Diving. When people discard letters, financial records, and other personal information in the garbage dump without shredding, criminals scavenge those dumps looking for sensitive information such as credit card, bank account social security numbers, and use that information to commit crimes.

With the advent of Internet, the most popular way to steal identity is through "phishing". Like in traditional fishing where fishermen trolls the river in a boat to catch fish, in "phishing", attackers trolls the Internet using email message with convincing content as baits to steal users personal information. The email directs the user

* Correspondence: vramanat@gmu.edu
Department of Computer Science, George Mason University, Fairfax, VA 22030, USA

via a hyperlink to a website owned by criminals that looks very similar to a legitimate website. The user will then be asked to enter personal and financial information either to update existing information or to purchase a product. In reality, this lets the criminal to have access to that valuable information which they then use to commit fraud or to sell it to a bidder. Phishers can also trick users into downloading malicious codes or malware after they click on a link embedded in the email. This is a useful tool in crimes like economic espionage where sensitive internal communications can be accessed and trade secrets stolen. Phishing has been around since 1996 but has become more common and more sophisticated. Recent phishing attack on the Gmail system stole emails of US government officials, contractors, and military personnel [2].

Considerable research has been done towards protecting users from phishing attacks. They include firewalls, black listing certain domains and Internet protocol (IP) addresses, spam filtering techniques, client side toolbars, and user education. Each of these existing techniques has some advantages and some disadvantages. For example, existing filters have misclassification rates, the blacklist approach is harder to maintain with every expanding IP address/domain space, while the user ignores client side toolbar warnings.

The main contribution of this research is a multi-layered phishing detection method using previously developed modeling techniques that includes topic modeling technique Probabilistic Latent Semantic Analysis (PLSA), classifier ensemble technique AdaBoost and Co-Training algorithm that employs labeled and unlabeled data. The main goal of our novel approach is to detect phishing before it gets to the user. Towards that goal, we have developed the detection method, called phishGILLNET, by incorporating the power of natural language processing techniques. Similar to a "gillnet" that catches fish by its gill thus preventing its movement once caught, phishGILLNET tries to catch phishing attacks by the tone, wordings, and other linguistic variations in the content. By serving as a server side filter, phishGILLNET prevents movement of a phish towards the end user. The first layer of phishGILLNET (phishGILLNET1) employs PLSA to build a topic model and uses a topic level similarity function for classification. Unlike earlier approach that employed topic models, our model employed editing function and dictionary lookups to specifically account for intentionally misspelled words in phishing emails. The second layer of phishGILLNET (phishGILLNET2) employs classifier ensemble technique AdaBoost and topic probabilities as features to build a robust classifier using several base learners. To further expand phishGILLNET to handle labeled and unlabeled email data, the third layer (phishGILLNET3) employs Co-Training to build a classifier using topic distributions as features and the best classification technique obtained in the second layer. To the best of the authors' knowledge, this is the first attempt that demonstrates the power of topic model using Co-Training for phishing detection. The size of the corpus we employed is significantly larger (approximately 400,000) than that employed by authors of the Co-Training technique (few thousands) as well as by earlier researchers. Thus, our research is an additional proof of concept of the Co-Training algorithm in employing unlabeled data.

This article is organized as follows. We first review the state-of-the-art protection techniques and present their advantages and disadvantages (see Section 2). The multi-layered phishing detection method phishGILLNET is presented in Section 3. The modeling techniques employed by phishGILLNET namely PLSA, AdaBoost, and Co-Training are described in Sections 4, 5, and 6, respectively. The experimental design is presented in Section 7. The architectural components and results obtained on the public corpus for each layer of phishGILLNET, namely, phishGILLNET1, phishGILLNET2, and phishGILLNET3, are presented in Sections 8, 9, and 10, respectively. The performance comparison with the state-of-the-art tools is presented in Section 11. This article concludes with a discussion of the developed methodology and suggestions for future research in Section 12.

2 Background

The primary motivation for attackers using phishing is to steal identity from users. Several techniques have been developed to protect users from phishing attacks. The protection strategies are classified according to where in the attack flow that strategy belongs (see Figure 1). In Figure 1, the protection techniques are numbered 1-6 and shaded in grey. phishGILLNET is a server-side filter/classifier (numbered 3 in Figure 1). Non-shaded ones are the main components in the data flow. Some of the detection tools and their advantages and disadvantages are summarized in Table 1.

Network Level Protection

The network level protection is typically achieved by blocking a range of IP addresses or a set of domains from entering the network. DNSBL [3] is a database widely used for this purpose by several Internet service providers. This list is updated with new addresses, after observing for a period of time abusive behavior. Hence, this approach is reactive. Attackers evade this protection technique by hijacking legitimate user's PC and constantly moving from one IP to another IP address. Snort [4] is an open source software that is employed at network level. Rules to enforce protection must constantly be manually updated.

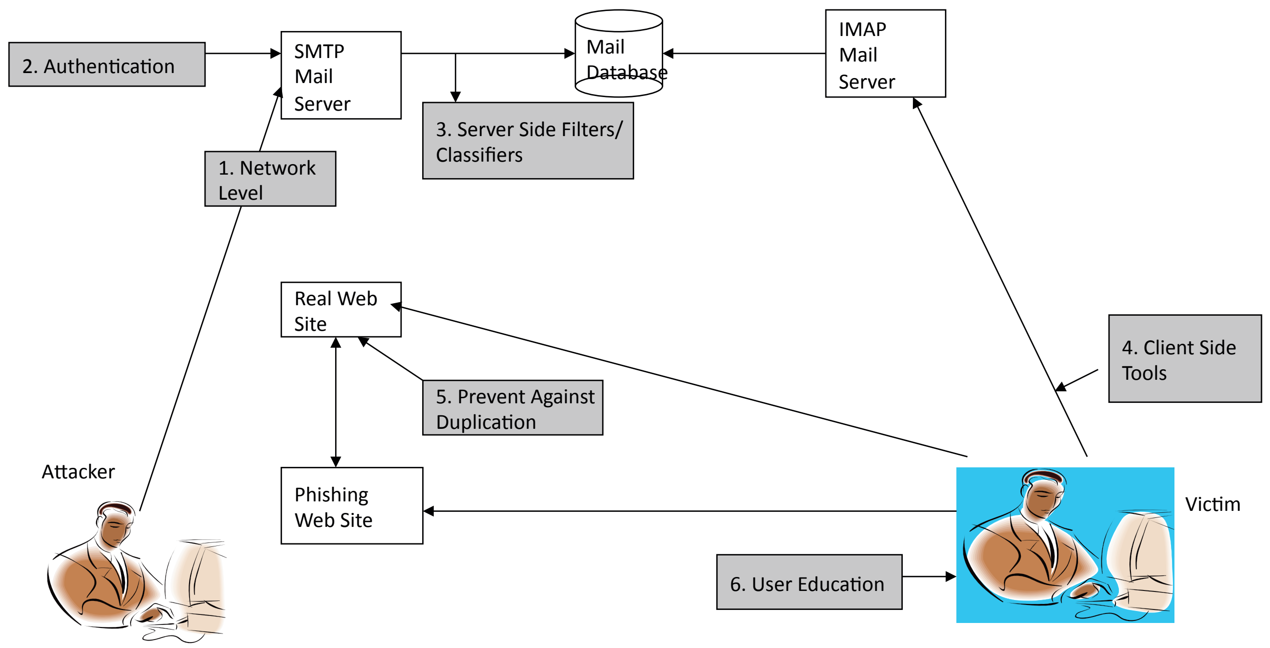

Figure 1 Phishing protection techniques.

Kim and Huh [5] compared four different classification techniques to detect DNS-poisoning-based phishing attacks using routing information gathered over 1-week period. Authors observed that k-nearest neighbor algorithm achieved best results with a false positive rate of 0.7% and true positive rate of 99.4%.

Authentication

There are two levels of authentication: user level and domain level. Typically, the email service provider authenticates user, before he or she sends an email (user level). The domain level authentication is performed in the provider-provider communication (one mail server

Table 1 Phishing detection tools

Tool	Type	Description	Advantages	Disadvantages
Snort [4]	Network Level	Heuristic/rule engine	Good at detecting level attacks	Rules require manual adjustments. Does not look at content.
SpamAssassin [9]	Server Side Filter	Heuristic engine that uses email specific features	Good at detecting email header spoofing.	High false positives
PILFER [10]	Server Side Filter	Utilize 10 features extracted from email to classify	Better performance than SpamAssasin.	Did not use content from body of the email. Susceptible to short lived phish domains.
SpoofGuard [30]	Client Side Tool	Plug-in to a browser	Warns user if link points to phish site.	Users do not pay attention to warnings. Not all email clients are browser based.
CatchingPhish [31]	Client Side Tool	Detects fake website based on rendered images	Browser independent. Good results on small data sets.	Processing time is high. Susceptible to screen resolution.
CallingID [32], CloudMark [33], Netcraft [34], and FirePhish [35]	Client Side Tool	Utilizes blacklist of domains.	Good for domains that employ domain level authentication.	Phish domains are short lived. Does not look at email content.
eBay Account Guard[36]	Client Side Tool	Utilizes blacklist of eBay URLs	Protects eBay users.	Specific website tool.
IE Phishing Filter[37]	Client Side Tool	Records specific user website visiting patterns.	Adapts to user website visit pattern.	Works only on internet explorer.

to the other mail server). The user level authentication is performed using password as credentials. The password authentication can easily be cracked as evidenced by surge in phishing attacks. Microsoft has developed a technology called Sender ID [6] while Yahoo has a similar technology called DomainKey [7]. Both these techniques perform domain level authentication. In order for these domain level techniques to work, providers on the sender and the recipient side must implement the same technology. Due to lack of agreement between email providers, this technology is not that prevalent.

Server Side Filters and Classifiers

Server side filters and classifiers typically extract features from the email and train a classifier to classify phishing email versus non-phishing email. Classifiers can be trained directly on various features extracted from the data or by applying dimensionality reduction techniques before training the classifier. Kim et al. [8] applied three dimensionality reduction methods, namely, Centroid, Orthogonal Centroid, and Linear Discriminant Analysis and tested their effectiveness on three different classifiers: Support Vector Machines, k-Nearest Neighbor, and Centroid-based classification. Authors concluded that dimension reduction techniques achieve high efficiency without sacrificing prediction accuracy. SpamAssassin [9] is a widely used host-level filter. This is a rule-based filter that requires constantly changing for the rule to be effective. Attackers figure out the rule being employed and bypass these filters by appropriately constructing the email. PILFER [10] is another email classifier that is trained using ten features extracted from email data. Both these filters have high misclassification rates. Abu-Nimeh et al. [11] presented a comparative evaluation of classification techniques such as Logistic Regression, Bayesian Adaptive Regression Trees, Support Vector Machines, Random Forests, and Neural Network. Authors trained the classifier using 43 features on a private ham email and public phishing email data and showed that random forest outperformed other classifiers when weighted equally but resulted in worst false positive rate. Neural network had the highest Area Under ROC Curve. Later work by Abu-Nimeh et al. [12] developed a method to detect phishing using Bayesian Additive Regression Trees and obtained better prediction than their earlier work. Miyamoto et al. [13] did a similar comparison of machine learning techniques for phishing website detection using about 3,000 website data. They obtained F-measure of 0.85 using AdaBoost. Toolan and Carthy [14] classified emails using C5.0 algorithm and ensemble of different classifiers. Authors obtained an F-measure of 99.31% using the publicly available dataset (PhishingCorpus and SpamAssassin) of 8 K emails. Gansterer and Pölz [15] developed a feature-based classifier for ternary classification, spam versus phish versus good. Authors utilized 11,000 phishing emails from a proprietary data source and publicly available TREC corpus for good and spam and obtained a classification accuracy of 97%. Bergholz et al. [16] trained a classifier using features obtained using Dynamic Markov Chain and Class-Topic Models. Authors obtained results better than PILFER on the same public corpus and showed effectiveness of topic features. Later work by Bergholz et al. [17,18] included additional features such as identification of hidden salting, embedded logos and external links and evaluated on a proprietary real life data from a commercial internet provider of size 40 K. Toolan and Carthy [19] proposed and ranked 40 different features using the information gain criteria. Khonji et al. [20] did an evaluation of feature selection algorithms and feature subset search methods on the same public corpus that most of the other research has been conducted. The study showed feature subset of 21 heuristic features yielded F-measure of 99.39%. Al-Momani et al. [21] achieved classification accuracy of 99.7% by applying a clustering algorithm for phishing detection while Zhan and Thomas [22] obtained a true positive rate of 99% by applying Stochastic Learning Weak Estimation approach. Yearwood et al. [23] obtained profiles of phishing activity by solving the problem using a multi-class classification problem utilizing features extracted from URLs in the emails. This study is closely related to Bergholz et al. [16,17], in the sense that, we use topic model PLSA (as compared to CLTOM) for phishing detection. However, our topic model is built to account for intentional misspelling and uses part-of-speech (verbs, nouns, adjectives, and adverbs) to build the model. We also show the effectiveness of our method on a large corpus of unlabeled data using Co-Training.

