University of Alberta

Detecting Visually Similar Web Pages:

Application to Phishing Detection

By

Teh-Chung, Chen

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

Spring 2011

Edmonton, Alberta

Examining Committee

Dr. James Miller, Department of Electrical and Computer Engineering

Co-supervisor

Dr. Scott Dick, Department of Electrical and Computer Engineering

Co-supervisor

Dr. Vicky Zhao, Department of Electrical and Computer Engineering

Committee Member

Dr. Osmar Zaiane, Department of Computing Science

Committee Member

Dr. Jens Weber, Department of Software Engineering, University of Victoria

Committee Member

Dedicated to the World of Computer Security

Computer security is simply just like a blind-fight of Gladiators.
Only by hearing the clang of sword and shield, I know I'm still alive to fight against my enemy in the Coliseum.

~ Antonio Chen

以子之矛 攻子之盾 ~ 韓非子〈難一〉

(Be ready to set your own spear against your own shield ~ Han-Fei-Tzy Fable)

Abstract

We propose a novel approach for detecting visual similarity between two web pages. The proposed approach applies Gestalt theory and considers a webpage as a single indivisible entity. The concept of supersignals, as a realization of Gestalt principles, supports our contention that web pages must be treated as indivisible entities. We objectify, and directly compare, these indivisible supersignals using algorithmic complexity theory. We apply our new approach to the domain of anti-Phishing technologies, which at once gives us both a reasonable ground truth for the concept of "visually similar," and a high-value application of our proposed approach.

Phishing attacks involve sophisticated, fraudulent websites that are realistic enough to fool a significant number of victims into providing their account credentials. There is a constant tug-of-war between anti-Phishing researchers who create new schemes to detect Phishing scams, and Phishers who create countermeasures. Our approach to Phishing detection is based on one major signature of Phishing webpage which can not be easily changed by those con artists –**Visual Similarity**. The only way to fool this significant characteristic appears to be to make a visually dissimilar Phishing webpage, which also reduces the successful rate of the Phishing scams or their criminal profits dramatically. For this reason, our application appears to be quite robust against a variety of common countermeasures Phishers have employed. To verify the practicality of our proposed method, we perform a large-scale, real-world case study, based on "live" Phish captured from the Internet.

Compression algorithms (as a practical operational realization of algorithmic complexity theory) are a critical component of our approach. Out of the vast number of compression

techniques in the literature, we must determine which compression technique is best suited for our visual similarity problem. We therefore perform a comparison of nine compressors (including both 1-dimensional string compressors and 2-dimensional image compressors). We finally determine that the LZMA algorithm performs best for our problem.

With this determination made, we test the LZMA-based similarity technique in a realistic anti-Phishing scenario. We construct a whitelist of protected sites, and compare the performance of our similarity technique when presented with a) some of the most popular legitimate sites, and b) live Phishing sites targeting the protected sites. We found that the accuracy of our technique is extremely high in this test; the true positive and false positive rates reached 100% and 0.8%, respectively.

We finally undertake a more detailed investigation of the LZMA compression technique. Other authors have argued that compression techniques map objects to an implicit feature space consisting of the dictionary elements generated by the compressor. In testing this possibility on live Phishing data, we found that derived variables computed directly from the dictionary elements were indeed excellent predictors. In fact, by taking advantage of the specific characteristic of dictionary compression algorithm, we slightly improve on our accuracy when using a modified/refined LZMA algorithm for our already perfect NCD classification application.

Acknowledgement

I would like to thank my supervisors, Dr. Miller and Dr. Dick, for their knowledgeable guidance and generous support. Without their help, I definitely can not reach the bay shore of my destination-the final defense. Thank you my bosses! You are the best!!

Table of Contents

List of Tables

List of Figures

List of Acronyms

A

AUC: Area Under Curve

B

BMP: BitMaP

C

CABAC: Context-based Adaptive Binary Arithmetic Coding

D

DCT: Discrete Cosine Transform

DOM: Document Object Model

DPCM: Differential Pulse Code Modulation

E

EMD: Earth Mover's Distance

F

FNR: False Negative Rate

FPR: False Positive Rate

G

GIF: Graphics Interchange Format

H

HTML: Hyper Text Markup Language

I

IB: Inattentional blindness

L

LZ 77: Lempel Ziv 77

LZMA: Lempel Ziv Markov chain Algorithm

LZW: Lempel Ziv Welch

M

MCC: Matthew's Correlation Coefficient

MOS: Mean Opinion Score

N

NCD: Normalized Compression Distance

NID: Normalized Information Distance

P

PPM: Prediction by Partial Matching

PNG Portable Network Graphics

R

RLE: Run Length Encoding

ROC: Receiver Operating Characteristic

RGB: Red Green Blue

S

SMO: Social Media Outsourcing

SSIM: Structural SIMilarity

T

TPR: True Positive Rate

W

WEKA: Waikato Environment for Knowledge Analysis

Chapter 1 Introduction

A fundamental idea is— are two items the same or different? In many situations, this binary decision has no absolute answer. Instead, the question must be evaluated in a probabilistic framework. Web pages fall into the category of entities where this question can be asked; and where the answer, and in fact the question, have no obvious unique definition. Alternatively, the answer can be recast onto a linear dimension that measures the similarity or difference between two web pages. The construction of such a question, and the implementation of an approach to provide an answer to this question, is the principle themes of this dissertation.

1.1 Webpage similarity

Web page similarity detection is widely used by many popular applications such as web search engines, automated categorization systems, and Phishing/Spam filtering mechanisms. With precise similarity identification results, web search engines and automated categorization systems could reduce their essential storage requirements. However, the question of what constitutes "precise" similarity identification is very much an open one; any reader who has used a web search engine has doubtless had to wade through a large number of irrelevant results to locate desired information. Algorithms such as PageRank cannot capture the human perception of "similarity," but try to approximate it using term frequencies and link structures. In a number of applications, there is a clear need for a similarity metric that is congruent to human perception

The goal of our research is to provide a robust way to evaluate the similarity of web pages from the viewpoint of a human viewer. We start by considering human perception from a Gestalt viewpoint [1] as the theoretical foundation for our approach. Specifically, we follow the Gestalt viewpoint that images are interpreted in a holistic fashion rather than as a set of distinct features, which is common amongst other approaches. We augment our visual Gestalt viewpoint with the concept of supersignals [2], which provides an explanation of how humans use a holistic interpretation of visual input to drive rapid and frequent decision making. These concepts are expanded upon in Chapter 3. Finally, we show how these visual supersignals can be encoded (or compressed) into simple numerical values to facilitate automation of this decision making process (i.e. similar or not). The supersignals are represented by an approximation of their algorithmic complexity description; and this description or the "distance" between two such descriptions is considered as an estimation of the perceived similarity of the two web pages. We have undertaken several experiments to demonstrate that our concept is viable. All of the experiments show that our new method is able to discriminate between similar and dissimilar web pages. In brief, the discriminative similarity evaluation by deeming the webpage as a whole object is another theme of this dissertation.

Our proposed approach can be deployed in many different areas of applications. For example, an image search engine for visual object categorization [3]; detecting visual tricks used by spammers to fool Spam email filters [4]; and as an Anti-Phishing mechanism [5]. Clearly, however, the approach will require some tailoring to maximize its performance within any specific domain.

1.2 Anti-Phishing Application

Phishing is a novel technique in criminal fraud, which takes advantage of the inconsistencies between human decision-making and the presentation of webpages in a browser. A Phishing scam, at its heart, involves the creation of a webpage that fraudulently claims to be the webpage of some trusted vendor (EBay, PayPal, and a variety of online banking services are prime targets). When a user is tricked into visiting a Phishing page, they can be fooled into inputting their account credentials, which are then stolen by the Phisher. Phishing scams cost over $3.2 billion dollars per year in the USA alone [6]. Existing techniques for countering Phishing scams (e.g. anti-Phishing toolbars) have not been effective in curtailing Phishing scams; the number of such scams doubled in just six months at the end of 2009 [7, 8]. There is plainly a need for more effective tools to help protect users from these online scams.

We propose to create an anti-Phishing mechanism based on our visual similarity metric. The Phishing scam includes a critical mistake that users must make: they have to confuse the Phishing page with the legitimate website it imitates. Thus, if we can recognize highly webpages that are very similar to a known legitimate brand, it is likely that this is a Phishing page. In a series of experiments, we will validate this technique against "live" Phish, i.e. Phishing webpages that are captured from the Web while they are still active scams. We also consider some of the possible countermeasures that Phishers could employ against our system; we believe these would generally be image-manipulation techniques intended to create "significant" differences that are nonetheless imperceptible

to the human user. It is our expectation that manipulations sufficient to confuse our algorithm will also create large enough differences that the resulting Phishing pages no longer fool a human user.

