



Deliverable 6.2

Second Phase Prototype Report

Fraunhofer IAIS

Version 04

23. June 2007



IST 2006 027600

AntiPhish

Anticipatory Learning for Reliable Phishing Prevention

Specific Targeted Research or Innovation Project

2.4.3 Towards a global dependability and security framework

D 6.2 Second Phase Prototype Report

Due date of deliverable: M29 (31 May 2008)

Actual submission date: 19 June 2008

Start date of project: 01. Jan. 2006

Duration: 36 months

Lead Contractor for this Deliverable: Fraunhofer IAIS

Revision: 04

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history

Deliverable administration and summary		
Project acronym: AntiPhish		ID: IST-2006-027600
Document identifier:	AntiPhish-del-D62-SecondPhasePrototypeReport-d-v01	
Leading partner: Fraunhofer IAIS		
Report version: v04		
Report preparation date: 23 June 2008		
Classification: Public		
Nature: Report		
Author(s) and contributors: Gerhard Paaß, Andre Bergholz, Frank Reichartz, Siehyun Strobel, Anja Pilz, Anouar Boulal, Sebastian Glahn, in collaboration with all partners		
Status:		Plan
		Draft
		Working
	X	Final
		Submitted
		Approved

The AntiPhish © Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Date	Edited by	Status	Changes made
31.05.08	Andre Bergholz	Draft	draft
10.06.08	Andre Bergholz	Working	Refinements
18.06.08	Gerhard Paaß	Working	Integration of additional parts
23.06.08	Gerhard Paaß	Final	Including proposals from K.U.Leuven

Notice that other documents may supersede this document. A list of latest public AntiPhish deliverables can be found at the AntiPhish webpage at www.AntiPhishResearch.org/publications.

Copyright

This report is © AntiPhish Consortium 2008. Its duplication is allowed only in the integral form for anyone's personal use for the purposes of research or education.

Citation

Gerhard Paaß, Andre Bergholz, Frank Reichartz, Siehyun Strobel, Anja Pilz, Anouar Boulal, Sebastian Glahn (2008). Deliverable D6.2 Second Phase Prototype Report. Fraunhofer IAIS, AntiPhish Consortium , www.antiphishresearch.org.

Acknowledgements

The work presented in this document has been conducted in the context of the EU Framework Programme project IST 2006 027600 AntiPhish. AntiPhish is a 36-month project that started on January 1st, 2006 and is funded by the European Commission as well as by the industrial partners. Their support is appreciated.

The partners in the project are Fraunhofer Institute for Intelligent Analysis and Information Systems (FHG), Symantec LIRIC Limited (LIRIC), Symantec Ltd. (Symantec Ireland), TISCALI Services S.r.l. (Tiscali) and K. U. Leuven / ICRI-LIIR (K.U. Leuven). The content of this document is the result of extensive discussions within the AntiPhish© Consortium as a whole.

More information

Public AntiPhish reports and other information pertaining to the project are available through AntiPhish public web site under www.antiphishresearch.org.

Table of contents

Executive summary.....	6
1 Introduction	7
2 Machine Learning Framework	8
2.1 Interface to Streaming Email Data.....	8
2.2 Active Learning	8
2.3 Incorporation of New Features	9
2.4 Miscellaneous improvements.....	9
3 Data and Features.....	10
3.1 Feature Processing	10
3.2 Feature Selection.....	10
4 Evaluation Criteria	11
4.1 Requirements	11
4.2 F-Value.....	12
4.3 Cross validation and Comparison of Methods	12
5 Results of Experiments with the Second Prototype.....	14
5.1 Basic Results	14
5.2 Spam Emails.....	14
5.3 Evasion Classification	15
6 Conclusions and future work.....	18
References.....	19

Executive summary

This document describes and documents the second prototype of the AntiPhish Learning Platform (APL) which is the outcome of WP6. A corresponding milestone M6.2 was the software demonstration at the end of the first phase in May 2008. The report concentrates on the progress that has been made since milestone M 6.1. The second prototype will be further adapted to stream processing and will be deployed at Tiscali in to filter part of a real-world email stream.