Several research has been done for phishing website detection. Xiang et al. [24] proposed a layered anti-phishing approach for detecting phishing web sites. Authors used a comprehensive feature-based detection algorithm to detect and filter out non-login form web pages and achieved 92% true positive rate and 0.4% false positive rate. Khonji et al. [25] proposed a technique for detecting phishing website by lexically analyzing URL tokens. Authors evaluated 70 K phishing URLs and obtained classification accuracy of 97%. Zhang et al. [26] proposed a text classifier, image classifier, and an algorithm that fused the two-classifier results to detect phishing web page detection and they concluded that fusion outperformed the performance of individual classifiers. Hsu et al. [27] proposed a solution for phish URL detection using suffix tree clustering methodology while Khonji et al. [28] proposed a heuristic solution. Wenyin et al. [29] developed an approach to detect phishing target from the content of the webpage. Most of the above research is limited to website detection; however, we propose a generic content-based approach that can be applied to email, web pages, blogs, and social networking posts.

Client Side Tools

Tools that operate on the client side (i.e., user's machine) include user profile filters and browser-based toolbars. SpoofGuard [30], CatchingPhish [31], CallingID [32], CloudMark [33], NetCraft [34], FirePhish [35], eBay toolbar [36], and IE Phishing Filter [37] are some of the client side tools. User profile filters observe user's website visiting pattern and maintain a list of URLs in local database. When a user visit's a URL that is different from his/her website visits, it warns the user with a dialog. Toolbars are built and trained using the typical pattern of phishing website URLs. Some patterns of phish website URLs include IP address in the URL, long URLs, many dots in the URL, etc. This technique is very susceptible to technology changes (such as IPV4 versus IPV6, tiny URLs) and hence it is not robust. Moreover, most users do not pay attention to the warning dialogs and hence it is not an effective protection technique. Abu-Nimeh and Nair [38] presented a new attack using DNS poisoning that bypass the client side toolbars. Their evaluation of seven tools concluded that none of them were able to detect the attack there by making these tools ineffective. Jain and Richariya [39] implemented a prototype web browser to detect phishing URLs. Authors did not compare their implementation with other browser-based tools and hence the effectiveness of the tool is not clear. Lin et al. [40] evaluated domain highlighting, the approach where browser highlights the domain name in the address bar, and concluded that this approach cannot be relied upon solely to detect and prevent phishing attacks. Chen et al. [41] presented a scientific assessment of user interface design elements such as font type, color, message placement, icon type, etc., used in various tools and concluded that existing tools fail to consider preference of the user while displaying warning and errors. Author's findings conclude that users prefer customization and personalization of these tools. Felt and Wagner [42] examined the threat of phishing on mobile devices. Authors analyzed 100 mobile applications and 85 web sites and found that attackers can spoof mobile web site. Authors found that Android and Apple-sponsored sites are top phish targets.

Prevent Against Duplication

This technique involves making the legitimate website harder to reproduce. In all legitimate websites, the login page is not protected. Hence, an attacker can easily copy the code, styles, graphics, and HTML to create a fake website. Hence, a protection approach could be to make this copy harder. There is no earlier work done in this area and hence it should be subject for future research.

User Education

This involves educating the user about phishing attacks and pattern of phish email. The basic mode of educating user is posting help pages in websites and warning the user about phishing. MailFrontier [43] has setup a website containing screenshots of several phish emails. Robila and Ragucci [44] evaluated the effect of user education in differentiating phishing and good emails. The authors presented a lecture on how to identify phishing emails and the harm of falling for a phish in an introduction to computing class. At the end of the lecture, the authors presented student with both phishing and good emails. Students were then asked to identify the email type. The study concluded that students identified phishing emails correctly after the lecture. Students also acknowledged the usefulness of the lecture. Similar study was also conducted at the Indiana University [45]. Arachchilage and Cole [46] designed an educational mobile game for home computers to protect users from phishing attacks. The game was designed to educate users to recognize phishing URLs. Authors developed a prototype simulator using Google App Inventor Emulator. Tseng et al. [47] also designed a game to educate users about phishing based on the content of the website. Moore and Clayton [48] conducted a study of how attackers discover potential hosts for phishing websites and concluded that search engine as one primary source. Authors of the study concluded public disclosure of phishing sites, such as the one done by phishtank.com, significantly reduces host compromise by attackers.

Existing protection techniques are ineffective in stopping the phishing attacks from reaching the end user. Network level protection using domain and IP address blacklisting require periodic updates and are reactive in nature as list can be updated only after observing abuse pattern for some time period. Moreover, attackers can compromise legitimate user's machine to conduct phishing attacks and hence blacklisting may block legitimate user from using the web. Existing server side filters and classifiers result in misclassification and use feature sets that are susceptible to technology changes. The classifiers that use content for attack detection do not consider intentionally misspelled, conjoined, and disjointed words. Attackers make subtle changes to the text of the email by using different words at different times and by using misspelled words to avoid detection by filters that require an exact word match. Thus, these filters often fail to detect phishing emails. Client side tools and filters expose the user one step closer to the attack. As users do not pay attention to warning dialogs, they end up falling for phishing attacks. The goal of this research is to stop the attack before it reaches the user. This is accomplished by building a robust multi-layered content-driven phishing detection methodology, phishGILLNET, which is described in Section 3.

3 phishGILLNET methodology

A schematic representation of phishGILLNET is shown in Figure 2. Gillnetting is a common fishing method used by fishermen in the ocean and in some freshwater areas [49]. A "gillnet", as the name implies, is a net that catches a fish by its "gill". It is a layer of netting hung vertically in the water by a float line on the top and a weighted line at the bottom. The mesh size, depth, and length of gillnet are determined by the species of fish that fishermen is trying to catch. The net allows the head of the fish to pass through but not its body. When the fish attempts to pass through, it gets stuck in the net by its gill and could neither move forward nor backward. Just like a gillnet is used to catch a fish by its gill, phishGILLNET is used to catch phishing attacks based on the linguistic variation in the content.

phishGILLNET is a multi-layered methodology for detecting phishing attacks (Figure 3). Just like gillnet comes in various mesh sizes, the mesh size of the first layer of phishGILLNET (phishGILLNET1) is larger than the second layer (phishGILLNET2) and the second layer (phishGILLNET2) is larger than the third layer (phishGILLNET3). Phishing attacks missed by phishGILLNET1 are caught by phishGILLNET2 and the ones missed by phishGILLNET2 are caught by phishGILLNET3. All three layers of phishGILLNET employ PLSA (see Section 4) to build a topic model that discovers phishing topics and non-phishing topics. phishGILLNET1 performs classification on unseen data using Fisher similarity function. phishGILNET2 builds a finer mesh utilizing PLSA topic features and AdaBoost (see Section 5). By employing PLSA, AdaBoost, and Co-Training (see Section 6), phishGILLNET3 further expands detection capability by building robust classifier from labeled and unlabeled data.

In order to build PLSA topic model, which all three layers of phishGILLNET employs, the methodology requires preparation of Term Document Frequency (TDF) matrix. Figure 4 shows the main components to build TDF, namely, Parser and TDF Matrix Builder. Both these components are described below:

Parser

Raw email data are typically present in Multipart Internet Mail Extension (MIME) format. phishGILLNET utilizes words and hyperlinks present in the body of the email to build PLSA model. Parser consists of the following:

MIME Parser

Parses email MIME message and extracts email headers and email body. Email body is further separated into HTML body part and text body part. For emails containing only text MIME part, the parser extracts text and hyperlinks. In a phishing email, these hyperlinks link to the phishing website.

HTML Parser

MIME message-containing HTML body part is included as multipart/html part in the email body part. When the MIME parser detects a HTML part, it invokes the HTML parser to separate out text, style-sheets, hyperlinks, and scripts. For the purpose of building PLSA model, both text and hyperlinks are considered.

Tokenizer

This tokenizes text present in email body and hyperlinks into separate words. Tokenizer utilizes white space (tabs, space, new lines) as token delimiters for the text. The hyperlinks are tokenized after replacing all non-alphanumeric characters with space.

TDF Matrix Builder

A term-document matrix describes the frequency of terms that occur in a collection of documents. The rows of the matrix correspond to document (d_i) in the collection and the columns correspond to terms (w_j) that present in those documents. For the text part, the terms w_j belong to one of the part-of-speech tags (adjectives, adverbs, nouns, and verbs). For the hyperlinks part, all terms are used to build TDF. The matrix entries $n(d_i, w_j)$ denote the number of times word w_j occurs in document

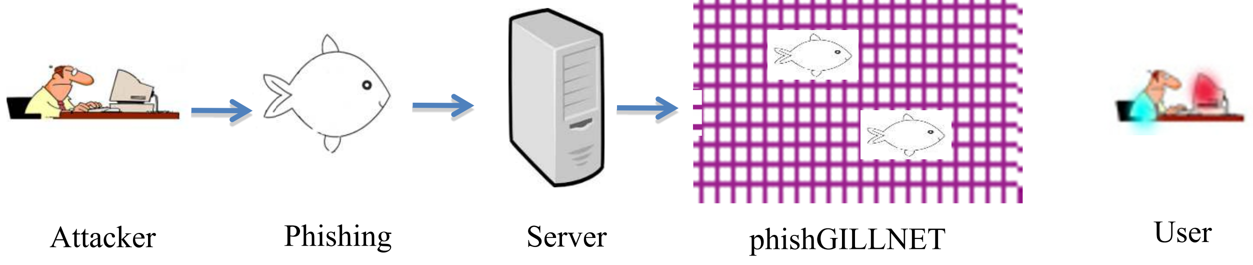

Figure 2 phishGILLNET.

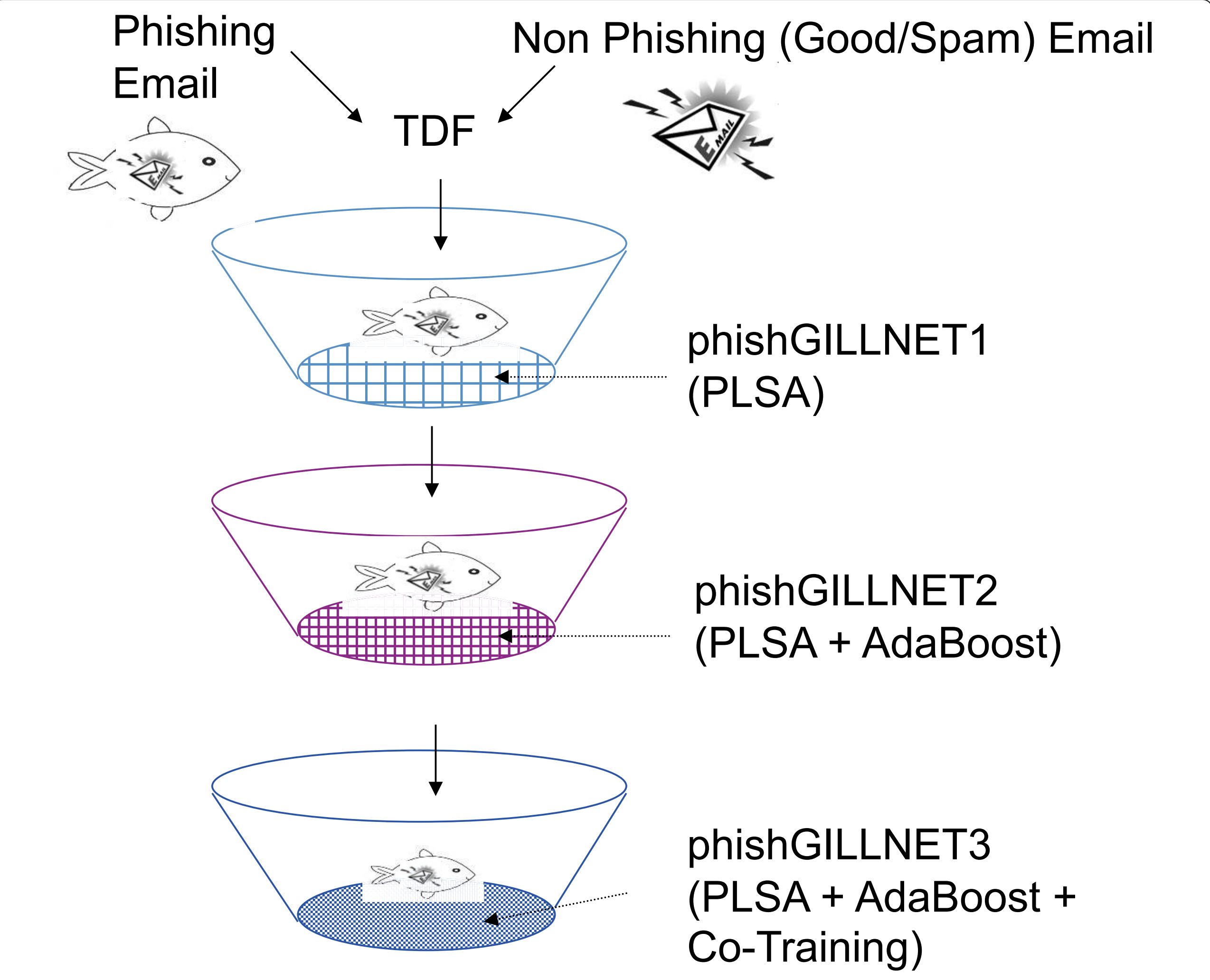
Phishing Email
Non Phishing (Good/Spam) Email
TDF
phishGILLNET1 (PLSA)
phishGILLNET2 (PLSA + AdaBoost)
phishGILLNET3 (PLSA + AdaBoost + Co-Training)

Figure 3 Multi-layered phishGILLNET.

d_i. Prior to building TDF Matrix, the following pre-processing steps must be accomplished.