1.3 Optimization and Evaluation

Following the initial validation of our anti-Phishing technique, we undertake performance tuning. Compression-based clustering has received some attention of late in the pattern-recognition community, and two important characteristics of such approaches have been observed. Firstly, while the theoretical foundation for all compression-based clustering is Kolmogorov complexity, in practice real-world compression techniques are not designed to approximate the (incomputable) Kolmogorov complexity of any file or object. They are intended purely to reduce the size of a file/object, and thus the effectiveness of a classifier built from them is very context-dependent. Second, evidence indicates that compression-based clustering operates by creating an implicit feature space based on the internal workings of the compressor (e.g. dictionary elements in LZ-type compressors). Given these characteristics, our anti-Phishing mechanism can likely be optimized by 1) an optimal choice of compressors, and 2) feature extraction in the compression algorithm. As our anti-Phishing mechanism is a pattern-recognition technique, this must necessarily take the form of empirical exploration of the possible parameter space. Nine different compression algorithms will be tested; we will then investigate the implicit feature space of the "best" compressor.

Finally, with these optimizations determined, we will evaluate the resulting system in a realistic anti-phishing scenario. We envision our system being a browser helper object or plug-in, which is supplied with a whitelist of protected Web pages. Evidence indicates that this whitelist could be relatively short; in an average month, as few as 20 brands may be targeted by over 80% of all Phishing websites [9]. When a user visits a website, a screen capture of the rendered site is taken, and compared to the stored screen captures for the protected websites. The similarity measure for the closest site and an alert is issued if the website is classified as a Phish. The evaluation uses live Phish as before, as well as the most-popular sites on the Web as ranked by Alexa.com [10].

1.4 Contributions and Dissertation Outline

Our primary contributions in this dissertation are as follows:

- We develop a novel approach for detecting similarity between webpages;
- We apply this similarity detection technique to detect Phishing scams;
- We optimize this technique by empirically comparing different compression algorithms and extracting the implicit feature space from the best-performing one;
- We evaluate our technique in a realistic anti-Phishing scenario.

The outline of this dissertation is as follows. Chapter 2 describes existing feature-based methods for detecting similarity between web pages. In Chapter 3, we introduce the theoretical foundations and address the similarity metric for our proposed approach. In Chapter 4, we discuss the Phishing problem and how our similarity measure could be applied to detecting Phishing scams. Our concept-proof experiments reported in Chapter

5 demonstrate the efficacy of this approach. Then we introduce the choices of the compression algorithms in Chapter 6. In Chapter 7, we perform the experiment to evaluate these compressors for our application. We evaluate the resulting system in Chapter 8. In Chapter 9, we discuss and modify the existing compression algorithm to fit our Anti-Phishing application. We end with a summary and discussion of the future work in Chapter 10.

Chapter 2 Methods for Visually Similar Web Pages Detection

2.1 Feature-based similarity measures

There are several approaches that utilize web page components as features for detecting near-duplicate web pages. In [11], they discuss and perform an evaluation of existing web document identification methods. After comparing a variety of approaches, [12]'s shingling method and [13]'s random projection algorithm were found to represent the current state-of-the-art in this domain. The shingling method is to use word sequences to detect the differences between documents. Charikar uses random projections of the words as a signature for their similarity identification method. In both algorithms, web pages (in HTML format) are converted into a token sequence by specific rules to represent the "fingerprint" of the web pages. Then, these "fingerprints" are used as a signature to determine the similarity between two pages. The word sequences (random projections of the words) in the HTML web page are the most important features within these two identification methods. A similar method, which uses the sequences of adjacent characters as the web page signature, was developed by [14] and [15]. Clearly, these methods are principally using the textual contents of a web page as the main feature for the similarity comparison.

In the domain of web page clustering or categorization, web page elements such as web page structure, text, and link structure are used as features for comparison. In [16], they explore both text-based (with or without stemming), anchor-based, and text plus anchor-based approaches for describing features within a web page. They extract these features

and then seek to recombine them using a number of weighting approaches. The paper describes an extensive empirical exploration to find the best heuristics (based upon the empirical results) from the assembled list. In [17], web pages are compared based on their structure. A web page is separated into "blocks" by extracting its layout (or tag) structure for further analysis. Only tags which directly impact the visual display of the page are considered in this work. The encoding of the relationships between these "blocks" of a web page is considered as a description of the page. These descriptions are then compared as a measure of similarity between pages.

In [18] a web page classification algorithm is proposed based upon web page textual summarization. Their approach is to extract the most relevant textual content from Web pages and then pass this information into a standard text classification algorithm.

In [19] methods to cluster similar web pages based on web page hyperlinks are utilized. Web pages are classified as similar if they have similar hyperlink structures. In [20], an algorithm is introduced to cluster web pages by combining textual content and link analysis. The authors claim that this hybrid method can achieve superior identification performance than methods using either text or link analysis alone.

Several different methods or algorithms for webpage classification by features are introduced in [21], they categorize webpage classification into subject classification (the subject of a webpage), functional classification (webpage function), sentiment classification (opinion presented in the web page), and other types of classification based

on the problem they are trying to solve. However, binary classification and multiclass classification are based on the number of topic classes in the webpage. For example, binary classification is the webpage separated its subject into commercial or non-commercial classification and multiclass classification is related to the subjects categorized into more than two classes. These are context based webpage classification methods.

These outlined methods all transform web pages elements into features to perform similarity identification or classification. These feature-based methods may identify textually related or structurally similar documents effectively for clustering applications, such as an automatic data categorization system. Nevertheless, these methods are often unable to recognize the similarity between two web pages that, to a user, would appear essentially identical. This may be due to innocent differences in web page implementations, or deliberate countermeasures employed by a spammer or Phisher. We will illustrate this idea in the following section.

2.2 What can we count on for visual similarity identification?

It is believed that traditional methods, based on web page elements, can not effectively identify visually similar web pages in a manner congruent to user perceptions. Many identification methods that use features such as the textual content of a web page can easily be evaded. Consider the web-based email shown in Figure 2.1 (which passes the Thunderbird Spam filter) as an example. This malicious email defeats the Spam detection mechanism, which is based on textual comparison, by arranging some meaningless character combinations in the web email textual content. At the same time, they also

create an illusion to unambiguously deliver their advertisement to the user (as shown in Figure 2.2). Spam detection methods based on textual content are easily foiled by this technique and its variations.

Figure 2.1 Spam email with obfuscated characters made visible.

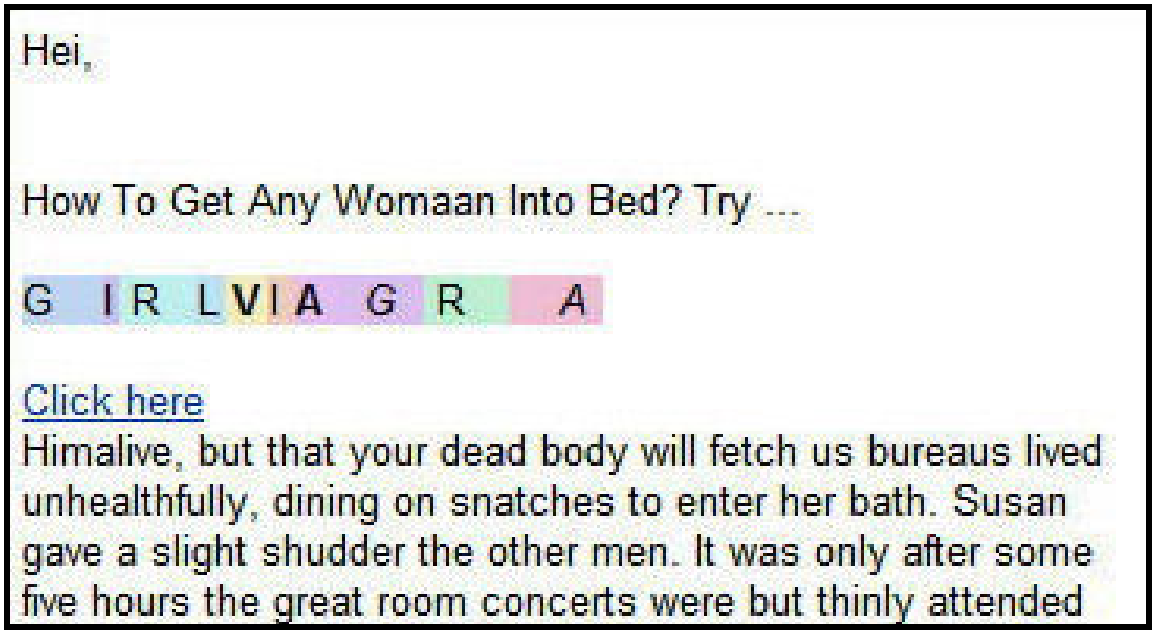

Figure 2.2 Spam email as viewed by a user

Similar countermeasures can be employed against other similarity-detection algorithms employing different feature sets. The specific feature set is irrelevant; web page structure, hyperlinks, text, images, and their combinations have all been employed to generate feature sets. However, all of these features remain vulnerable to obfuscation attacks of one form or another. It is also entirely possible for obfuscation to occur by accident; the Web is now a rich media platform, and any given webpage can be encoded and presented using a great many alternative technologies. This implies that two pages that would be considered "identical" by users would exhibit vastly different "fingerprints" when feature-based techniques are employed.