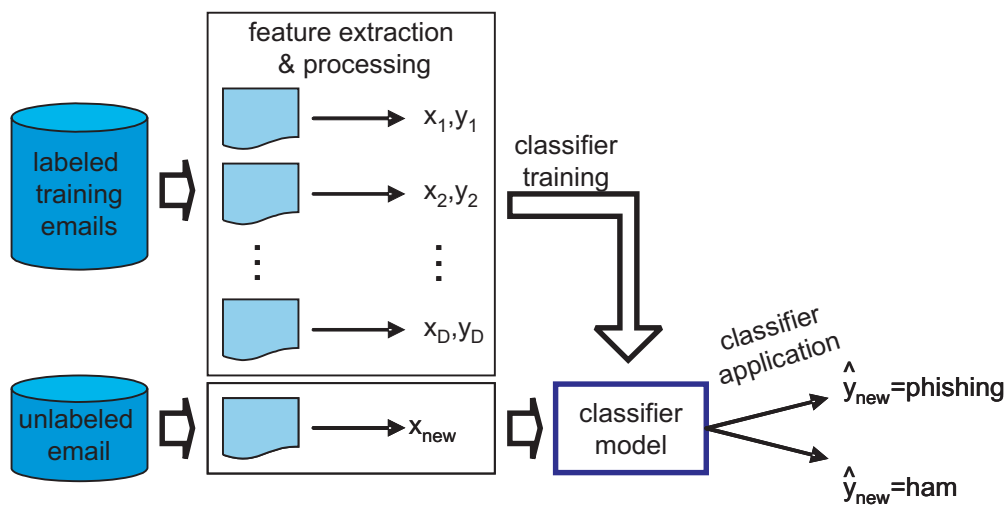
1 Introduction

Machine learning techniques, in particular automatic classification, have become popular in email and phishing detection. In contrast to manually constructed filter rules they automatically assess the relevance of input features $x = (x_1, \dots, x_m)$ (e.g., email characteristics) and establish a function to determine the desired classification y (e.g., phishing or nonphishing)

$$y = f(x, \gamma)$$

The vector of unknown parameter values γ is determined in a training phase in such a way that the relation between x and y in the observed data $(x_1, y_1), \dots, (x_D, y_D)$ is reproduced according to some optimization criterion. In the application phase the same features are extracted from a new incoming email. Based on these features and the model the classifier produces a classification of the email. The overall machine learning approach is summarized in Figure 1.

Figure 1: Machine Learning Approach for Email Filtering



2 Machine Learning Framework

The AntiPhish machine learning framework described in deliverable D6.1 has been extended to cope with more advanced machine learning algorithms, such as ensemble classifiers and active learning. Moreover, we provide a first interface to streaming email data to facilitate the AntiPhish deployment at the industrial partners. In addition various new email features provided in the feature extraction system of WP5 have been integrated. In the following sections we describe the improvements in more detail.

2.1 Interface to Streaming Email Data

We introduce a new operator StreamIterator that is similar to our previously introduced Loop operator. The stream iterator is able to process a stream produced by a listener. The listener monitors a source and produces one object at a time, which is passed to the stream iterator. Through this mechanism an undetermined number of emails can be processed one at a time. A reference implementation providing a listener for the file system has been included in the prototype.

2.2 Active Learning

Active learning is a technique to deal with the problem of too few labelled data. This is an important scenario in the case of email classification as it is impossible to manually label hundreds of thousands of emails. Active learning works as follows:

1. Given is a training set $S = L_0 \cup U_0$ composed of labelled data L_0 and unlabeled data U_0 . In addition there is a validation set V of labelled data.
2. For $i = 0, 1, 2, \dots$
 1. A model M_i is trained using the labelled data L_i .
 2. Apply M_i to the validation data V and measure performance.
 3. If the performance is good enough on a validation set: **BREAK**
 4. The model is applied to the unlabeled data U_i . It generates a classification together with a degree of confidence.
 5. A subset $W_i \subseteq U_i$ of borderline data classified with a low degree of confidence is identified and labelled manually.
 6. A new training set is formed $L_{i+1} = L_i \cup W_i$ and $U_{i+1} = U_i - W_i$.
3. The resulting model M_i may be applied to the test set T .

Active learning may be used in conjunction with semi-supervised learning. In semi-supervised learning the data classified with high degree of confidence is labelled automatically using the assigned class, in active learning the data classified with low degree of confidence is labelled manually. Active learning also leads to a faster

convergence to a good classification model and to smaller model as only the most important training examples are considered.

For the second prototype we provide first workflows for both semi-supervised learning and active learning. We plan to further optimize the parameters that guide the active learning process, such as confidence thresholds and stopping criteria. Furthermore, we will look into classifier-specific active learning algorithms, for example for support vector machines.