Stop Words Removal

Stops words are words that do not contain important significance for building the model. Some example stop words include the, at, like, etc. We remove stop words from all the tokenized email text.

Stemming

Stemming is a method for removing inflexional endings from certain words. For example, word "consigned", after stemming becomes "consign". Porter's Stemming [50] algorithm is employed to stem words in email body.

Dictionary Lookup

WordNet [51] dictionary is employed to lookup words in dictionary. WorldNet database has Part-of-Speech (POS) extractor. It identifies verbs, nouns, adverbs, and adjectives. Words found in WorldNet database forms part of the input for building TDF matrix using text. For the hyperlinks TDF, WordNet lookup and spell checker is skipped.

Spell Checker

Attackers intentionally misspell words in a phishing email to avoid detection by standard spam filters. For words that are not found in WordNet database, Google's spell check API [52] is utilized to retrieve words that are similar to the misspelled word.

Levenshtein Distance

Levenshtein distance [53] is a metric for measuring the amount by which two words differ. The metric is also called edit distance. It is the minimum edit operations required to transform one word to another. The edit operations include insertion, deletion, and substitution of a new character. In a phish email, there are misspelled

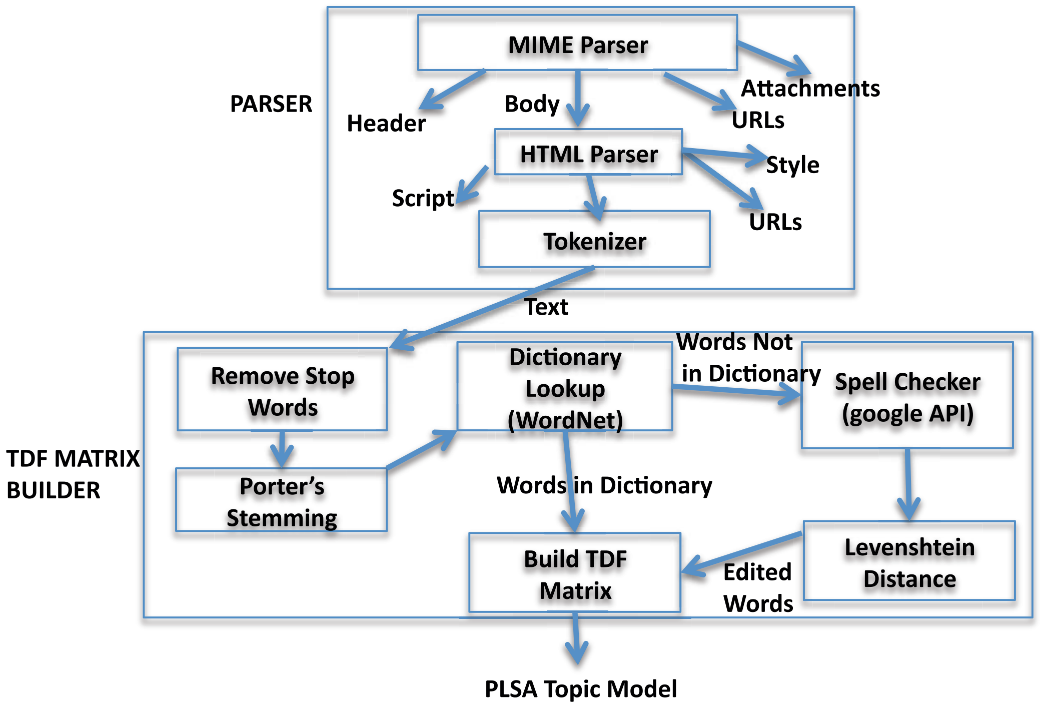

Figure 4 Parser and TDF matrix builder.

words, which after edit operation is found in dictionary. Examples include "vuln'a'rability", "youaccounts", etc. Also, there are terms made of garbage characters that are never found in dictionary. We consider only misspelled words that can be corrected after certain edit operation. After obtaining the suggested words using Google API, Levenshtein distance is computed. Only those words whose edit distance is less than some configured threshold (default value of 5) are further included for building TDF matrix.

Build TDF Matrix
For email body text, using words, (specifically adjectives, adverbs, nouns, and verbs), that found directly in dictionary and edited words using Levenshtein' edit operation, the term-document-frequency matrix is created. For email hyperlinks, all terms are used to build TDF matrix.

Thus, phishGILLNET accounts for misspelled words, conjoined words, and POS tags present in email body before building the TDF matrix. Once the TDF matrix is built using components described above, all three layers of phishGILLNET employs PLSA to build the topic model for phishing detection. The PLSA modeling technique is described in the following section.

4 PLSA
PLSA is a technique for topic discovery proposed by Hofmann [54,55]. The technique is closely related to Latent Semantic Analysis (LSA). While LSA is based on the foundations of linear algebra to perform a Singular Value Decomposition of co-occurrence tables, PLSA is a statistical method that defines a latent class model to perform probabilistic mixture decomposition. PLSA handles both synonyms, different words with similar meanings, and polysemy, words whose meaning changes according to context. PLSA has been applied in the field of information retrieval, natural language processing, machine learning, and image processing.

Model
The PLSA model maps the high-dimensional vector of words of a document to a lower dimensional vector of topics. The PLSA modeling is shown in Figure 5. Suppose we have a collection of documents, $d_i \in \{d_1, d_2,..., d_i\}$, and a set of words that occur in those documents $w_j \in \{w_1, w_2,...,w_J\}$. PLSA then associates a latent topic variable $z_k \in \{z_1, z_2,...,z_K\}$ with the occurrence of each word in a particular document. PLSA assumes conditional independence, such that words and documents are conditionally independent for a given topic. Thus, the PLSA model for the word-document co-occurrence can be expressed using the following join probability model:

$$P(d_i, w_j) = P(d_i) \sum_{k=1}^{k} P(w_j|z_k)P(z_k|d_i)$$

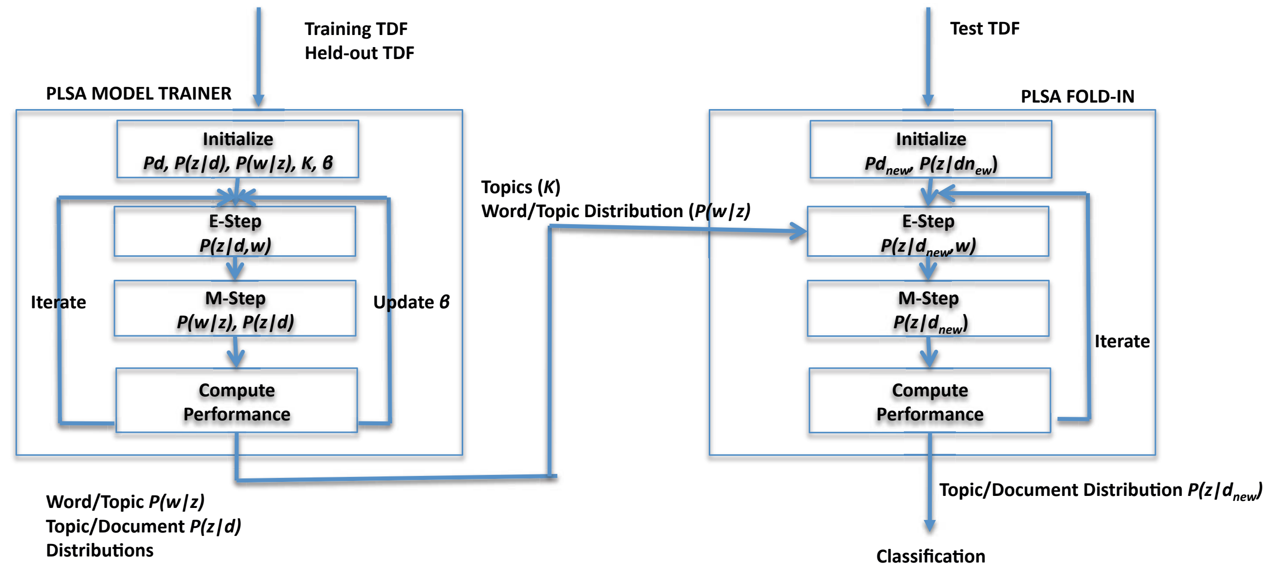

Figure 5 PLSA model.

where, $P(d_i)$ is the probability that a word will be observed in a given document d_i, $P(w_j|z_k)$ is the probability of a particular word conditioned on latent topic variable z_k, $P(z_k|d_i)$ is the probability distribution of specific document over the latent variable space, and K is the number of topics. The probability $P(w_j|z_k)$ corresponds to words that make up a given topic while the probability $P(z_k|d_i)$ corresponds to topics that a given document belong to. Unlike traditional cluster algorithm wherein a document may belong to just one cluster, PLSA gives the probabilities with which a given document may belong to one or more topics.

Expectation Maximization (EM) Algorithm

The model parameters $P(w_j|z_k)$ and $P(z_k|d_i)$ are estimated by maximizing the data log-likelihood using EM algorithm [20,21]. The maximum likelihood is

$$l = \sum_{d_i \in D} \sum_{w_j \in W} n(d_i, w_j) \log P(d_i, w_j)$$

By applying Bayes' rule, E-Step of the EM algorithm is given by

$$P(z_k|d_i, w_j) = \frac{P(z_k)P(w_j|z_k)P(d_i|z_k)}{\sum_{l \in K} P(z_l)P(w_j|z_l)P(d_i|z_l)}.$$

The M-Step obtained by maximizing the expected data log-likelihood is given by following expressions

$$P(w_j|z_k) = \frac{\sum_{d_i \in D} n(d_i, w_j)P(z_k|d_i, w_j)}{\sum_{w_m \in W} \sum_{d_i \in D} n(d_i, w_m)P(z_k|d_i, w_m)}$$

$$P(z_k|d_i) = \frac{\sum_{w_j \in W} n(d_i, w_j)P(z_k|d_i, w_j)}{n(d_i)}$$

The model parameters are estimated by iteratively alternating the E-Step and the M-Step until some desired termination criteria is satisfied. Stopping criteria may include no measurable difference in the log-likelihood between successive iterations or the maximum number of iterations.

TEM Algorithm

In order for the PLSA model to generalize well on "new" (unseen) documents, Hofmann [54,55] proposed a modified EM algorithm for PLSA called TEM algorithm. TEM is closely based on deterministic annealing. In TEM, a control parameter β is introduced in the E-step of the algorithm. The modified E-step is given as follows:

$$P(z_k|d_i, w_j) = \frac{P(z_k)[P(w_j|z_k)P(d_i|z_k)]^{\beta}}{\sum_{z_t \in Z} P(z_t)[P(w_j|z_t)P(d_i|z_t)]^{\beta}}.$$

In the above expression, substituting β with value of 1 yields the E-step of standard EM algorithm. The main advantage of the TEM algorithm over the standard EM algorithm is that TEM avoids model over-fitting. The optimal value of β is obtained by starting with a value of 1, evaluate the performance of EM on a held-out dataset, decrease the value of β, and check if the performance improves. The iterative procedure is stopped when there is no measurable increase in performance.