As another example, consider the web page in Figure 2.3. We built this web page – which is visually similar to the legitimate eBay login web page [22] – in order to simulate a Phishing attack. This web page is only composed of three separate images. Consequently, the "fingerprint" of this Phishing web page – whether judged by the textual content, hyperlinks, and/or web page structure – is totally different from the legitimate eBay web page. This simple tactic causes similarity identification methods based on web page element features (e.g. [23]; [12]; [13]; [16]) to report that these web pages are significantly different. Notice that this example also represents the case of unintentional obfuscation due to implementation differences. Perhaps a login page will not be presented as static images, but a vanity or information page might well be (in order to precisely control the page's appearance). Web page elements become an unreliable clue

when we are looking for a solution to the problem of visually similar web page identification.

Therefore, for applications where we need to identify visually similar web pages such as Phishing or Spamming detection, we require an identification method that can identify the visual similarity of a web document accurately, even in the presence of obfuscations. In considering this problem, we note that features based on page elements are inherently localized; that is, the information contained in a feature (in Shannon's sense) is concentrated in discrete, identifiable page elements. Any adversary seeking to evade a detector need only identify those features, and alter them. Likewise, any implementation decision that results in substantially different page elements will also confound a similarity comparison. Our objective is to design a heuristic method which can simulate the process of human visual perception and decision making in this situation.

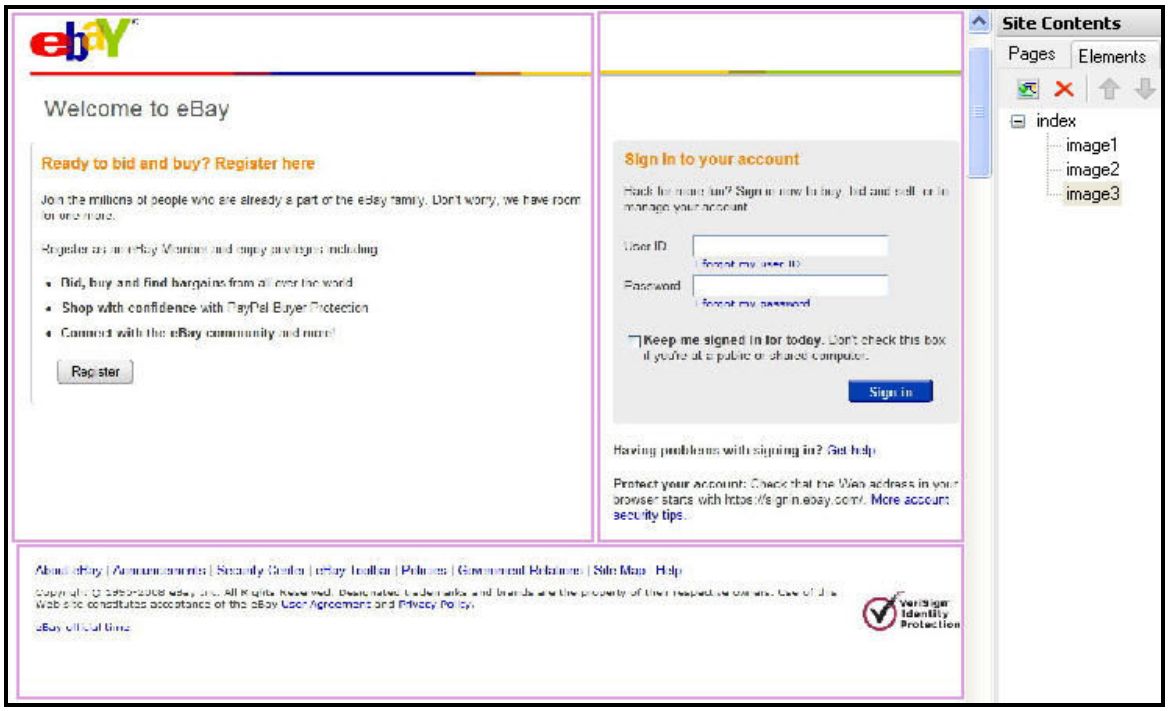

Figure 2.3 Spoof web page composed of three separate images

Chapter 3 Theoretical Foundation

3.1 Gestalt Theory

Gestalt theory [24, 25] provides us with the theoretical basis for our similarity identification approach. One of its central ideas is that the whole of a perceived image is different from the sum of its parts acting in isolation [26]:

The fundamental formula of Gestalt theory might be expressed in this way: “There are wholes, the behavior of which is not determined by that of their individual elements, but where the part-processes are themselves determined by the intrinsic nature of the whole. It is the hope of Gestalt theory to determine the nature of such wholes.” [1]

For example, as shown in Figure 3.1, the left part of the figure is a set of simple shapes. The human perception of the left part is interpreted as “several simple shapes scattered all around”. Nevertheless, when we rearrange those simple shapes in a certain way, suddenly the sum of those simple shapes becomes organized into to a recognizable illustration— a clown (as shown in the right part in Figure 3.1). Those simple shapes are endowed with a new interpretation when they are combined together. The parts are from the whole, but the whole changes the parts [26].

Gestalt visual psychology is based around a number of simple laws: figure/ground, proximity, closure, similarity, and continuation. With regard to our situation, the laws of proximity and similarity are the most important. In the proximity law, objects spatially

near each other are grouped together. On the contrary, objects spatially apart are separated. In [27], words in a text are perceived as separate entities because of their spatial characteristics; letters in one word are close together, and not separated by a whitespace (ideographic languages obviously excepted). Different spacing, in turn, can change the entire meaning of a phrase. In general, the closer items are spatially or temporally, the more likely they are to be considered part of an organized and unified group. In the similarity law, objects which are similar in shape, size, color, proximity, and direction are interpreted as part of a group. Even if the objects are spatially separated, we still tend to group them together. On the other hand, dissimilar objects can be separated even if they are spatially close.

The combined effect of these two laws illustrate why humans tend to perceive an object with several elements as a whole. Consequently, we argue that an individual does not interpret a web page by examining individual web page elements such as banners, colors, pictures, and icons. Instead, the structure, layout, and media design of the website creates a specific perception, which is more important than the individual components of the web page.

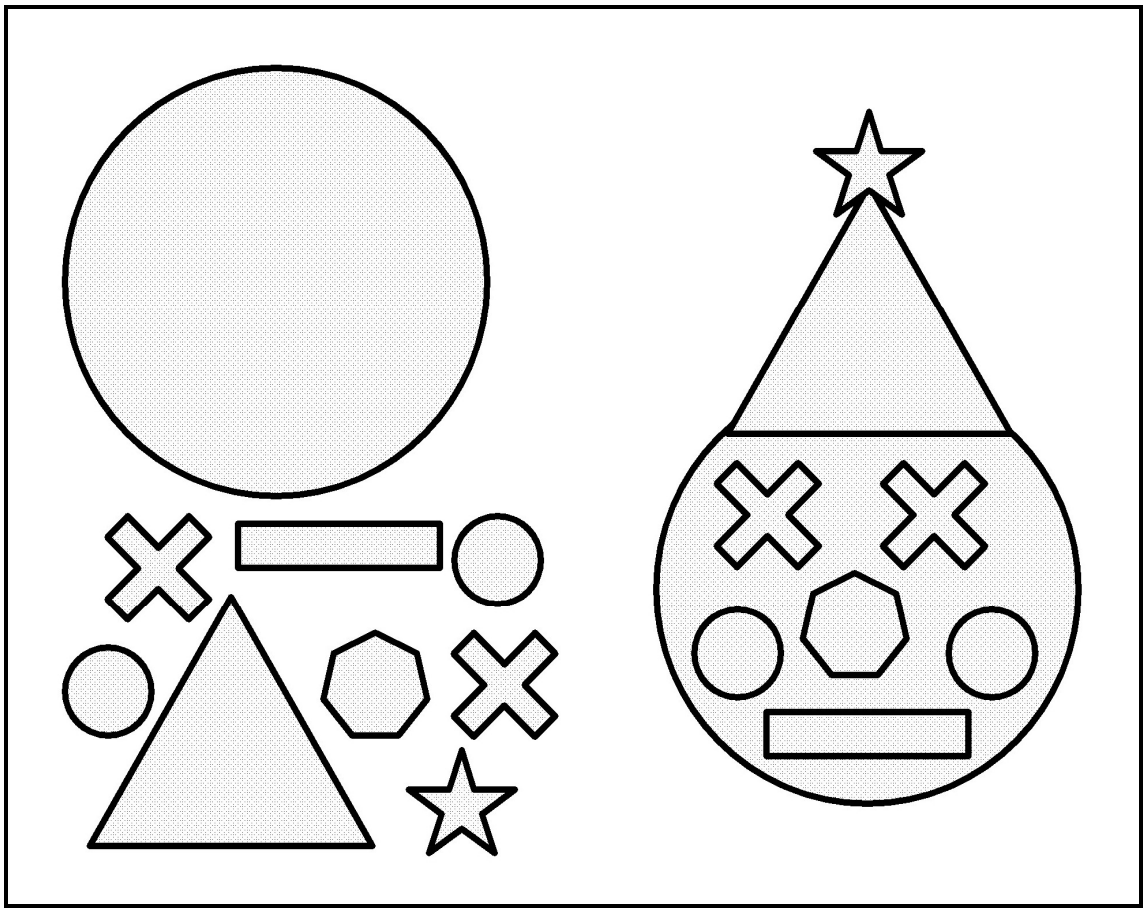

Figure 3.1 The perception of parts and wholes

3.2 Inattentional Blindness

Inattentional blindness (IB) provides another argument against the use of feature-based systems for similarity detection [28]. IB can be summed up as the phenomenon of "looking without seeing". When IB happens, even though an individual's eyes are wide open and various objects are imaged on their retinas, individuals seem to perceive only very little thing. At the beginning stage of visual perception, the brain spontaneously and automatically performs Gestalt grouping operations to integrate all retinal inputs for further processing. However, only stimuli which capture attention at the later stages of