2.3 Incorporation of New Features

The second prototype includes a number of new email features not present in the first prototype. Most notable are image-related features, such as image distortion and logo detection. The second prototype provides an interface to the following new features:

1. **Logo detection:** We detect the presence of a predefined set of company logos, such as eBay or PayPal. As classifier input we record the total number of logos and distinct logos plus indicators for each individual logo.
2. **Image distortion:** We detect image obscuring techniques (see [Battista et al. 07]). As feature we extract indicators, such as the number of detected objects per area or the presence of large objects in images.
3. **Financial term detection:** We use conditional random fields to automatically detect financial terms. We record the number of detected financial terms plus an indicator of the presence of financial terms.
4. **DMC on structural data:** Inspired by the success of the dynamic Markov chain features based on the complete email text we employ this technique for the analysis of links and link texts. Again, we generate different models for ham and phishing emails and record the outputs of these models as input for our AntiPhish classifier.

The features are described in more detail in the deliverables D5.2 and D5.3.

2.4 Miscellaneous improvements

The second prototype improves the usability and handling of phishing email classification. We provide several operators that allow caching and external storage of intermediate results to facilitate experimental evaluation. We provide an interface to the statistical software package R, which enables rich evaluation and visualisation possibilities. Other external tools, such as RapidMiner and gnuplot, can also be incorporated into the analysis process.

In addition, the topic models and dynamic Markov chain algorithms have been re-implemented and now lead to a reduced runtime and model size.

3 Data and Features

For the evaluation of the second prototype we focus on the email data provided by Symantec. We have compiled a real-life dataset of 20000 ham and phishing emails over a period of almost seven months, from April 2007 to mid-November 2007. These emails were provided by our project partners. We assume a ratio of four to five ham emails for one phishing email. Specifically, our corpus consists of 16364 ham emails and 3636 phishing emails. For an additional experiment we added 20000 spam emails from the same time period to create an overall corpus of 40000 emails.

3.1 Feature Processing

It is not advisable to directly use the unmodified feature values as input for a classifier. We perform scaling and normalization. Scaling guarantees that all features have values within the same range. Many classifiers are based on distance measures, such as the Euclidean distance, which overemphasizes features with large values. We perform a Z-transformation to ensure that all features have an empirical mean of 0 and an empirical standard deviation of 1. Additionally, we normalize the length of the feature vectors to one, which is adequate for inner-product based classifiers.

3.2 Feature Selection

In practice, machine learning algorithms tend to degrade in performance when faced with many features that are not necessary for predicting the correct [Thrun et al. 91]. The problem of selecting a subset of relevant features, while ignoring the rest, is a challenge that all learning schemes are faced with. Feature selection can be understood as a search in a state space. In this space every state represents a feature subset. Operators that add and eliminate single features determine the connection between the states. Because there are $2^n - 1$ different non-empty subsets of n features, a complete search through the state space is impractical and heuristics need to be employed.

A common and widely used technique is the wrapper approach proposed by Kohavi and John [Kohavi John 97] a search algorithm uses the classifier itself as part of the evaluation function. The classifier operates on an independent validation set; the search algorithm systematically adds and subtracts features to a current subset. In our experiments, we apply the so-called best-first search strategy, which expands the current node (i.e., the current subset), evaluates its children and moves to the child node with the highest estimated performance. We combine the best-first search engine with compound operators [Kohavi John 97], which dynamically combine the set of best-performing children. The underlying idea is to shorten the search for each iteration by considering not only the information of the “best” child node (as described in the greedy approach above) but also the other evaluated children. More formally, by ranking the operators with respect to the estimated performance of the children a compound operator c_i can be defined to be the combination of the best $i+1$ operators.

4 Evaluation Criteria

4.1 Requirements

In Deliverable D22 Yearly Updated Requirements Specification Document a number of criteria for the AntiPhish system were specified:

- **Phishing False Negative Rate** - The number of phishing messages not blocked at policy enforcement points divided by the number of phishing messages sent through policy enforcement points.
The target for the prototype is 0.1, i.e. at most 10% of the Phishing messages may be not blocked.
- **Phishing False Positive Rate** - The number of legitimate messages blocked (as phishing) at policy enforcement points divided by the number of total messages sent through policy enforcement points.
The desired rate for the prototype is 0.01, which means that a fraction of 1% of all messages may be blocked ham emails.
- **Recall** – The ratio of phishing messages blocked divided by the number of total phishing messages.
This is just $1 - \text{Phishing False Negative Rate}$.
- **Precision** – The ratio of phishing messages blocked divided by the total number of messages blocked.
This value is difficult to evaluate as not only Phishing emails but also spam emails may be blocked. If all messages are considered together this value may be never reached as spam messages outnumber Phishing messages. In practice it is a difficult classification task to distinguish Phishing messages from spam messages.
- **Labelled Content Processing Message Throughput** – The volume of labelled data to be processed by the analysis system in generating blocking criteria to be disseminated to enforcement points, measured by the number of messages per second through the analysis system.
- **Labelled Content Processing Volume Throughput** – The volume of labelled data to be processed by the analysis system in generating blocking criteria to be disseminated to enforcement points, measured megabytes per second through the analysis system.
- **Latency** – The latency from the time a sample of a phishing is acquired from a labelled source to the time an effective rule can be deployed for blocking the phishing messages, measured in seconds.
- **Personnel** – The number of people required to operate the system at any given time to maintain the desired performance.

According to deliverable D22 the different systems should fulfil the criteria shown in the following table. Requirements are given in context of current production system performance and required performance of prototype and projected production systems, along with projected goals for a production system.

Table 1: Performance Criteria for the Prototype

Criterion	Prototype	Production	Goal
Phishing False Negative Rate	0.10	0.05	0.01
Phishing False Positive Rate	0.01	0.001	1/1M
Recall	0.90	0.95	0.99
Precision	0.95	0.99	0.999999
Labelled Content Processing Message Throughput [/sec]	12	1,200	10,000
Labelled Content Processing Volume Throughput MB/sec	0.12	12	100
Latency [sec]	3,000	300	30
Personnel [Persons]	1	20	5

4.2 F-Value

An important summary measure is the F-value, which is a compromise between precision and recall. It is defined as the weighted harmonic mean of precision and recall

$$F = \frac{2 * precision * recall}{precision + recall}$$

4.3 Cross validation and Comparison of Methods

The performance of classifiers may be estimated by an independent test set. A prerequisite is that the examples in the training and test set are mutually independent and follow the same distribution as the examples in the training data. In this case the mean of the estimated performance figures, e.g. precision, recall or f-value, is an unbiased estimator of the true performance. However, it is difficult to estimate the variance around the true value. The APL framework provides workflows for training and testing classifiers by a holdout set.

In **k-fold cross-validation**, the original sample is partitioned into k subsamples. Of the k subsamples, a single subsample is retained as the test set data for testing the model, and the remaining $k - 1$ subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the test set. The k results from the folds then can be averaged (or otherwise combined) to produce a single estimation. There is a workflow for executing crossvalidation in APL. This can be done on a single compute node or in parallel on different computers.

The different performance values obtained from a cross-validation vary because of the random selection of test and training sets and are distributed around the true value. A number of papers have investigated a way to estimate a confidence interval for the true value. Based on theoretical analysis and a large number of empirical evaluations [Bouckaert Frank 2004] propose the following **r-times repeated k-fold cross-validation procedure**, where $r > 0$ and $k > 1$. For fold $i \leq k$ and run $j \leq k$ we get differences $x_{ij} = a_{ij} - b_{ij}$ between the performance criteria for two classifiers to be compared, e.g. their f-values. Then we define estimates for the mean and variance by

$$m = \frac{1}{k * r} \sum \sum x_{ij} \quad \hat{\sigma}^2 = \frac{1}{k * (r - 1)} \sum \sum (x_{ij} - m)^2$$

As the folds are not independent we may not use $\hat{\sigma}^2$ directly in a significance test. [Bouckaert Frank 2004] propose an inflation factor which increases the variance. They use the following test statistic

$$t = \frac{\frac{1}{k * r} \sum \sum x_{ij}}{\sqrt{\left(\frac{1}{k * r} + \frac{n_2}{n_1} \right) \hat{\sigma}^2}}$$

where n_1 is the number of instances used for training and n_2 the number of instances used for testing. The test is called “corrected repeated k-fold cv test”. Theoretical analysis and many empirical experiments show that this procedure can reliably compare the quality of different classifiers [Bouckaert Frank 2004]. APL provides a workflow to evaluate this test.

5 Results of Experiments with the Second Prototype

5.1 Basic Results

In deliverable D6.1 we have shown that by the introduction of model-based features we were able to substantially outperform previous results by Fette et al. [Fette et al. 07] on a public dataset. On this dataset we achieved with our best feature combination an F-measure of 99.46% compared to 97.64% by Fette et al., which corresponds to a reduction in error of more than 75%. The details are described in the paper [Bergholz et al 08b] accepted for publication at CEAS08.