Folding-In

When a new (unseen) document is given, to compute the probability distribution of topic(s) that new documents belong to, a folding-in technique is employed. In PLSA, this is achieved by keeping the distribution of words that make up a topic ($P(w_j|z_k)$) fixed while the distribution of topics that new document belong to ($P(z_k|d_{\text{new}})$) is adapted at each M-step. The distribution $P(w_j|z_k)$ is obtained during the training phase of PLSA. The E-Step of the EM algorithm for folding-in is given by

$$P(z_k|d_{new}, w_j) = \frac{P(z_k)P(w_j|z_k)P(d_{new}|z_k)}{\sum_{z_t \in Z} P(z_t)P(w_j|z_t)P(d_{new}|z_t)}.$$

and for the M-Step, $P(w_j|z_k)$ is obtained from the training phase. The distribution of topics to new document is given by the following expression

$$P(z_k|d_{new}) = \frac{\sum_{w_i \in W} n(d_{new}, w_j)P(z_k|d_{new}, w_j)}{n(d_{new})}$$

Classification

The folding-in technique yields the probability distribution of new (unseen) documents belonging to one or more topics ($P(z_k|d_{\text{new}})$). Given a training set of labeled samples, belonging to one or more categories, the category of the new unlabeled document is obtained using a similarity function. After obtaining probability estimates using fold-in, to categorize new documents to a specific category, similarity scores between new documents and documents in the training set are computed. The category of the document in the training set that yields the highest similarity score is the category of the new document. A commonly used similarity function is the Euclidean distance function. However, the Euclidean distance is not a good metric for computing similarity between two probability distributions. Hofmann [54,55] derived the following Fisher-Kernel functions for the generative statistical PLSA model. The kernel consists of two components. The Kernel function due to the contribution of topic probabilities is given by

$$K1(d_i, d_n) = \sum_{z_k = Z} P(z_k|d_i)P(z_k|d_n)/P(z_k).$$

The above kernel function computes the overlap between topics and thus captures words with similar meanings and words that belong to the same topic. The contribution due to word to topic probability distribution is given by the following kernel function.

$$K2(d_i, d_n) = \sum_{w_j \in W} P(w_j|d_i)P(w_j|d_n) \sum_{z_k \in Z} \frac{P(z_k|d_i, w_j)P(z_k|d_n, w_j)}{P(w_j|z_k)}.$$

In the above kernel function $K2$, words with multiple meanings (polysemy) contribute to the similarity score.

The PLSA technique described above is applied to all the layers of phishGILLNET. The classification in phishGILLNET1 is achieved using the Fisher similarity function described here.

5 AdaBoost

The classification using similarity function is an efficient technique but not as robust as employing classification technique. To cope with data variability, one employs classifier ensemble. The idea behind classifier ensemble is to combine predictions of multiple classifiers and produce a single classifier. The prediction result from the combined classifier is generally better than that of individual classifiers. Results from an ensemble are less dependent on strangeness of employing a single training set and thus it reduces bias and variance. There are several ways of forming an ensemble or a collection. The two most popular ones are bagging and boosting. Both these methods rely on re-sampling of the data to obtain different training sets for each of the classifiers. Here, we employ the boosting technique, specifically, AdaBoost.

The idea behind AdaBoost, developed by Freund and Schapire [56], is to produce a series of classifiers. The training data used for each member of the series is chosen based on the performance of earlier classifiers in the series. Incorrectly predicted examples are selected more frequently than correctly predicted examples. Thus, boosting produces classifiers that are better in prediction that the current ensemble. Unlike bagging, AdaBoost considers performance of the earlier classifiers. The algorithm is detailed as follows:

Given input training data (x_1, y_1), (x_2, y_2),....,(x_m, y_m), where x_i belongs to feature space X and y_i belongs to label set Y = {-1, +1},

Step 0: Initialize weights for the first iteration, $D_1(i) = 1/m$

For iteration index t = 1,...,T, where T is the number of iterations,

Step 1: Train a weak learner using distribution D_t.

Step 2: Obtain weak hypothesis

$$h_t : X \to \{-1, +1\}$$

with error

$$\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i).$$

Step 3: Compute

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right).$$

Step 4: Updates weights for this step

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is the normalization factor.

The final strong classifier, which is a weighted majority of T weak hypothesis, is given as

$$H(x) = \text{sign}\left(\sum_{t\in T}\alpha_t h_t(x)\right).$$

phishGILLNET2 employs AdaBoost using several learning algorithms such as C4.5 decision trees [57], rule-based classifier RIPPER [58], random forest [59], support vector machines [60], and logistic regression [61] to build a robust classifier. The features to build the classifier are the topic distribution probabilities obtained from the PLSA model.

6 Co-Training

One of the major problems in building a classifier is the non-availability of labeled data. A classification task for phishing detection requires labeled phish examples and non-phish examples. While there are many data sources for obtaining general spam emails and good emails, there is very few labeled phishing email public corpus. As phishing emails and general spam emails share similar characteristics, human annotation result in incorrect labeling and hence the available corpus may not be perfect. Co-Training is an algorithm to solve this non-availability problem. The algorithm is proposed by Blum and Mitchell [62], for the problem of semi-supervised learning where there are both labeled and unlabeled examples. The goal of Co-Training is to enhance performance of learning algorithm when only a small set of labeled examples is available. The algorithm trains two classifiers separately on two sufficient and redundant views of the examples and lets the two classifiers label unlabeled examples for each other. The assumptions of the algorithm are that each view is conditionally independent given the class label and that each view is sufficient on its own for the purpose of classification. The algorithm works as follows: Given a set of labeled training instances (L) and a set of unlabeled instances (U), select u instances randomly from U to create a smaller pool U'. Iterate for k iterations the following steps:

- Split each instance x, and build two views x_1 and x_2.
- Use the training set L to build a classifier h_1 using x_1.
- Use the training set L to build a classifier h_2 using x_2.
- Label p positive and n negative instances from U' using the classifier h_1.
- Label p positive and n negative instances from U' using the classifier h_2.
- Add labeled instances to the training set L.
- Select 2 x (p + n) instances from unlabeled set U and to add it pool U'.

The idea behind the Co-Training algorithm is that the classifier h_1 adds examples to the labeled set which are in turn used by the classifier h_2 in the next iteration and vice versa. This process should make classifiers h_1 and h_2 to agree with each other after several iterations. Blum and Mitchell [62] validated the Co-Training algorithm using 1,051 web page data where x_1 consisted of words that appeared on the web page and x_2 consisted of words in all the hyper links that pointed to the web page. Nigam and Ghani [63] proposed a variant to the Co-Training called Co-EM algorithm. The Co-EM algorithm is not incremental in nature and it labels all unlabeled data at each iteration. Furthermore, only the data labeled by one classifier are used by the other classifier and vice versa. Co-Training was applied to email domain by Kiritchenko and Matwin [64]. Authors used Co-Training to classify interesting versus uninteresting email. Chan et al. [65] demonstrated Co-Training for spam classification on a corpus of 2,883 emails. Wan [66] applied Co-Training to cross-lingual sentiment classification. Kumar and Daumé [67] extended Co-Training to unsupervised spectral clustering algorithm where in clusters identified in one view is used to label data in other view so as to modify the graph structure.

Here, we propose to utilize Co-Training to evaluate the effectiveness of topic model to classify phishing on a large corpus of labeled and unlabeled data. phishGILLNET3 employs Co-Training to build classifiers on two views of the data, namely text view and hyperlink view, starting with small labeled sample and pool of unlabeled samples, and iteratively build a robust classifier for phishing detection. The experimental design including the datasets employed, data preparation, training and test strategies, and performance measures, is described in the following section.

7 Experimental design

In this section, we present the details on experiments designed to build and evaluate phishGILLNET. This includes datasets employed, data preparation, training and test strategies, and measures to evaluate performance.

7.1 Datasets

Four publicly available email datasets and one publicly available phish URL dataset were used to evaluate phishGILLNET. Email datasets include (i) ham (good) emails from SpamAssassin corpus [68], (ii) phishing emails from the PhishingCorpus [69], (iii) good emails from Enron Email Dataset [70], and (iv) spam emails from

SPAM Archive [71]. Phish URL dataset includes (v) PhishTank [72].

(i) SpamAssassin [68]

SpamAssassin corpus contains a total of 6,047 messages, of which, 4,150 messages are good and the remaining are spam. These messages were collected by the SpamAssassin project for the years 2002-2003 and made available to the research community. For evaluation in this study, spam messages are not used (only 4,150 good messages are used instead).

(ii) PhishingCorpus [69]

PhishingCorpus contains 4,550 phishing emails. These emails were collected by an individual for the period 2004-2007 and donated to the research community. For evaluation, all the phishing emails from this corpus were used.

(iii) Enron Email Dataset [70]

This dataset contains data from about 150 senior management people of Enron that was made public by the Federal Energy Regulatory Commission during its investigation. This dataset contains approximately 500,000 emails. Out of the Enron emails, we employed 136,226 emails from the inbox and sent folder of the mailbox, thus ensuring only good emails from this corpus.

(iv) SPAM Archive [71]

SPAM archive contains spam emails collected by Bruce Guenter [71] using various bait accounts since 1998. We used all spam emails of January 2011 through November 2011. This accounted for 336,070 emails thus size of the total corpus was 470,000 (approximately). SPAM archive does not distinguish between “spam” and “phishing” emails. Thus, it is an ideal dataset to evaluate the architecture using Co-Training, which is a semi-supervised algorithm that employs labeled and unlabeled data.

(v) PhishTank [72]

PhishTank URLs are manually verified by human experts that it is a confirmed phish attack. We collected 48,000 phish URLs from phishtank.com for the year 2011.

7.2 Data preparation

Two sets of public dataset combinations were used to build and evaluate the PLSA model. The first set of experiments (combination1) employed datasets (i) & (ii) while the second set of experiments (combination2) employed (iii) & (iv). The first set is a much smaller public corpus than the second set. In combination1, there are a total of 8,700 messages, 4,550 phishing, and 4,150 good emails. While in combination1 all emails are labeled, the combination2, specifically (iv), does not distinguish between phishing and spam emails. In order to compute misclassification errors, phishing emails in SPAM archive were segregated using the following semi automated approach. Hyperlinks in emails were extracted using a HTML parser. SURBL [73] provides a reputation lookup service for domains that are confirmed phish hosting domains. By using a combination of phishtank.com URLs and domain reputation data from SURBL, if a match is found for the SURBL domains or phishtank URL in the hyperlinks present in an email, then that email is labeled as a “phish” email. This resulted in phish emails of 47,783 out of 336,070 spam emails. Thus, the distribution of emails in combination2 is 10% phish, 61% spam, and 29% good. According to the Internet Security Threat Report 2010 from Symantec [74], that collected and analyzed billions of emails from 2009, in a realistic mail system, 85-90% of all emails are spam and 5-10% of all spam emails are phish. Thus, to have realistic distribution of data in combination2, our experiments were conducted with 10% phish, 80% spam, and 10% good emails. Thus, the size of the corpus used for combination2 is 400,000 emails, which is 10 times the size of corpus used by Bergholz et al. [17] and one of the largest email corpus used for phishing detection. Also, we used public corpus and hence our results can be reproduced.

All the messages were parsed using a MIME parser to separate email headers from email body. Multipart messages containing HTML parts were further parsed using a HTML parser to extract the body text and hyperlinks. Both MIME and HTML parsers were written in this study using Java programming language. For evaluation, only messages that contain body text and hyperlinks were considered. Thus, messages that failed parser and attachments were not included for building models.

7.3 Training and testing

Experiments were conducted using *k*-fold cross-validation strategy with a *k* value of 10. Thus, 90% of the dataset was used during training and 10% of the dataset was used for testing. In order to build the PLSA model, the training data are further split into 90% for building the topic model and 10% for computing perplexity, thus, independent datasets for training, computing perplexity, and testing. The TDF matrix builder (see Section 3) is used to build the term-document matrix for each set. The topic distribution probabilities on the test set is derived using PLSA fold-in (see Section 4). Classification in phishGILLNET1 (see Section 8) is achieved using Fisher similarity function while phishGILLNET2 (see Section 9) employs AdaBoost and phishGILLNET3 employs AdaBoost and Co-Training (see Section 10).