processing are perceived. This has been experimentally demonstrated [28]; these experiments show that objects' features or stimuli such as location, motion, frequency and color were not perceived about 75% of the time in an inattentional condition. This is a well-known psychological phenomenon.

Consequently, when users try to identify similar web pages (that is, recognize pages that have been previously seen), they spontaneously ignore fine details and perceive only the whole image. If they are later asked to recall the details of the web page (such as the exact color of icons, the exact text, or other web page components), a likely answer could be "I don't know" or "I didn't notice that" [28]. That is, the entire web page is likely to leave a single impression on their memory. Empirical research into phishing scams supports this notion; users commonly do not observe the address bar, status bar, or security indicators displayed by modern web browsers, and can be fooled up to 90% of the time by high-quality Phishing sites [29].

3.3 Supersignals

Related to Gestalt psychology is the concept of "supersignals" [2], which seeks to explain how humans make rapid decisions when bombarded with a massive set of inputs. In this scenario, visual inputs tend to dominate, but other senses can play a role. We view the idea of supersignals as extending the basic Gestalt visual processing into a decision making mechanism. In our situation, the Gestalt process transforms the visual representation of a web page, producing a supersignal which acts as the input to the decision making process (is this page similar, or the same, as a page which was

previously viewed?). The content of these supersignals change with a number of factors – such as the individual's perceived familiarity with the situation. Supersignals can be thought of as trying to provide an explanation of an individual's behavior when they encounter a complex, but familiar, situation. The person can reduce the complexity of the situation by generating a supersignal which collapses a number of features into one impression, based upon their previous experience. Differences between the novice and the experienced driver are a good illustration of this idea. A novice needs to direct attention to many variables and traffic situations at once. Driving is a highly complex business for them. Any input variables can cause unexpected circumstances which need more processing time for the novice to handle. Complexity causes trouble, stress, and anxiety which make the situation more complicated to deal with. On the contrary, an experienced driver doesn't notice this complex situation as requiring the processing of so many independent variables. They are able to generate many supersignals (that is, they are able to integrate large numbers of inputs into a small number of holistic signals) to reduce the complexity. Once the complicated situation has been simplified, the driver can recognize similar situations from their prior experience, and apply an appropriate response from that experience. This recognition is also not an "optimized" comparison; decision theory tells us that people will select the first familiar situation that comes to mind and appears to fit the current circumstances; this will often take just a fraction of a second. People do not usually spend time reflecting on how closely the current situation matches the prior experience; it is simply "good enough" [2].

Similarity identification for humans is a decision making process related to many complicated factors. To complete the analogy, an individual who is new to the Internet acts as an inexperienced driver. However, the vast majority of Internet users can be consider as "highly experienced drivers" and hence their similarity decision process only accepts a very small number (potentially only 1) of input(s), or supersignal(s), into their decision making process. In order to design our similarity measure to be congruent to human perception, we will generate a "supersignal" that represents the whole of a web page.

We view the construction of supersignals as a sampling or compression process. Cognitively, an individual is attempting to wade through a massive number of inputs and reduce it to a minimal, but sufficient, single representation to allow a decision to be successfully undertaken. This sampling approach is undoubtedly highly non-linear, and is characterized by emphasizing and integrating seemly important aspects of the input; while ignoring or discarding the seemly unimportant aspects of the input. Hence, the process is unlikely to correspond to sampling in a traditional mathematical sense. In essence, an individual is seeking to reduce the input to a single irreducible form. Hence, we view the process as having parallels to approaches for defining Algorithmic Information theory [30] ; and hence, we seek to objectively measure the relationship (the basis of the decision) between supersignals within this framework. While, algorithmic information theory principally studies "complexity measures" on strings; clearly translating supersignals, or more accurately our representation of supersignals, into an appropriate form is not so straightforward. In addition, we seek to use these complexity

measures as the basis of objectifying our supersignals representation. Within Algorithmic Information Theory, Kolmogorov complexity [31] can be used to provide a theoretical definition of an objective evaluation of a pseudo-irreducible form of a signal; and the numerical difference between two such objective approximations can be considered proportional to the actual difference between two arbitrary signals.

3.4 Similarity Metric

While it can be stated that Kolmogorov Complexity is objective, this is clearly a theoretical position, as Kolmogorov Complexity is incomputable[1] in anything apart from contrived situations. However, [32] recently demonstrated that Kolmogorov Complexity can be successfully approximated by current compression techniques. Kolmogorov Complexity can be viewed as the ultimate compressor – producing for any arbitrary string (or file or image), a minimum description of that string, given some form of description language. Hence, practical compression approaches that compress arbitrary strings or files or images can be viewed as approximations to the optimal, however unattainable, compressor. Kolmogorov complexity can be viewed as the limiting case for compression technology. Specifically, [33] introduce the Normalized Information Distance (NID), which approximates Kolmogorov complexity within known limits. Further, they prove that NID is a valid metric within these limits. They claim that NID can "discover all similarities between two arbitrary entities; and represents object similarity according to the dominating shared features between two objects."

[1] In a Turing sense.

NID can be defined as follows: Let K($x|y$) refer to the Kolmogorov complexity, that is the length of the shortest binary program, which accepts as input y and outputs x; and let K(x) refers to the Kolmogorov complexity of x, that is the length of the shortest binary program with no inputs that outputs x. The value max (K($y|x$), K($x|y$)) can be considered as the length of the shortest binary program (with the reference universal prefix Turing machine) that with input y, computes x, and with input x, computes y. Given these definitions, NID can be defined as

$$NID(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{3.1}$$

Further details about NID and its properties can be found in [33]. However, the fact that Kolmogorov complexity is incomputable implies again that NID can not be used directly. Hence, [33]; [32] provide an approximation to this metric based upon real – world compression algorithms (denoted *C*) rather than the Kolmogorov complexity.

3.4.1 Normalized Compression Distance

Normalized Compression Distance (NCD) is described as a parameter-free distance metric which is believed to be able to uncover all similarities with a single metric [32, 33]. It is a practical metric approximating NID. It is computed from the lengths of compressed

data files, images, strings, etc. using real-world compression algorithms. For an arbitrary compression algorithm *C*, NCD is given by:

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \tag{3.2}$$

Clearly, the relationship between the denominators of NID and NCD is straightforward; however, the relationship between the numerators is less so. It is shown that the numerator of NID can be rewritten as

$$\max\{K(x,y) - K(x), K(x,y) - K(y)\} \tag{3.3}$$

Note that

$$K(x,y) = K(xy) = K(yx) \tag{3.4}$$

where *xy* and *yx* denote the concatenation of two signals. [32] argues that this numerator can be effectively approximated by

$$\min\{C(xy), C(yx)\} - \min\{C(x), C(y)\} \tag{3.5}$$

If we now assume that the symmetry property holds for the compression algorithm *C*, we have [32]:

$$\min\{C(xy), C(yx)\} = C(xy) \tag{3.6}$$

Clearly, it is important to understand that NCD is an approximation of NID; and that the symmetry property may not hold for all real-world compression algorithms. In addition, practical compression algorithms may invalidate common properties found in theoretical measurement systems, e.g. is monotonicity ($C(xy) \geq C(x)$) a guaranteed property of all block-coding compressors? Hence, this final approximation will require empirical verification within our context. (Note that much of the literature on applying the NCD uncritically treats it as a "universal similarity metric," e.g. [34-36]. We find such a sweeping assertion to be dubious at best when modeling a phenomenon as complex as human perception.)

In our context, the NCD value is a nonnegative number representing the distance/difference between two images (approximations of supersignals) which in turn represent a web page. Using the usual interpretation of similarity as an inverse of distance, we assert that the more similar two objects (images or web pages) are, the smaller the NCD distance between them should be.