For the real-life data provided by Symantec we achieve excellent results. The table summarizes the results for classification. We have calculated the statistical significance in a paired t-test and conclude that the improvement of using all features vs. the basic features is significant with a *t*-value of 8.70 with nine degrees of freedom, which makes the improvement significant with a probability of more than 0.999.

Table 2: Results for Phishing Detection on a Public Corpus

Features	Precision	Recall	F-Measure
Without Topic, DMC	98,67%	94,76%	96,67%
All features	99,94%	99,69%	99,82%
Feature Selection	99,97%	99,81%	99,89%

We used feature selection on each individual fold to make the result more comparable to related work. Note that we no longer can use 90% of the data for training in each individual fold, because we reserve 30% of the training data for the feature selection process, i.e., for the validation of individual models based on different feature sets. As shown in the table we achieved an even better result using fewer features and less training data. Again, the improvement over the basic features is statistically significant with a probability of more than 0.999.

We also observed which features were selected in the process. Besides our model based features (i.e., DMC on text, DMC on links and topic features) some structural features (total number of body parts, number of discrete and composite body parts), some link features (total number of links, number of deceptive links, number of image links) and the wordlist features placed among the selected ones. This seems reasonable as the structural features are complementary to the purely content-based advanced features.

5.2 Spam Emails

To test the effect of spam emails we added 20000 spam emails from the same time period to our corpus to create an overall corpus of 40000 emails. The spam emails were merged with the phishing class to form an overall ``unwanted" class. We ran an

unchanged version of our program on this extended corpus using the features from the feature selection and observed that the performance deteriorates only very slightly to an F-measure of 98.48%. Details are in the table below.

Table 3: Results for Phishing Detection in the Presence of Spam

Corpus	Precision	Recall	F-Measure
Without spam	99,97%	99,81%	99,89%
With spam	99,01%	97,96%	98,48%

Note that in general spam classification may require different features than phishing filtering, as spammers use different salting methods. In our classifier we only used an untrained version of SpamAssassin as a spam indicator. Nevertheless the performance remained quite high. If a dedicated spam filter would be used as an additional feature the performance could be considerably increased.

5.3 Evasion Classification

Salting is the intentional addition or distortion of content, usually in spam emails, aimed to obfuscate or evade automatic filtering. There exists surface salting, e.g., images containing random pixel dots, and hidden salting, e.g., text in invisible colour. We have developed a hidden salting simulation model based on cognitive theory [De Beer Moens 08]. For an input email the model simulates which text will actually be seen by the user based on a fixed set of known salting tricks. In addition, the model indicates how many times each of the known salting tricks was used in the email. The results of this work are described in more detail in the paper [Bergholz et al 08a] accepted for publication in CEASo8.

It was proposed in the project consortium to close the loop and address the problem of identifying when our hidden salting simulation system fails, most likely because some spammer has come up with a yet unknown salting trick. To this end, we try to detect differences between the simulated perceived text as generated by our hidden salting simulation system and the message text as obtained by applying OCR to the email message rendered by some rendering engine.

We develop robust text comparison metrics that allow us to compare the simulated perceived message text to the true message text obtained by applying OCR to the rendered email. We use the distances computed with these metrics' as features to train a classifier. An outlier as detected by this classifier would be a candidate email for containing a new, unseen salting trick. We automatically determine the threshold of when to consider an email as outlier. We perform experiments to simulate the detection of a new trick by disabling the detection of a known trick. On publicly available email data we are able to achieve good results for the tested tricks.

In the feature extraction phase we intercept requests for drawing text primitives, and build an internal representation of the characters that appear on the user screen.

This representation is a list of attributed glyphs (i.e., positioned shapes of individual characters, with rendering attributes and any concealing shapes). Glyphs are listed in their compositional order (i.e., the order in which they are drawn according to the source text). Then, we test for glyph visibility (i.e., which glyphs are seen by the user) according to the following glyph visibility conditions:

Clipping The glyph is drawn within the physical bounds of the drawing clip.

Concealment The glyph is not concealed by other glyphs or shapes.

Font Colour The glyph's fill colour contrasts well with the background colour.

Glyph Size The glyph's size and shape is sufficiently large.

Failure to comply with any condition results in an invisible glyph and is an indication of hidden salting. We eliminate all invisible glyphs (i.e., retain only what is expected to be perceived by the user).