7.4 Performance evaluation metrics

The quality of the PLSA model is evaluated using two measures of performance, namely, log-likelihood and perplexity. The training dataset is split into a set for building the model (training data) and a set (held out) for validating the model using these performance measures.

Log Likelihood

The log likelihood on the training dataset can be computed using the following expression.

$$l(Training\,Data) = \sum_{d_i \in D} \sum_{w_j \in W} n(d_i, w_j) \log P(d_i, w_j)$$

Perplexity

Perplexity, a measure of uncertainty in natural language models, gives a better assessment of how well the model generalizes on unseen (new) data. The lower the perplexity, the better the generalization and hence the classification. Perplexity for a PLSA model is defined by Hofmann [54,55] as follows

$$\text{Perplexity} = P(\text{HeldOutData}) = \exp\left[-\frac{\sum_{h,j} n(d_h, w_j) \log P(w_j|d_h)}{\sum_{h,j} n(d_h, w_j)}\right]$$

where $n(d_h, w_j)$ is the number of times the word w_j occurs in held out document d_h and $P(w_j|d_h)$ is the probability that word w_j occurs in document d_h. One can see that classification is proportional to the number of topics.

The classification performance is measured using the following standard measures of performance, namely, Precision, Recall, F-measure, and Area under the ROC Curve (AUC). They are defined as follows

$$\text{Perplexity} = P(\text{HeldOutData}) = \exp\left[-\frac{\sum_{h,j} n(d_h, w_j) \log P(w_j|d_h)}{\sum_{h,j} n(d_h, w_j)}\right]$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{F}}$$

$$\text{Recal} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F = \frac{2 * (\text{precision} \times \text{Recall})}{(\text{precision} + \text{Recall})}$$

where TP is number of true positive, FP is number of false positive, and FN is number of false negative. ROC curve is a plot of true positive rate versus false positive rate. The two-dimensional depiction of classifier performance in a ROC curve is reduced to single scalar value representing expected performance by computing the AUC. The AUC of a classifier is equal to the probability that a classifier will rank a randomly chosen positive example higher than the randomly chosen negative example.

Experiments conducted using publicly available datasets and performance of each layer of phishGILLNET are reported in the following sections.

8 phishGILLNET1

phishGILLNET1 is the top layer of the multi-layered phishing detection methodology. It employs PLSA topic modeling technique to discover phishing and non-phishing topics and Fisher similarity function for classification. The architecture of phishGILLNET1 and experimental results are reported in this section.

8.1 Architecture

The architecture of phishGILLNET1 is shown in Figure 6. The architecture has four main components: parser, TDF matrix builder, PLSA model trainer, PLSA fold-in and Classifier. The architecture employs the parser to parse data and TDF matrix builder to build the TDF matrix (described in Section 3). It employs the PLSA modeling technique (described in Section 4) to build the topic model.

PLSA Model Trainer

The input to the model is the TDF matrix of the training dataset. In this study, TEM algorithm described earlier in Section 4 was employed to build the topic model. PLSA algorithm is implemented using Java programming language.

Initialization

PLSA requires number of topics, K, to be specified at initialization similar to cluster analysis. The probability distributions are initialized using random numbers.

E-Step

The joint probability distribution values are computed using initialized probability distribution.

M-Step

In the M-step, word-topic and topic-document probabilities are computed using expressions given in PLSA model section.

Compute Performance Metric

Performance measure, log likelihood, and perplexity are computed according to the equations given in the experimental design section (see Section 7).

PLSA Fold-In

In fold-in, test data probability distributions are computed using the $P(w|z)$ value from the training phase as input. The TEM algorithm is employed to compute distributions on the test data set, while $P(w|z)$ is kept fixed.

Classifier

phishGILLNET1 categorizes email as phishing versus non-phishing using a similarity function. Using labeled emails as input dataset, containing phishing emails and non-phishing emails, topic distribution probabilities and word distribution probabilities are obtained by building a PLSA model. The similarity score is computed using Fisher Kernel similarity function between test emails and emails in the training set. The label of the training email that yields the highest similarity score is considered the label of the test email.

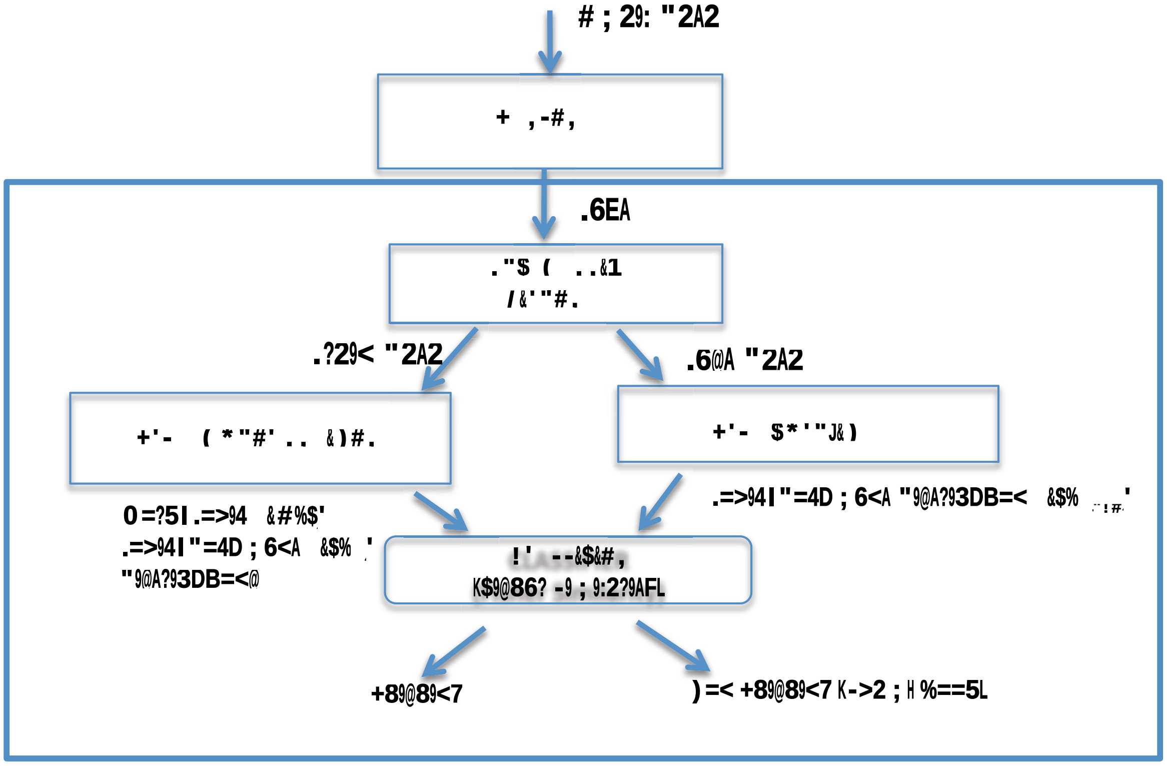

Figure 6 phishGILLNET1 architecture.

8.2 Results

Experiments were conducted using two combinations of datasets, combination1 containing 8 K emails and combination2 containing 400 K emails (see Section 7). Experiments were repeated on two machines (i) Mac OS X (10.6.5), 2.66 GHz Intel Core i7, 4 GB RAM and (ii) Cent OS (linux 2.6.18), 1.99 GHz Intel Core 2 Duo, 4 GB RAM. The average computation time is measured and reported here. For phishing detection, PLSA model consisting of phishing and non-phishing topics is first developed. Parsed email data are used to build the TDF matrix. After various pre-processing steps, that includes tokenization, stop words removal, and porter's stemming, the POS tags are extracted using WordNet [51]. We further observed phishing emails contained intentionally misspelled words, such as, "verificacion", "verifcation", and conjoined words such as "yourchasebank", "yourpaypal". Words that were not found in WordNet direct lookup were further processed using Google's suggestion API [52] and Levenshtein [53] editing function. If the edit distance is within the threshold value of 5 and if the second lookup in WordNet succeeded, those words were added to build the TDF matrix.

The TEM algorithm, detailed in Section 4, is employed to build the PLSA model. The number of topics, K, chosen for evaluation includes values ranging from 2 to 200. The maximum number of TEM iterations for convergence was set to 500. The annealing parameter β was initialized to value of 1.0 and decremented in increments of 0.25 to see if performance improves on the held out dataset.

Results from the PLSA model training and model evaluation are presented in Tables 2, 3, and 4 and Figure 7. In Table 2 the word-to-topic distribution probabilities of top 12 words for two topics (a phishing topic and a non-phishing topic) are shown. From this table, it is evident

Table 2 phishGILLNET1–PLSA word/topic probability distribution

Topic (*z*) (phishing)		Topic (*z*) (non-phishing)	
Word (*w*)	**Probability *P*(*w*|*z*)**	**Word (*w*)**	**Probability *P*(*w*|*z*)**
Bank	0.058	Ocean	0.024
Online	0.046	Honolulu	0.014
Banking	0.033	Imminent	0.013
America	0.032	Assuring	0.010
Account	0.021	Handsome	0.009
Update	0.019	Builder	0.007
Security	0.017	Lush	0.005
Customer	0.014	Lousy	0.005
Below	0.013	Roads	0.005
Link	0.013	Vantage	0.005
Click	0.011	Sweetness	0.005
Please	0.011	Wine	0.004

Table 3 phishGILLNET1–PLSA model performance

Number of topics	Dataset combination1 (8 K public corpus)		Dataset combination2 (400 K public corpus)	
	Perplexity	Computation time (min)	Perplexity	Computation time (min)
2	523.56	1.65	6742.42	32.31
10	278.81	2.52	4441.96	45.20
25	277.62	3.12	1748.55	52.00
50	274.31	3.63	1593.10	65.00
100	273.33	7.42	1461.78	112.50
200	271.27	15.38	1425.36	185.20

which words make up a phishing topic and which words make up a good topic. Two methods of evaluating the performance of a PLSA model, log likelihood on the training data and perplexity on the held out, were used. A model that yields lowest log-likelihood on the training data is considered the best model. The number of EM steps was varied from 1 to 350. The plot obtained for the (number of topics) K value of 10 topics on combination1 datasets is shown in Figure 7. As it can be seen from the figure, the negative log-likelihood (see Section 7) drops steeply until EM iterations of 15 and drops gradually after that indicating (almost) convergence and triggering the stopping criteria. The log-likelihood is not a good measure for model generalization. The model with the lowest perplexity is the one that generalizes well for classifying new/unseen data. In order to evaluate performance on held out data, the number of topics was varied from $K = 2$ to $K = 200$. As it can be seen from Table 3 on the dataset combination1, the perplexity for a K value of 10 yielded 278 and did not change significantly for higher values of K. On the dataset combination2, a K value of 200 yielded 1,475 and did not change significantly for values larger than 200. The PLSA models were then evaluated for classification performance on test data. This requires computation of topic/document probability distributions on test data. This is achieved using the PLSA's folding-in technique where the TEM algorithm is employed by keeping the word/topic probability distributions fixed. The Fisher similarity score was then computed between each test data and training data. The label of the training data that yields the highest similarity score is the label of the test data. It can be seen from Table 4 that the PLSA model yielded *F*-measure of 98.3% on dataset combination1 and 98.1% on dataset combination2. Results on the large public corpus of 400 K emails show the robustness of phishGILLNET1 for phishing detection. A K value of 200 yielded the best *F*-measure and lowest false positive on dataset combination2. One can see (Table 4) that performance is almost perfect for K value of 200 and both precision and *F*-measure are very close to 1. The corresponding computation time (average on two machines) on 200 topic model on dataset combination2 is approximately 3 h.

In order to compare the performance of phishGILLNET1 with that of support vector machines, the TDF matrix of dataset combination2 was utilized. To build the SVM classifier, first the dimensionality reduction technique, Principal Component Analysis, was applied to TDF for computation reasons. In addition, features were selected by applying the information gain criteria. WEKA software using the libSVM library was used to build the SVM classifier. Results from SVM with feature selection are reported in Table 5. It can be seen that SVM results (*F*-measure of 95.9%) are worse than phishGILLNET1 (*F*-measure of 98.1%). In addition, SVM took close to 9 h to train, whereas phishGILLNET1 using 200 topics took approximately 3 h.