3.4.2 Compression Algorithms and Supersignals

The gzip, bzip2, and PPM are popular data compression methods. Gzip is a Lempel-Ziv type compressor with a 32-kilobyte window [37]. Its reliability, speed, and simplicity make it become one of the most popular compressors. Bzip2 is a fast compressor which uses the Blocksorting algorithm [38]. It provides good compression and an expanded window of 900 kilobytes which has the ability to detect longer-range patterns. PPM (Prediction by Partial Matching) [39] is a compressor using a mix of statistical models arranged by trees, suffix trees or suffix arrays. It provides better performance but with the

side effect of slower speed and heavy memory consumption. Other compression algorithms such as LZMA provides better compression ability with the demand of smaller file size and faster processing time.

Different data compression algorithms lead to different varieties of NCD. Some data compression programs use many complex schemes that involve stochastic modeling of the data at many levels simultaneously. While the NCD metric is in theory application neutral, in practice we believe the choice of the compressor needs to be tailored to the application domain. For example, the "Blocksort" virtual compressor (which also uses the Blocksorting algorithm) is appropriate for frequency analysis, spectral analysis, and substring matching combined. Clearly no unique or optimal presentation exists for the construction of our pseudo- supersignals – the input to the compression stage. This topic needs further research to find the most appropriate mechanism for this encoding; and a variety of empirical evaluations to confirm any such supposition.

Chapter 4 Existing Anti-Phishing Mechanisms

4.1 Phishing Problem

According to a report released by Gartner [6], Phishing scams cost $3.2 billion in losses, victimizing 3.6 million people in the U.S. alone in 2007. And in 2008 the number of Phishing attacks on U.S. Consumers increased by 40 percent [40]. From July to December in 2009, the detected Phishing attacks from APWG [7] have increased to 126,697 incidents which is twice than the Phishing attack count for the first half year in that year [8]. In 2010, APWG [7] has found a professional Phishing crime organization called "Avalanche" was responsible for two-thirds of all phishing attacks in the second half of 2009 [8]. This indicates the Phishing scam has become more sophisticated and complicated than ever before. It is not a simple game of script kiddies [41] anymore. It has become an organized crime business.

Phishing is a type of online identity theft in which sensitive information is obtained by misleading people to access a malicious web page. While there are many different types of Phishing attacks, the most common Phishing scam seen today is a deceptive attack. Researchers usually conceptualize Phishing scams as a seven-step process [42] as shown in Figure 4.1. The scam begins when a malicious payload is sent to a user (often an email asking the user to login to one of their accounts). The second stage is when the user attempts to navigate to the login page for that account – but is instead directed to an attack page. The third and fourth stages are when the user is prompted for their account's credentials – and provides them. In stage five, these credentials are transmitted to the

Phisher, who uses them to impersonate the user in stage six. Finally, in stage seven, the Phisher is able to steal the assets held in the account. Phishing attack sites are designed to seem visually very similar to the legitimate site they are impersonating. As we have argued, user recognition of a website (or what they believe is the website) takes place in the blink of an eye [29]. Users do not consciously reflect when they make this decision. We suggest that Anti-Phishing methods need to be developed to explicitly account for the user's perceptions and actions.

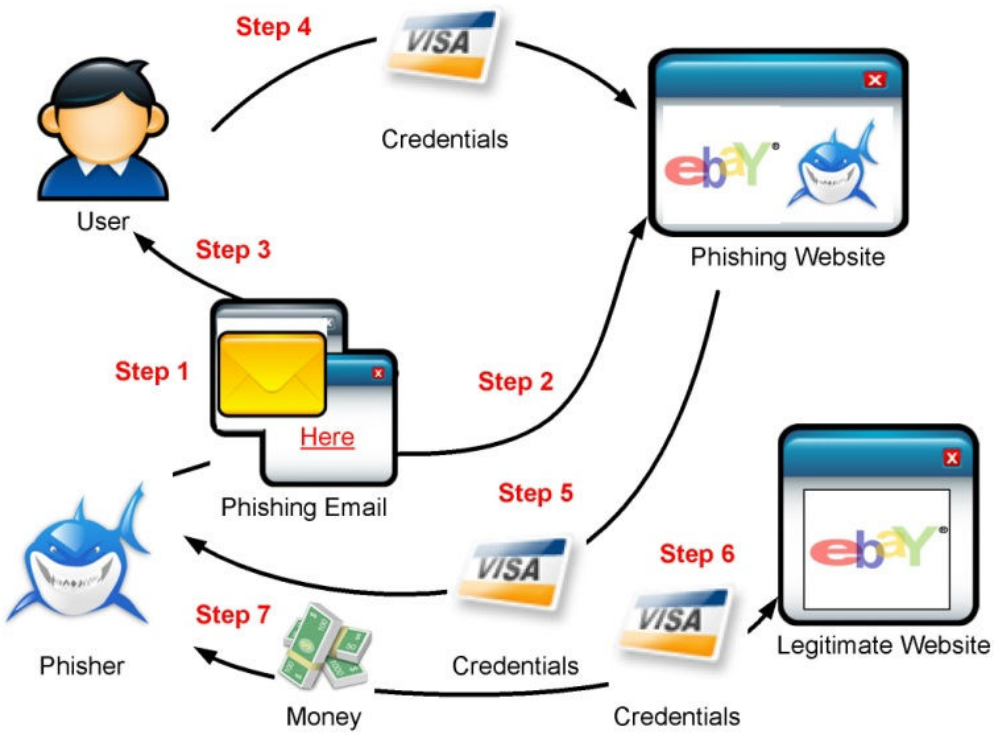

Figure 4.1 The Phishing seven-step process [42]

4.2 Existing Anti-Phishing Solutions

Email-filtering approaches are server-side techniques that try to interfere with stage 1 of the Phishing scam. Email is a very common vector for delivering a malicious payload to a user – in this case, a message involving a "call to action" regarding one of the user's accounts, and a "helpful" link to the website. By preventing users from receiving these emails, these approaches attempt to completely deflect the Phishing attack. They are closely related to Anti-Spam solutions, in that both rely on an analysis of the content of an email message. Therefore, they also suffer from the same weaknesses as Anti-Spam approaches. In particular, the image-based technique demonstrated in Figure 2.1 and Figure 2.2 is a potent countermeasure against content analysis. Some well-known email filtering solutions include Thunderbird [43] and PILFER [44].

Anti-Phishing tool bars are the most popular and widely-deployed solutions to fight Phishing websites. Most of the toolbars employ blacklists and whitelists. They determine the URL currently being viewed, and send it to the blacklist/whitelist (B/W) database for filtering. The result will be delivered back to the user with either an alert warning the user of a possible Phishing scam, or an assurance that the site is legitimate. IE8 [45], FireFox3 [22], Spoofguard [46], and Netcraft [47] are popular toolbars in wide usage. The performance of the tool bar depends on the B/W database (except for SpoofGuard, which uses heuristics). Unfortunately, the average life time of a Phishing website is 3.4 days [9]. These short-lived websites can defeat these tool bars because the database is not updated fast enough to protect users from a brand-new Phishing site. It simply takes time to detect a new Phishing site. Another trick known as "DNS/URL redirection or domain

forwarding" [48] can fool the B/W databases by rapidly changing the DNS/URL IP address mapping in a dynamic DNS domain server. The "blocked" Phishing website can be back to business in only one minute. Although other heuristic analysis methods such as domain registration lifetime checking are proposed and deployed in the toolbars, the Phishers still can find a way to fool classifiers built on these heuristics. Human factors must also be acknowledged as another threat to the Anti-Phishing solutions. Some users input their credentials even when they receive warnings from the toolbars [49] .

Mutual authentication is another common Anti-Phishing solution. By acquiring secret messages or preauthorized signals from the server (legitimate website) with a secure connection (usually SSL connection), the client (the user) can make sure they are browsing the legitimate website and their credentials can be safely transmitted. The most challenging part of this client and server-side method is human factors. Firstly, users must choose to install the software on their system, and follow complicated instructions in setting it up. Research indicates that the majority of computer users will never change the default configuration of their software [50]. Secondly, once users have set the system up, false positives will reduce the alertness of users. Such false alarms are known to destroy user trust in any Anti-Phishing system [29]. Thirdly, Phishers still have social engineering tricks that can fool users into disabling the Anti-Phishing systems. For example, a letter entitled "Incompatibility notice for your Anti-Phishing system," if apparently sent from a trusted authority, can trick users into removing their Anti-Phishing system. Some well-known mutual authentication solutions include DSS[51], Pass Mark

(now part of the RSA Identity Protection & Verification Suite) [52] , and Yahoo Mail's sign-in seals [53].

In [54], they develop the BogusBiter, a client side based anti-Phishing tool which feeds a relatively large number of bogus credentials to the Phishing website. They hope to cover the real user credentials by flooding the Phisher many fake ones generated by their system. The main limitation for this mechanism is it can only provide a second layer of defense from the Phishing attack. Their tool only activates after the browser based anti-Phishing tool bar has identified the website as a suspected Phish. BogusBiter then attempts to increase the difficulties for the Phisher to use the acquired credentials. The second limitation (also mentioned in their work) is that by using some filtering techniques such as meaningful-word and statistical filtering, the Phisher is still able to pick out the valid credentials from a large number of invalid credentials. They also mentioned the nonstandard login web page can also be used to increase the difficulties to deploy this mechanism to the real world scenarios.