We propose three measures to compare the simulated perceived text and the true message text of an email:

1. **Length:** The difference in normalized text length. We normalize whitespaces and simply calculate the difference between the two lengths. This is simple and robust with respect to text order changes and OCR errors.
2. **TOC:** The tolerant overlap coefficient, a combination of token-based and edit-distance like measures. We use an edit-distance to match tokens with some degree of freedom, then use a token based measure for overall text distance.
3. **Complexity:** Difference of "information content": The Kolmogorov complexity is a measure of the computational resources needed to specify the string. We use Lempel-Ziv compression for its approximation.

In our experiments we simulate the detection of a new salting trick by disabling the detection of one of the tricks implemented in our salting simulation system.

The simulated perceived text is generated by our hidden salting simulation system. We obtain the true message text of an email by rendering it using standard Java rendering and then OCRing the rendered email using the publicly available OCR engines gocr and ocrad.

As a classifier, we train a one-class SVM. The setting is as follows. The training set contains only emails, which do not contain the disabled trick. They represent the "one class" of so far "normal" emails, i.e., emails that contain no or only known salting tricks. The test set then contains both emails with and without the disabled trick. The classifier marks emails as outliers, which indicates that they are not in the same class as all the emails from the training set. In other words, these emails are "non-normal" or suspicious, which indicates that they may contain a previously unseen salting trick.

The following table shows the results we obtain for the detection of two salting tricks, font colour and font size.

Table 4. Results for Salting Trick Detection

Trick	OCR-engine	FPR %	FNR %	F2 %
Font colour	gocr	7.53	17.85	84.15
Font size	gocr	9.18	10.10	87.08
Font colour	ocrad	20.43	11.41	77.08
Font size	ocrad	30.61	10.85	69.11

The F2-measure weights outlier recall twice as high as outlier precision. The table suggests that we achieve better results using the goocr engine than using the ocrad engine. Though the evaluation in f-measure does not differ very much we can observe significant differences in the false positive and the false negative rate. As we pointed out earlier, the false positive rate is of particular importance. Currently we are experimenting with a commercial OCR engine, which may give much better results.

6 Conclusions and future work

The result described in the preceding sections shows that the AntiPhish Learning Platform (APL) fulfils the requirements specified in deliverable D2.2. The computing environment has been evaluated many times and is adequate to implement the real-world filtering application for the prototype deployment at Tiscali.

It has turned out, however, that the publicly available phishing corpora and the email collection provided by Symantec do only contain specific types of emails due to privacy requirements. Therefore the filtering of real world email during the last phase of the project requires a thorough retraining of all filters and perhaps the inclusion of new features, which up to now were not necessary. Therefore the new developments are required for the next months.

References

- [Battista et al. 07] Battista Biggio, Giorgio Fumera, Ignazio Pillai, Fabio Roli: Image Spam Filtering Using Visual Information. ICIAP 2007: 105-110
- [Bergholz et al 08a] André Bergholz, Gerhard Paaß, Frank Reichartz, Siehyun Strobel, Marie-Francine Moens, Brian Witten (2008): Detecting Known and New Salting Tricks in Unwanted Emails. Accepted for publication at Conference on Email and Anti-Spam (CEAS) 2008.
- [Bergholz et al 08b] André Bergholz, Gerhard Paaß, Frank Reichartz, Siehyun Strobel, Jeong-Ho Chang (2008): Improved Phishing Detection using Model-Based Features. Accepted for publication at Conference on Email and Anti-Spam (CEAS) 2008.
- [De Beer Moens 08] J. De Beer and M.-F. Moens (2008). Challenging hidden text salting in digital media. Submitted for publication.
- [Bouckaert Frank 04] R. R. Bouckaert and E. Frank. Evaluating the replicability of significance tests for comparing learning algorithms. In Proceedings of the Pacific- Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 3–12, 2004.
- [Cortes Vanik 1995] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:pp. 273 - 297, 1995.
- [Fette et al. 07] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In Proceedings of the International World Wide Web Conference (WWW), pages 649–656, 2007.
- [Kohavi John 97] R. Kohavi and G. John. Wrappers for feature subset selection. AI Journal, 97(1-2):273–324, 1997.
- [Thrun et al. 91] S. Thrun et al. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, Pittsburgh, PA, 1991.
- [Tsochantaridis et al. 2004] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, Yasemin Altun: Support vector machine learning for interdependent and structured output spaces. Proc. ICML 2004.
- [Vapnik 1995] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.