9 phishGILLNET2

phishGILLNET2 is a finer layer than phishGILLNET1. Instead of using Fisher similarity function for categorization using topic distribution probabilities, AdaBoost is employed to build a robust classifier using PLSA topic distribution probabilities as feature. Furthermore,

Table 4 phishGILLNET1–classification performance

Number of topics	Dataset combination1 (8 K public corpus)			Dataset combination2 (400 K public corpus)		
	FPR	Precision	*F*-measure	FPR	Precision	*F*-measure
2	0.02	0.977	0.975	0.025	0.971	0.970
10	0.00	0.983	0.983	0.014	0.976	0.976
25	0.00	0.983	0.983	0.010	0.977	0.977
50	0.00	0.983	0.983	0.009	0.981	0.980
100	0.00	0.983	0.983	0.004	0.981	0.981
200	0.00	0.983	0.983	0.001	0.981	0.981

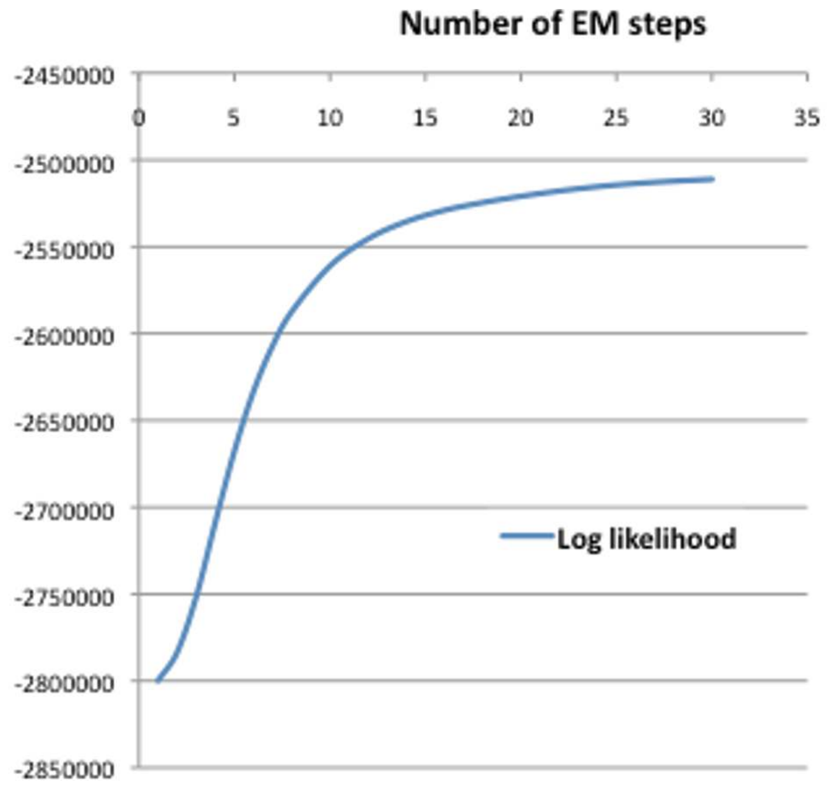

Figure 7 phishGILLNET1 performance–log likelihood versus number of EM steps.

phishGILLNET2 performs 3-class classification (phish, spam, good) as well as binary classification (phish, not phish). The architecture of phishGILLNET2 and experimental results are reported next.

9.1 Architecture

The architecture of phishGILLNET2 is shown in Figure 8. phishGILLNET2 employs AdaBoost as the classifier ensemble. The PLSA topics are discovered as before in phishGILLNET1 and topic distribution probabilities on training data are estimated. AdaBoost classifier ensemble is built using these probabilities as features and several weak learners. Existing classification techniques such as C4.5 decision tree, rule based-classifier (RIPPER), random forest, support vector machines, and logistic regression are used as the weak learners in phishGILLNET2. The performance of the classifier is compared using the metrics reported in Section 7. The open source software WEKA was used for the implementation of phishGILLNET2.

Table 5 Classification performance of SVM on dataset combination2

Method	FPR	Precision	*F*-measure	Computation time (min)
SVM with feature selection	0.14	0.963	0.959	530

9.2 Results

Experiments were conducted on the public dataset combination2. The total email corpus of 400,000 emails was used for validating this architecture (40,000 phish, 40,000 good, and 320,000 spam). Experiments were conducted using *k*-fold cross validation, with a *k* value of 10. Thus, for each trial, 90% of the emails were used for training and 10% were used for testing. PLSA topic models were built for number of topics (*K*) 50, 100, and 200. Each model thus results in corresponding number of topic distribution probabilities (features) 50, 100, and 200, respectively. Classifiers were then built using these features and AdaBoost algorithm. Experiments were conducted on two machines (i) Mac OS X (10.6.5), 2.66 GHz Intel Core i7, 4-GB RAM and (ii) Cent OS (linux 2.6.18), 1.99 GHz Intel Core 2 Duo, 4-GB RAM. The average computation times are measured and reported here. The computation times reported here are the times to perform the cross-validation after extraction of topic features. Time to build the PLSA models is reported in Section 8.

Results from the experiments are presented in Tables 6 and 7. Only top five performing classifier results are presented here. The classification performance is reported in Table 6 for 3-class classification and Table 7 for binary classification. For the 3-class problem, boosting with the random forest technique as the base learner yielded the

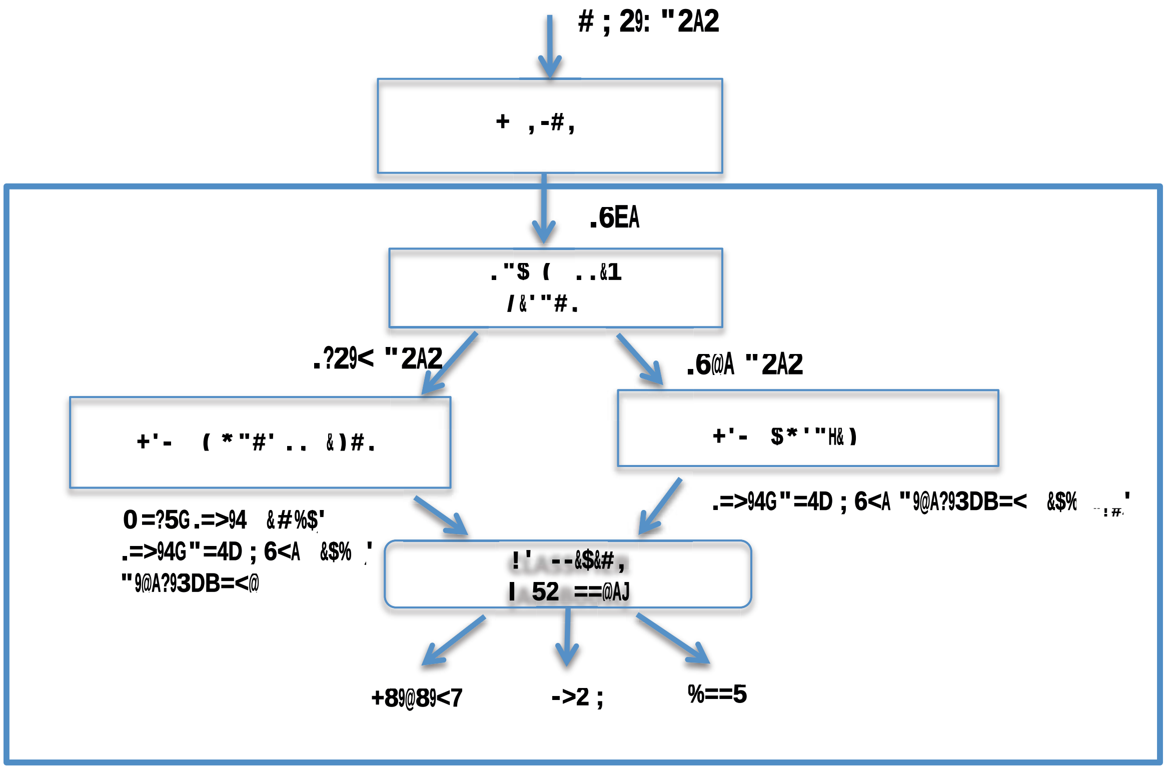

Figure 8 phishGILLNET2 architecture.

best precision and best *F*-measure of 97.7% for a *K* value of 200. For the 2-class problem, boosting using the logistic regression base learner yielded the best precision and *F*-measure of 99.7% for *k* value of 200. Thus, for the binary classification phishGILLNET2 resulted in better *F*-measure (99.7%) compared to phishGILLNET1 (*F*-measure 98.1%). Boosting using random forest technique yielded 99.5% for the same number of topics in phishGILLNET2. Random forest is computationally faster than most of the other techniques that were evaluated. Results from phishGILLNET2 shows boosting significantly improves classification performance.

Table 6 phishGILLNET2–3-Class (phish versus spam versus good) classification performance

Topics	Weak learner for boosting	TPR	FPR	Precision	Recall	*F*-measure	ROC Area	Time (s)
50	C4.5	0.954	0.088	0.954	0.954	0.954	0.944	1.84
50	RIPPER	0.964	0.069	0.964	0.964	0.964	0.955	12.07
50	Random forest	0.974	0.079	0.973	0.974	0.973	0.996	3.09
50	SVM	0.91	0.199	0.907	0.91	0.908	0.867	12.41
50	Logistic	0.909	0.238	0.905	0.909	0.905	0.957	2.42
100	C4.5	0.967	0.068	0.967	0.967	0.967	0.961	5.06
100	RIPPER	0.974	0.043	0.975	0.974	0.975	0.971	16.6
100	Random forest	0.976	0.075	0.975	0.976	0.975	0.997	3.31
100	SVM	0.964	0.095	0.964	0.964	0.963	0.94	11.32
100	Logistic	0.971	0.065	0.97	0.971	0.97	0.989	5.05
200	C4.5	0.969	0.061	0.969	0.969	0.969	0.961	8.93
200	RIPPER	0.972	0.048	0.973	0.972	0.972	0.968	24.77
200	**Random forest**	**0.977**	**0.06**	**0.977**	**0.977**	**0.977**	**0.996**	**3.7**
200	SVM	0.97	0.071	0.971	0.97	0.97	0.953	18.62
200	Logistic	0.971	0.065	0.97	0.97	0.97	0.989	6.15

Table 7 phishGILLNET2–binary (phish versus not phish) classification performance

Topics	Weak learner for boosting	TPR	FPR	Precision	Recall	*F*-measure	ROC Area	Time (s)
50	C4.5	0.985	0.055	0.985	0.985	0.985	0.966	0.79
50	RIPPER	0.989	0.051	0.989	0.989	0.989	0.968	4.17
50	Random forest	0.993	0.053	0.993	0.993	0.993	0.999	1.31
50	SVM	0.939	0.355	0.935	0.939	0.937	0.792	12.67
50	Logistic	0.938	0.421	0.932	0.938	0.933	0.957	1.0
100	C4.5	0.995	0.02	0.995	0.995	0.995	0.987	1.58
100	RIPPER	0.997	0.012	0.997	0.997	0.997	0.993	6.82
100	Random forest	0.994	0.052	0.994	0.994	0.994	0.999	2.32
100	SVM	0.992	0.069	0.992	0.992	0.992	0.961	10.55
100	Logistic	0.995	0.023	0.995	0.995	0.995	0.994	2.17
200	C4.5	0.996	0.019	0.996	0.996	0.996	0.991	2.51
200	RIPPER	0.994	0.024	0.994	0.994	0.994	0.987	7.85
200	Random forest	0.995	0.037	0.995	0.995	0.995	0.999	2.87
200	SVM	0.988	0.098	0.988	0.988	0.988	0.945	10.78
200	**Logistic**	**0.997**	**0.018**	**0.997**	**0.997**	**0.997**	**0.997**	**4.11**

10 phishGILLNET3

phishGILLNET3 is the third layer of the multi-layered phishGILLNET. This layer employs AdaBoost and Co-Training algorithm to build a robust classifier using large corpus of unlabeled data. Labeling data to build classifiers require significant time and human labor. phishGILLNET3 eliminates the need for fully labeled corpus. The architecture and experimental results are reported next.