We believe that any effective Anti-Phishing solution must be robust against the counterattacks of determined, inventive adversaries. This means that it is not enough to find features that discriminate Phishing web pages from legitimate pages; those features must also be extremely difficult or impossible for the Phisher to alter. Our approach is to examine the Phishing scam and find a critical item in the scam that cannot be changed without severely weakening the effectiveness of the scam. By finding such critical

characteristics of Phishing websites, we can design a classifier that is robust against the Phishers efforts to defeat it.

4.3 A Key characteristic of Phishing websites

Phishing websites usually look similar to a legitimate website, but not exactly the same. Designing a visually similar webpage is a crucial step in the Phishing scam. On arrival at a website (the point between stages three and four of the scam), a user faces a choice: they must either choose to believe that the site is legitimate, or that it is a fake. This choice is not made after a period of reflection; instead, the user looks for the "supersignal" of a recognized, trusted website. This decision is clearly for the Phisher; if the user is suspicious at this point, they will probably not provide their account credentials, and the scam fails. Thus, the Phisher must craft a page that closely imitates the legitimate page, causing the user to erroneously recognize the supersignal of the legitimate page. Once this decision is made, psychological studies [2] tell us that it is unlikely to be revisited until a significant amount of contrary evidence is observed. Therefore, a visually similar webpage becomes an inevitable element in the Phishing scam – and thus a characteristic that can be considered as a fundamental component of a Phishing attack. Although a Phisher's goal is to make the Phishing website as similar to the legitimate website as possible, there are still differences between them. This is mainly due to the frequent updates of the legitimate website. Pictures, advertisements, and new information are renewed by the legitimate webmaster from time to time to keep the site fresh and interesting to users. Phishers, however, do not expend the effort on such renovations, leading to a difference. As long as the users believe they are browsing the

legitimate website, this small difference is irrelevant to the Phisher. Thus, by detecting visual similarity between an unknown page and a known legitimate page, we can recognize an attempt to trick the user. Because this attempted deception is critical for the Phisher, we believe that Phishers will not be able to alter this attribute of their sites to avoid detection. This attribute seems robust against anything short of a wholesale revamping of the current Phishing model.

A key point to note is that using Phishing web pages to test a similarity algorithm immediately gives us an excellent "ground truth" for our evaluations. The authors of Phishing toolkits (who are now usually professional Internet criminals) go to great lengths to craft fraudulent pages that will fool human beings. This means that we can assume that captured Phishing pages will be perceived as highly similar to the legitimate brand they imitate; to the point that human beings will confuse them even with significant financial consequences at stake. Furthermore, monitoring sites such as the Phish Tank provides us with a corpus of Phishing web pages that essentially covers the entire domain of interest (current Phishing scams); the entire purpose of the Phish Tank is, after all, to provide "accurate and actionable" information to the anti-phishing community [55]. In contrast, corpora for Web clustering or search cannot possibly both cover the entire domain of "clustering webpages" or "finding relevant webpages" and provide a ground truth for that corpus. For instance, [11] created a corpus of 1.6 billion web pages by using the Google web crawler, in order to compare two existing similarity algorithms. However, as they freely acknowledge, it was impossible to create a ground truth for this dataset (i.e. what pairs of web pages would in fact be perceived as highly similar). In [16], they

attempt to use human-created web directories (e.g. Yahoo! or the Open Directory Project) to create a ground truth, based on the relative position of two documents in the directory tree (the "familial distance"). The key assumption is that document similarity is monotonically related to this familial distance; however, as the authors acknowledge, this is not always so. Importantly, no evaluation or estimate of how frequently monotonicity fails is provided; thus, this evaluation metric cannot truly be said to be a ground truth. Our approach, by contrast, is a ground truth based on human perceptions. This does not mean (and we do not claim) that the NCD technique by itself is a complete anti-phishing solution.

Our choice of the NCD technique is intended to overcome the primary weakness of feature-based similarity comparisons: the ability of a Phisher to easily craft a web page that seems visually similar to the legitimate page, but is not detected as such. The problem has some parallels to preventing message forgery in cryptographic systems; the mapping from the original to the encrypted message should be extremely difficult to reverse-engineer. The central characteristics for a successful encryption are confusion (a complex, non-monotonic mapping from plaintext to cipher text characters) and diffusion, which is the scattering of information across a message. However, Phishing scams cannot be dealt with using cryptographic techniques, because the human user accepts many similar "messages" as being identical – whereas cryptographic techniques such as AES map a one-bit difference in the plaintext to a change in 50% of the bits in cipher text. Such approaches are not congruent to human perceptions. We do believe, however, that the characteristic of diffusion is useful in preventing the Phisher from reverse-

engineering our similarity technique and finding an economical countermeasure. NCD, being based on compression techniques, naturally makes use of information (in Shannon's sense) that is diffused throughout the image of a web page. Thus, we feel our NCD technique is likely to be highly resistant to obfuscation countermeasures.

As shown in Figure 4.2, Figure 4.3 and Figure 4.4, there are slight differences between the legitimate and Phishing websites targeting them. These real world web pages were collected from the Phish Tank [55] through 20/05/08 to 22/05/08. We can clearly observe that the text, pictures, links, and web structure in the Phishing web pages are not identical to the legitimate one. In the next section, we will report on experiments that demonstrate the utility of our similarity-based approach to detecting Phishing scams. The Phishing web pages used in these experiments are actual Phishing pages drawn from the Phish Tank [55], demonstrating that the method works in current real-world scenarios.

Figure 4.2 The Legitimate BOA web page

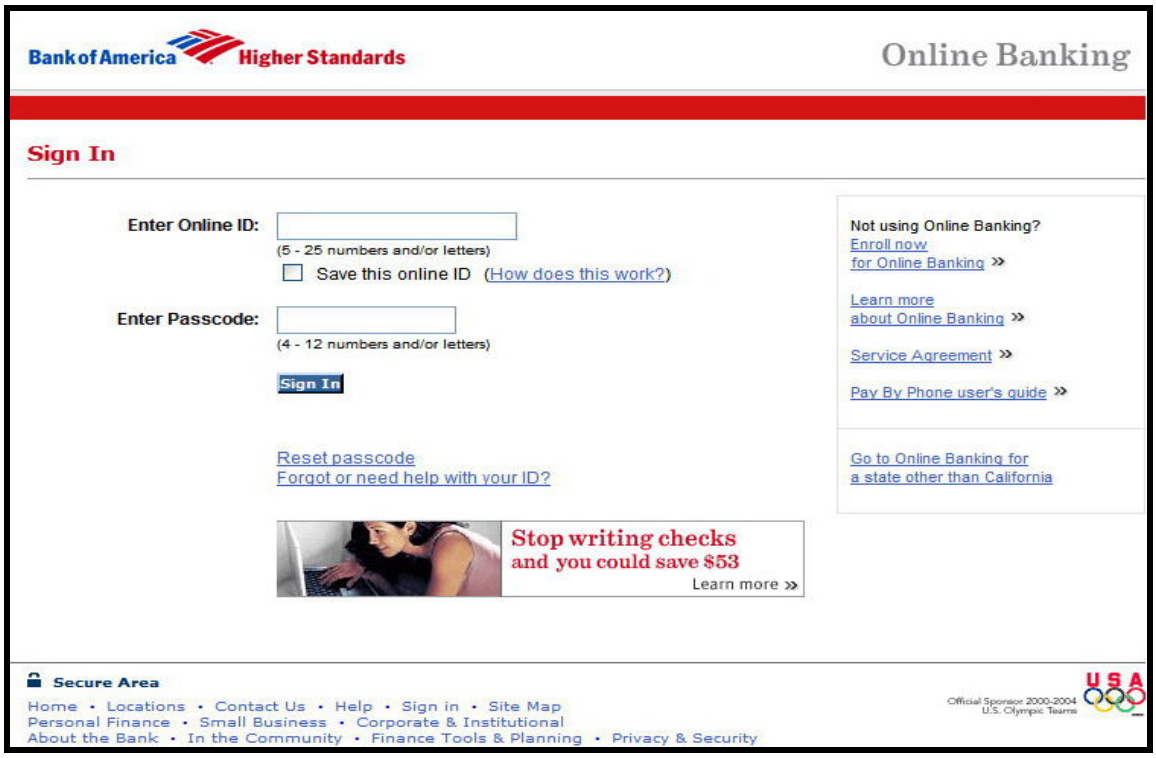

Figure 4.3 The Phishing BOA web page

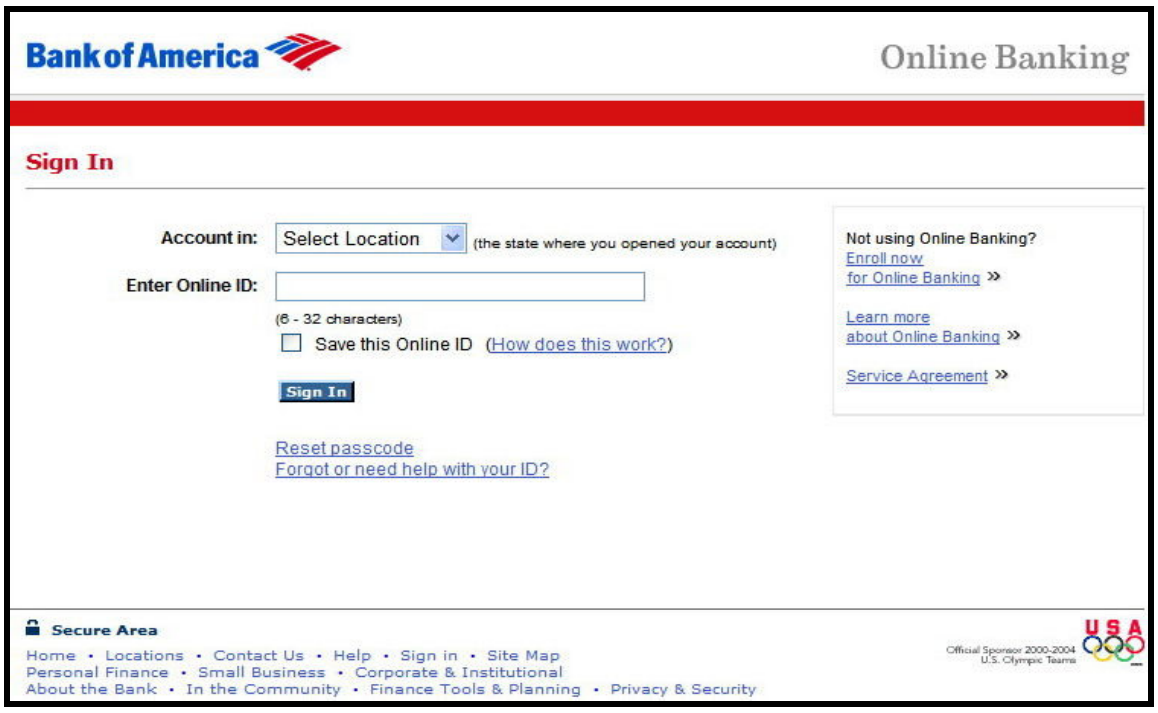

Figure 4.4 The Phishing BOA web page

4.4 Goals, research and hypothesis

We have discussed our webpage similarity approach, and the domain we are going to apply this approach to (anti-Phishing). Now we are going to explain our research work plan. Our long-term goals, research and hypothesis can be stated as follows: Can repeated research efforts and results into improving the representation of a supersignal (both the initial representation and the compression component) continually provide mechanisms which will defeat Phishers' attempts to construct effective Phishing web sites? We view these Phishing web sites as a "moving target"; as researchers produce mechanisms to prevent successful phishing attacks, Phishers will in turn produce new mechanisms to circumvent these defensive measures.

Our short-term goal, research and hypothesis (and the principal application in this thesis) can be stated as: Can an initial representation of a supersignal (both the initial representation and the compression component) defeat Phishers' current attempts to construct effective Phishing web sites? We will explore this hypothesis in the remainder of the thesis by empirically evaluating our initial representation against current Phishing sites found "in the wild." We have considered several possible representations of a supersignal (the DOM tree; the HTML code; the link structure of a page). We have chosen the rendered web page as the initial representation because that – and not, for instance, the DOM tree or HTML code – is what the user perceives at a web site. We therefore render the webpage and capture a high-resolution image representation thereof. And hence, subsequently two such images are the input to the compressor stage which produces the NCD value for the decision making process.

Chapter 5 Our Empirical Proof-of-Concept Evaluation

We conducted four experiments to validate our method for measuring similarity between web pages. Due to the fact that all the samples are from the real world, this objective is combined with an assessment of our proposed similarity-based system for recognizing Phishing web pages. The first experiment, "12 Pairs" was designed to test if pairs of highly similar websites could be matched together while being distinguished from less-similar pages. The second experiment, "Clustering" was designed to test whether we could find a cluster of similar pages within a larger group of dissimilar pages (i.e. to see if any successes from the "12 Pairs" experiment generalize to uneven distributions of similar and dissimilar web pages). Thirdly, we perform a large-scale experiment specifically to test the ability of our similarity measure to distinguish between legitimate and Phishing web pages. This last experiment involved a control group of 120 comparisons between legitimate websites (legitimate versus legitimate), and 320 Phishing web pages targeting those sites (Phishing site versus the legitimate they imitate). We analyze these results using both statistical tests for a difference in means, and using the ROC curve for a putative decision-threshold classifier. Finally, we conduct a small scale test of possible obfuscations (based on image-processing techniques) to examine the robustness of our similarity technique.

In using the NCD, one key decision is which of the many compressors in the literature will be employed. There is very little guidance in the literature on this point; the most significant result available is that NCD is skewed by the size of the objects to be

compared due to internal buffer limits in certain compression algorithms [56]. Moreover, our problem domain consists of rendered web pages. Converting a fundamentally 2-dimensional object (a rendered web page) into a one-dimensional string (i.e. treating an image as a row-major array) seems likely to destroy the spatial relationships in the image. However, among the very limited work in applying the NCD to images, there is virtually no exploration of two-dimensional compression techniques (e.g. the wavelet-based approach in JPEG images). [57] only used black and white images; [58] and [59] explored NCD for grayscale images, and [60] used grayscale textures, all employing string-based compressors. In [61], they created a new distance based on parsing the dictionary of a (one-dimensional) Lempel-Ziv-type compressor and applied it to color images. The only usage of a two-dimensional compressor in the NCD we could find was an image co-registration algorithm that compared JPEG and bzip2 [62], and this also examined sets of monochrome images (the red, green and blue channels were separated).

The paucity of guidance on applying NCD to images means that we need to select our compressors by considering how browsers render web pages from first principles. The web page's source code (written in one or more markup and/or scripting languages) must be transformed into a visual representation by a fault-tolerant browser; indeed, browsers will render web pages even if they contain substantial errors [63], following the display preferences set by individual users. This means that even the same webpage might not be rendered to the same dimensions for two different users, and might be rendered quite differently by different browsers; this includes the size of the rendered image! Two-dimensional compressors require a rectangular image and thus "concatenating" two

images of differing sizes (an essential step in the NCD) makes no logical sense. (Note that we cannot just concatenate files; two-dimensional compression is only appropriate for image data, not the headers of a JPEG file!) A one-dimensional compressor, on the other hand, imposes no such requirement. Furthermore, the co-registration results of [62] for the bzip2 compressor indicate that one-dimensional compressors can be relatively robust against spatial shifts (the relative variation of the NCD values is quite small under translation). These reasons provide a sound rationale for the use of one-dimensional compressors with a row-major image representation in our experiments; while this means we are following existing practice in applying the NCD to images, previous work has not developed a sound foundation for these choices. We are also applying the NCD to RGB color images rather than monochrome images.

5.1 The Twelve-pairs Experiment

The objective of this experiment is to see if we can group twelve legitimate web pages and twelve Phishing pages each targeting one of these pages together in pairs. Based on the argument that a legitimate page and a Phishing page targeting it are highly similar to one another, this experiment also tests the validity of our proposed similarity metric. The expected result for this test is that each legitimate page and its single Phish will be paired together as the most similar to one another, for all twelve pairs.

5.1.3 Design and methodology

We collected 12 different legitimate web pages and 12 Phishing web pages targeting them as the samples in this experiment (Table 5.1).We chose financial websites in Table 5.1 based on the frequency with which Phishing sites attempt to imitate them. The Phish

were captured in the PhishTank [64]. In addition, one Italian and one Spanish website are added to the group to check if there was a language or regional dependency in our similarity metric. There are a total of 24 samples in this test, each of which is compared against all 23 other samples. We use blocksorting [38] as the compressor for this experiment. Note that, although NCD is theoretically a distance metric (and therefore commutative), in practice this would require a "perfect" compression algorithm, which does not exist. Thus, the NCD values we observe are not commutative. Therefore, all the NCD values shown below are the average value of both orderings of every two websites. Lower NCD values indicate greater similarity (i.e. we consider distance the inverse of similarity).