10.1 Architecture

The architecture of phishGILLNET3 is shown in Figure 9. The motivation for this implementation is to evaluate the robustness of topic model, specifically PLSA, on a large corpus of unlabeled data. This architecture implements the Co-Training algorithm and applies to the email domain. The algorithm starts with small corpus of labeled emails (phishing and non-phishing). Using parser components (see Section 3), email data are parsed into text present in the body of the email and hyper links. The text and hyper links form two views for applying the Co-Training algorithm. One of the assumptions behind the Co-Training algorithm is that the two views should not be perfectly co-related. In a phishing email, the text in body of the email will contain enticing content asking the user to click the hyperlink and the hyperlink and accompanying web content will contain the impersonating entity. There may be some correlation between the two views (body text and

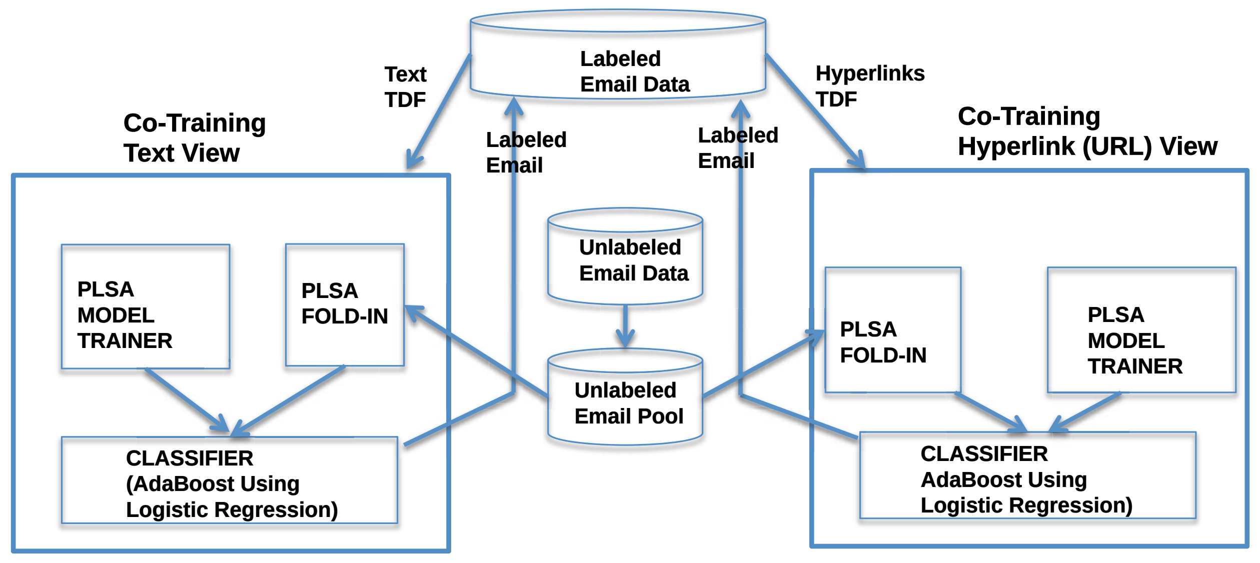

Figure 9 phishGILLNET3 architecture.

hyperlinks) but not perfect correlation. In addition, review of literature (see Section 2) shows that classifiers built just using hyperlinks and just using body text yields good classification performance. Hence, we apply co-training to the email body text and hyperlink views. For the body text, all words in the email are used to build the PLSA model for text view. For the hyperlinks, terms are extracted by replacing all non-alphanumeric characters as token separator in the hyperlinks. These terms are used to build the PLSA model for the hyperlink view.

For both views, once the PLSA model is built, the topic distribution probabilities are extracted as features. These features are used to build the classifier. The text classifier and the hyperlink classifier classify unlabeled email data and most confidently predicted email data are added to labeled corpus for the next iteration of co-training. The process repeats until there is no more unlabeled data to label.

10.2 Results

Experiments were conducted using the dataset combination2. Experiments were repeated on two machines to measure the computation time. Of the total corpus of 400 K, 10% of the data (40 K) was used as labeled data (*L*) for the first iteration of the Co-Training algorithm. The parameters that yielded the best performance in phishGILLNET2 are employed to build phishGILLNET3. This implies that number of topics *K* is 200 and the weak learner for the AdaBoost is logistic regression (see Table 7). The parameters of the Co-Training algorithm are *p* (phish) = 200 and *n* (not phish) = 1800. This unbalanced dataset is realistic with proportion of phishing emails in large-scale mail systems. The size of the unlabeled pool *U'* is 5000. After each iterations of Co-Training, the text view classifier labels 2,000 emails and the hyperlink view labels 2,000 emails resulting in 4,000 additional labeled data for the next iteration of Co-Training. The pool *U'* is replenished by selecting 4,000 additional emails randomly from the unlabeled set *U*.

Results from the Co-Training algorithm of the combined hyper link and text classifiers are tabulated in Table 8. After ten iterations of Co-Training, it is evident that phishGILLNET3 results in better performance than phishGILLNET2 (99.8% as compared 99.7%). More iterations of the algorithm resulted in an *F*-measure 100%. Results show the robustness of PLSA, AdaBoost, and Co-Training algorithm to detect phishing. Moreover, phishGILLNET3 achieves superior performance using 10% of the labeled data thus saving time, effort, and errors associated with human annotation.

Table 8 phishGILLNET3–binary (phish versus not phish) classification performance

Iteration number	TPR	FPR	Precision	Recall	*F*-measure	ROC area
5	0.997	0.014	0.997	0.997	0.997	0.987
10	0.998	0.015	0.998	0.998	0.998	0.99
15	0.999	0.014	0.999	0.999	0.999	0.989
20	1.0	0.012	1.0	1.0	1.0	0.991
25	1.0	0.012	1.0	1.0	1.0	0.991
30	1.0	0.013	1.0	1.0	1.0	0.991
35	1.0	0.015	1.0	1.0	1.0	0.991
40	1.0	0.009	1.0	1.0	1.0	0.993
45	**1.0**	**0.0009**	**1.0**	**1.0**	**1.0**	**0.999**

(PLSA 200 topics + AdaBoost with logistic regression weak learner + Co-Training)

11 Performance comparison

The performance of phishing detection architecture, phishGILLNET, is compared with state-of-the-art research that attempted to solve phishing detection. Performance of each layer of phishGILLNET was compared with ten different published researches ranging from year 2007 to 2011. Comparison was also performed using support vector machines using words (instead of topic probabilities) as features. In Table 9 we show characteristics of our work and the state-of-the-art research. The corpus used by phishGILLNET is exclusively public where as in the state-of-the-art six of them have used public, two private, and the other two mix of private and public. phishGILLNET has used the largest public corpus of size 400 K emails. Thus, results from phishGILLNET are repeatable. The corpus used by phishGILLNET is ten times more than the next [17] in terms of size. Thus, phishGILLNET demonstrates the scalability aspect. The most recent public corpus (year 2011) is used by phishGILLNET for evaluation. phishGILLNET2 supports both 3-class (phish, spam, good) and binary (phish, not-phish) classification. The only method that performs 3-class classification is that of Gansterer and Pölz [15]. All the others perform binary classification. phishGILLNET3 is the only method that handles unlabeled data. This is the most powerful feature and important contribution of phishGILLNET. To the best of authors' knowledge, there is no other study that applied Co-Training for phishing detection and certainly not at this scale. The closest research study is by Chan et al. [65] who applied Co-Training for spam classification on a small dataset of 2,883 emails.

Results of phishGILLNET comparing the state-of-the-art research are tabulated in Tables 10 and 11. The performance metric that is compared is the *F*-measure (for binary classification) and accuracy (for 3-class classification). On the 3-class classification (see Table 10), the comparison of phishGILLNET2 with the study of Gansterer and Pölz [15] on the accuracy metric shows that phishGILLNET2 resulted in a better performance (97.7%) compared with the best result obtained by Gansterer and Pölz [15].

Table 9 phishGILLNET and competing methods characteristics

Method	Year of publication	Corpus (public/ private/mix	Max data	Year when data source used generated	3-class classification	Can handle unlabelled?
Chan et al. [65]	2004	Public	2.8 K	NA	No	Yes
PILFER [10]	2007	Public	7.8 K	2002-2006	No	No
Abu-Nimeh et al. [11]	2007	Private	2.8 K	2005-2006	No	No
Bergholz et al. [16]	2008	Public	8 K	2004-2007	No	No
Abu-Nimeh et al. [12]	2009	Mix	6.5 K	2006-2007	No	No
Gansterer and Pölz [15]	2009	Mix	15 K	2007	Yes	No
Toolan and Carthy [19]	2010	Public	8.3 K	2004-2007	No	No
Bergholz et al. [17]	2010	Private	40 K	2007	No	No
Khonji et al. [20]	2011	Public	8.2 K	2003-2007	No	No
Al-Momani et al. [21]	2011	Public	8.7 K	2003-2007	No	No
phishGILLNET1	-	Public	8.7 K	2003-2007	No	No
phishGILLNET2	-	Public	400 K	2011 (phish-40 K, spam 320 K) 2001 (good-40 K)	Yes	No
phiahGILLNET3	-	Public	400 K	2011 (phish-40 K, spam-320 K) 2001 (good-40 K)	No	Yes

Thus, topic features using AdaBoost are robust for 3-class classification.

For the binary classification, of the ten state-of-the-art researches, only seven of them reported *F*-measure results. It is evident from the results in Table 11 that phishGILLNET3 resulted in an *F*-measure of 100%. phishGILLNET3 is the top ranked method followed by Bergholz et al. [17], which reported an *F*-measure of 99.89%. Thus, it is evident that the PLSA, AdaBoost, and Co-Training algorithm employed by phishGILLNET3 significantly boosts performance. Moreover, phishGILLNET3 has the additional advantage of not requiring 100% labeled samples thus saving significant manual work. Not relying heavily on manual annotation also has the advantage of the method being less prone to human error and disagreement, as one may consider a spam email as good email and vice versa. Thus, phishGILLNET3 is not only superior on *F*-measure, but also has these additional advantages. phishGILLNET2, that employed AdaBoost classifier and did not employ Co-Training, came close third with an *F*-measure of 99.70. However, phishGILLNET2 required fully labeled samples unlike phishGILLNET3. Another interesting observation is the top four of the top ten methods employed topic features for building classifiers. While the second and fourth ranked methods utilized features in addition to topic features, phishGILLNET utilized exclusively topic features. Thus, results from the top four methods prove the robustness of using topic features for phishing classification.

Table 10 Performance comparison–3-class classification

Method	Accuracy (%)	Rank
phishGILLNET2	97.70	1
Gansterer and Pölz [15]	97.00	2

12 Conclusions

A multi-layered methodology, called phishGILLNET, is proposed and evaluated for phishing detection. All three layers of phishGILLNET employ PLSA to discover phishing and non-phishing topics. phishGILLNET1 categorizes unseen data using Fisher similarity. phishGILLNET2 employs AdaBoost using PLSA topic features and builds a better classifier than phishGILLNET1. phishGILLNET3 builds a robust classifier using only a fraction of labeled samples and applying Co-Training to label additional samples. The novelty of this architecture comes from employing semantic features to build the detection model. Intentional misspelled words found in phishing are handled using Levenshtein editing and Google APIs for correction before building the TDF matrix. One of the important contributions of this article is the use of Co-Training on a large corpus of unlabeled data to detect phishing attacks.

The architecture developed is compared with ten state-of-the-art methods. The performance of phishGILLNET3 is better than all the other competing methods and achieves an *F*-measure of 100%. Evaluation of phishGILLNET3 is done on a very large dataset (400 K emails)

Table 11 Performance comparison–binary classification

Method	*F*-measure (%)	Rank
phishGILLNET3	100.00	1
Bergholz et al. [17]	99.89	2
phishGILLNET2	99.70	3
Bergholz et al. [16]	99.46	4
Khonji et al. [20]	99.396	5
Toolan and Carthy [19]	99.31	6
phishGILLNET1	98.10	7
PILFER [10]	97.64	8
SVM with feature selection (Table 5)	95.90	9
Abu-Nimeh et al. [11]	90.24	10
Abu-Nimeh et al. [12]	NA	-
Gansterer and Pölz [15]	NA	-
Al-Momani et al. [21]	NA	-

compared to other competing methods. Moreover, the corpus used is publicly available and hence experiments could be reproduced. phishGILLNET3 also has the powerful feature of incorporating unlabeled data during training. phishGILLNET is domain neural. It can be employed to detect phishing attacks at social networking posts (Facebook, Twitter, etc.), instant messages, chat, blog posts, etc. As long as the content is available in text, MIME and HTML formats, this architecture can handle all of them. Thus, phishGILLNET is a significant research contribution to detect phishing attacks.

Abbreviations

AUC: area under receiver operating characteristic; EM: expectation maximization; HTML: hyper text markup language; IP: Internet protocol; MIME: multipart Internet mail extension; PILFER: phishing identification by learning on features of email received; POS: part-of-speech; PLSA: probabilistic latent semantic analysis; TDF: term document frequency; TEM: tempered expectation maximization; URL: uniform resource locator.

Competing interests

The authors declare that they have no competing interests.

Received: 1 August 2011 Accepted: 26 March 2012
Published: 26 March 2012

References

1. National Data–Deter. Detect. Defend. Avoid ID Theft.http://www.ftc.gov/bcp/edu/microsites/idtheft/reference-desk/national-data.html. Accessed 21 July 2011
2. Google Says Phishers Stole E-mail From US Officials, Others, PCWorld Business Center.http://www.pcworld.com/businesscenter/article/229202/google_says_phishers_stole_email_from_us_officials_others.html. Accessed 21 July 2011
3. DNSBL Information–Spam Database Lookup.http://www.dnsbl.info/. Accessed 21 July 2011
4. Snort–Home Page.http://www.snort.org/. Accessed 21 July 2011
5. H Kim, JH Huh, Detecting DNS-poisoning-based phishing attacks from their network performance characteristics. Electron Lett. **47**(11):656–658 (2011). doi:10.1049/el.2011.0399
6. ID Sender,http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx. Accessed 21 July 2011
7. DomainKey Library and Implementor's Tools.http://domainkeys.sourceforge.net/. Accessed 21 July 2011
8. H Kim, P Howland, H Park, Dimension reduction in text classification with support vector machines. J Mach Learn Res. **6**, 37–53 (2005)
9. SpamAssassin: Welcome to SpamAssassin.http://spamassassin.apache.org/. Accessed 19 July 2011
10. I Fette, N Sadeh, A Tomasic, Learning to detect phishing emails. in Proceedings of the 16th international conference on World Wide Web, vol. 1. (Banff, AB, Canada, 2007), pp. 649–656
11. S Abu-Nimeh, D Nappa, X Wang, S Nair, A comparison of machine learning techniques for phishing detection. in Proceedings of the eCrime Researchers Summit, vol. 1. (Pittsburgh, PA, USA, 2007), pp. 60–69
12. S Abu-Nimeh, D Nappa, X Wang, S Nair, Distributed phishing detection by applying variable selection using Bayesian additive regression trees. in IEEE International Conference on Communications, vol. 1. (Dresden, Germany, 2009), pp. 1–5
13. D Miyamoto, H Hazeyama, Y Kadobayashi, An evaluation of machine learning-based methods for detection of phishing sites. in Proceedings of the 15th International Conference on Advances in Neuro-Information Processing (Springer-Verlag, Heidelberg, 2009), vol. 1. (Auckland, New Zealand, 2008), pp. 539–546
14. F Toolan, J Carthy, Phishing Detection using Classifier Ensembles. (eCrime Researchers Summit, Tacoma, WA, USA, 2009)
15. WN Gansterer, D Pölz, E-mail classification for phishing defense. in Proceedings of the 31st European Conference on IR Research on Advances in Information Retrieval, (Springer-Verlag, Heidelberg, 2009), vol. 1. (Toulouse, France, 2009), pp. 449–460
16. A Bergholz, J-H Chang, G Paaß, F Reichartz, S Strobel, Improved phishing detection using model-based features. in Proceedings of the Conference on Email and Anti-Spam (CEAS), vol. 1. (Mountain View, California, USA, 2008), pp. 1–10
17. A Bergholz, JD Beer, S Glahn, MF Moens, G Paaß, S Strobel, New filtering approaches for phishing email. J Comput Secur. **18**(1):7–35 (2010)
18. A Bergholz, G Paaß, L D'Addona, D Dato, A real-life study in phishing detection. in Proceedings of the Conference on Email and Anti-Spam (CEAS), vol. 1. (Redmond, Washington, USA, 2010), pp. 1–10
19. F Toolan, J Carthy, Feature Selection for Spam and Phishing Detection. in eCrime Researchers Summit, vol. 1. (Dallas, Texas, USA, 2010), pp. 1–12
20. M Khonji, A Jones, Y Iraqi, A Study of Feature Subset Evaluators and Feature Subset Searching Methods for Phishing Classification. (CEAS'11, Perth, Australia, 2011)
21. AAD Al-Momani, TC Wan, K Al-Saedi, A Altaher, S Ramadass, A Manasrah, LB Melhim, M Anbar, An online model on evolving phishing e-mail detection and classification method. J Appl Sci. **11**(18):3301–3307 (2011). doi:10.3923/jas.2011.3301.3307
22. J Zhan, L Thomas, Phishing detection using stochastic learning based weak estimators. in IEEE Symposium on Computation Intelligence in Cyber Security, vol. 1. (Paris, France, 2011), pp. 55–59
23. J Yearwood, M Mammadov, D Webb, Profiling phishing activity based on hyperlinks extracted from phishing emails. J Social Netw Anal Mining. **2**(1):5–16 (2011)
24. G Xiang, J Hong, CP Rose, L Cranor, A feature-rich machine learning framework for detecting phishing web sites. ACM Trans Inf Syst Secur **14**(2):1–28 (2011). Article 21
25. M Khonji, Y Iraqi, A Jones, Lexical URL analysis for discriminating phishing and legitimate websites. in Proceedings of 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, vol. 1. (Perth, Australia, 2011), pp. 109–115
26. H Zhang, G Liu, TWS Chow, W Liu, Textual and visual content-based anti-phishing: a Bayesian approach. IEEE Trans Neural Netw. **22**(10):1532–1546 (2011)
27. CH Hsu, P Wang, S Pu, Identify fixed-path phishing attack by STC. in Proceedings of 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, vol. 1. (Perth, Australia, 2011), pp. 172–175
28. M Khonji, A Jones, Y Iraqi, A novel phishing classification based on URL features. in IEEE GCC Conference and Exhibition, vol. 1. (Dubai, UAE, 2011), pp. 221–224
29. L Wenyin, G Liu, B Qiu, X Quan, Anti-phishing through phishing target discovery. IEEE J Internet Comput. **16**(2):52–61 (2011)
30. SpoofGuard.http://crypto.stanford.edu/SpoofGuard/. Accessed 21 July 2011
31. A Cordero, T Blain, Catching phish: detecting phishing attacks from rendered website images.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.9084&rep=rep1&type=pdf. Accessed 26 July 2011

32. CallingID–Your Protection from Identity Theft, Fraud, Scams and Malware. http://www.callingid.com/Default.aspx. Accessed 21 July 2011
33. CloudMark.http://www.cloudmark.com/en/products/cloudmark-desktopone/index. Accessed 26 July 2011
34. Netcraft Anti-Phishing Toolbar.http://toolbar.netcraft.com/. Accessed 26 July 2011
35. FirePhish,https://addons.mozilla.org/en-US/firefox/addon/firephish-anti-phishing-extens/. Accessed 26 July 2011
36. eBay Toolbar.http://download.cnet.com/eBay-Toolbar/3000-12512_4-10153544.html?tag=contentMain;downloadLinks. Accessed 21 July 2011
37. IE Phishing Filter.http://support.microsoft.com/kb/930168. Accessed 21 July 2011
38. S Ab-Nimeh, S Nair, Bypassing security toolbars and phishing filters via DNS poisoning. in IEEE Global Telecommunications Conference, vol. 1. (New Orleans, Louisiana, USA, 2008), pp. 1–6
39. A Jain, V Richariya, Implementing a web browser with phishing detection techniques. World Comput Sci Inf Technol J. **1**(7):28–291 (2011)
40. E Lin, S Greenberg, E Trotter, D Ma, J Aycok, Does domain highlighting help people identify phishing sites? CHI 2011. Vancouver, BC, Canada http://survey.mailfrontier.com/survey/phishing_uk.html (2011). Accessed 21 July 2011
41. Y Chen, F Zahedi, A Abbasi, Interface design elements for anti-phishing systems. in Proc of the 6th International Conference on Service-Oriented Perspectives in Design Science Research (Springer-Verlag, Berlin, 2011), vol. 1. (Milwaukee, Wisconsin, USA, 2011), pp. 253–265
42. AP Felt, D Wagner, Phishing on mobile devices. Web 2.0 Security and Privacy Workshop. (2011)
43. MailFrontier Phishing IQ Test–UK Edition.http://survey.mailfrontier.com/survey/phishing_uk.html. Accessed 21 July 2011
44. SA Robila, JW Ragucci, Don't be a phish: steps in user education. in Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education, vol. 1. (Bologna, Italy, 2006), pp. 237–241
45. TN Jagatic, NA Johnson, M Jakobsson, F Menczer, Social phishing Commun ACM. **50**, 94–100 (2007)
46. NAG Arachchilage, M Cole, Design a mobile game for home computer users to prevent from phishing attacks. in IEEE International Conference on Information Society, vol. 1. (London, UK, 2011), pp. 485–489
47. SS Tseng, KY Chen, TJ Lee, JY Weng, Automatic content generation for anti-phishing education game. in IEEE International Conference on Electrical and Control Engineering, vol. 1. (Yichang, China, 2011), pp. 6390–6394
48. T Moore, R Clayton, The impact of public information on phishing attack and defense. Commun Strat. **81**(1):45–68 (2011)
49. Gillnet–Wikipedia.http://en.wikipedia.org/wiki/Gillnet. Accessed 25 July 2011
50. The Porter Stemming Algorithm.http://tartarus.org/~martin/PorterStemmer/. Accessed 26 July 2011
51. WordNet.http://wordnet.princeton.edu/. Accessed 26 July 2011
52. Google Suggest API.http://code.google.com/p/google-refine/wiki/SuggestApi. Accessed 26 July 2011
53. Levenshtein distance.http://en.wikipedia.org/wiki/Levenshtein_distance. Accessed 26 July 2011
54. T Hofmann, Probabilistic latent semantic indexing. in Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, vol. 1. (Berkeley, California, USA, 1999), pp. 50–57
55. T Hofmann, Unsupervised learning by probabilistic latent semantic analysis. Mach Learn. **42**, 177–196 (2001). doi:10.1023/A:1007617005950
56. Y Freund, R Schapire, A short introduction to boosting. J Jpn Soc Artif Intell. **14**, 771–780 (1999)
57. R Quinlan, C4.5: Programs for Machine Learning. (Morgan Kaufmann Publishers, San Mateo, CA, 1993)
58. WW Cohen, Fast effective rule induction. in 12th International Conference on Machine Learning, vol. 1. (Tahoe City, California, USA, 1995), pp. 115–123
59. L Breiman, Random forests. Mach Learn. **45**(1):5–32 (2001). doi:10.1023/A:1010933404324
60. V Vapnik, *The Nature of Statistical Learning Theory*. (Springer, New York, 1995)
61. M Sumner, E Frank, M Gall, Speeding up logistic model tree induction. in 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (Springer LNCS), vol. 3721. (Porto, Portugal, 2005), pp. 675–683
62. A Blum, T Mitchell, Combining labeled and unlabeled data with co-training. in Proceedings of the Workshop on Computation Learning Theory, (ACM), vol. 1. (Madison, Wisconsin, USA, 1998), pp. 92–100
63. K Nigam, R Ghani, Analyzing the effectiveness and applicability of co-training. in International Conference on Information and Knowledge Management, (ACM), vol. 1. (McLean, Virginia, USA, 2000), pp. 86–93
64. S Kiritchenko, S Matwin, Email classification with co-training. in Proceedings of CASCON, (ACM), vol. 1. (Toronto, Ontario, Canada, 2001), pp. 1–8
65. J Chan, I Koprinska, J Poon, Co-training with a single natural feature set applied to email classification. in Proceedings of the International Conference on Web Intelligence, vol. 1. (Beijing, China, 2004), pp. 586–589
66. X Wan, Co-training for cross-lingual sentiment classification. in Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, vol. 1. (Suntec, Singapore, 2009), pp. 235–243
67. A Kumar, H Daumé III, A co-training approach for multi-view spectral clustering. in Proceedings of the 28th International Conference on Machine Learning, ACM, vol. 1. (Bellevue, WA, USA, 2011), pp. 393–400
68. SpamAssassin Publiccorpus.http://spamassassin.apache.org/publiccorpus/.. Accessed 21 July 2011
69. PhishingCorpus.http://monkey.org/~jose/wiki/doku.php. Accessed 21 July 2011
70. Enron Email Dataset.http://www.cs.cmu.edu/~enron/. Accessed November 2011
71. SPAM Archive.http://untroubled.org/spam/. Accessed November 2011
72. PhishTank.http://www.phishtank.com. Accessed November 2011
73. SURBL URI Reputation Data.http://www.surbl.org. Accessed November 2011
74. Internet Security Threat Reports.http://www.symantec.com/business/threatreport/archive.jsp. Accessed November 2011

doi:10.1186/1687-417X-2012-1
Cite this article as: Ramanathan and Wechsler: **phishGILLNET—phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training.** *EURASIP Journal on Information Security* 2012 2012:1.