Table 5.1 Samples list for 12 Pairs test

Name of the website	Collection date (yy/mm/dd)
1. ArkValley	08/02/26
2. BancadiRoma(it)	08/02/26
3. Chase	08/02/26
4. CitiBank	08/02/26
5. FifthThird	08/02/26
6. ibercajadirecto(es)	08/02/26
7. LloydsTSB	08/02/26
8. RBC	08/02/26
9. USBank	08/02/26
10. Wachovia	08/02/27
11. WaMu	08/02/27
12. NatWest	08/02/28

5.1.4 Interpretation of results

Consider the Royal Bank of Canada (RBC) website as an example. The NCD values shown in Table 5.2 are the NCD of the remaining 23 samples against the legitimate RBC website (RBC-L). The "-L" in this table refers to the legitimate website of that brand,

while “-P” denotes a Phishing webpage targeting that brand. Most of the NCD values are between 1.01~1.07. The values in Row 15 and 16 are different. An NCD of ~0 in Row 15 (RBC-L against RBC-L) indicates that the algorithm properly finds that there is perfect similarity between a webpage and itself. NCD value 0.632, found in Row 16 (RBC-L against RBC-P) is far less than the values against the other 22 web pages. According to this result, we can say RBC-L is most similar to RBC-P in this group of web pages.

Table 5.2 The NCD values of RBC-L against other 23 websites

Name of the website	NCD value
1.ArkValley-L	1.059
2.ArkValley-P	1.047
3.BancadiRoma(it)-L	1.029
4.BancadiRoma(it)-P	1.031
5.Chase-L	1.041
6.Chase-P	1.055
7.CitiBank-L	1.039
8.CitiBank-P	1.052
9.FifthThird-L	1.037
10.FifthThird-P	1.051
11.ibercajadirecto(es)-L	1.025
12.ibercajadirecto(es)-P	1.025
13.LloydsTSB-L	1.059
14.LloydsTSB-P	1.057
15.RBC-L	0.168
16.RBC-P	0.632
17.USBank-L	1.024
18.USBank-P	1.040
19.Wachovia-L	1.073
20.Wachovia-P	1.073
21.WaMu-L	1.048
22.WaMu-P	1.050
23.NatWest-L	1.013
24.NatWest-P	1.013

The same pattern holds true for all other websites; the most similar web pages are always a Phishing site against the legitimate site it targets. For example, the lowest NCD value for ArkValley-L is against ArkValley-P, and vice versa. As shown in Table 5.3, all twelve legitimate websites are paired against the Phishing sites targeting them. Again, the values reported for NCD are the average of both computations of NCD between two websites. To visualize these results, we have employed the quartet trees [65] as shown in

Figure 5.1. We selected this technique because the quartet-puzzling algorithm is based on identifying locally optimal pairings of elements (in the maximum-likelihood sense), which is a good match to the data we wish to visualize. We interpret Figure **5.1** as indicating that the twelve pairs have been successfully grouped together, as the two members of a pair always share the same parent node (i.e. the branch length between them is minimal).

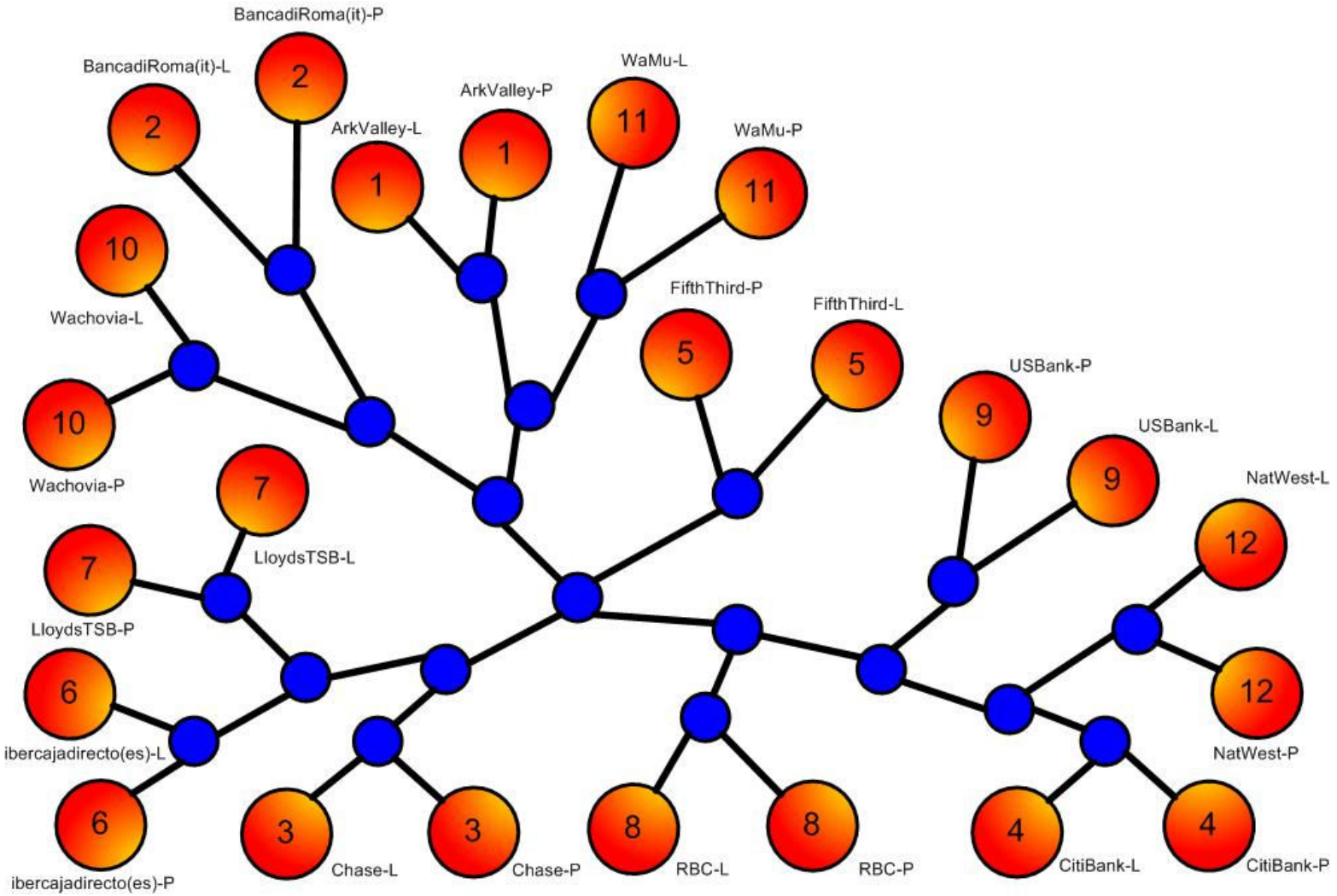

Figure 5.1 Quartet tree visualization for 12 pairs experiment

Table 5.3 The NCD Values for all 12 Pairs

Name of the website	In Pairs	NCD values
1. ArkValley-L against ArkValley-P	YES	0.558
2. BancadiRoma(it)-L against BancadiRoma(it)-P	YES	0.352
3. Chase-L against Chase-P	YES	0.748
4. CitiBank-L against CitiBank-P	YES	0.808
5. FifthThird-L against FifthThird-P	YES	0.890
6. ibercajadirecto(es)-L against ibercajadirecto(es)-P	YES	0.221
7. LloydsTSB-L against LloydsTSB-P	YES	0.284
8. RBC-L against RBC-P	YES	0.632
9. USBank-L against USBank-P	YES	0.834
10. Wachovia-L against Wachovia-P	YES	0.149
11. WaMu-L against WaMu-P	YES	0.218
12. NatWest-L against NatWest -P	YES	0.329

5.2 The Clustering Experiment

The objective for this experiment is to determine if the NCD similarity technique can detect a single "cluster" of highly similar web pages within a larger group of web pages. This experiment examines the performance of the NCD similarity technique when the groups of highly similar websites are not balanced in size. This is known as the imbalanced-dataset problem in machine learning (alternatively, the sample selection bias problem in statistical modeling), and can profoundly affect the performance of a model. The expected result in this experiment is that all of the highly similar web pages will have a lower NCD value against one another than against the dissimilar pages, and that pages outside of the "cluster" will have higher NCD values against one another.

Table 5.4 Sample list for the Clustering test

Name of the website	Legitimate/Phishing	Collection date (yy/mm/dd)
01-BOA-P	Phishing	08/05/09
02-BOA-P	Phishing	08/05/09
03-BOA-P	Phishing	08/05/09
04-BOA-P	Phishing	08/05/11
05-BOA-P	Phishing	08/05/13
06-BOA-L	Legitimate	08/05/09
07-Wachovia-L	Legitimate	08/05/11
08-LloydsTSB-L	Legitimate	08/05/09
09-AbbeyNational-L	Legitimate	08/05/10
10-NatWest-L	Legitimate	08/05/10

5.2.1 Design and methodology

We selected BOA (Bank of America) as the legitimate website and collected five Phishing websites against it from the Phish Tank[55] from 08/05/09 to 08/05/13. Then four other legitimate, and highly popular, financial websites were collected from 08/05/09 to 08/05/11. We selected financial websites because this should make the test more rigorous; conceptually, two financial websites should be at least somewhat more similar to each other than, say, an online auction site is to a bank website. These 10 samples are shown in

Table 5.4.The block sorting compression algorithm is used for the NCD calculation in the experiment.

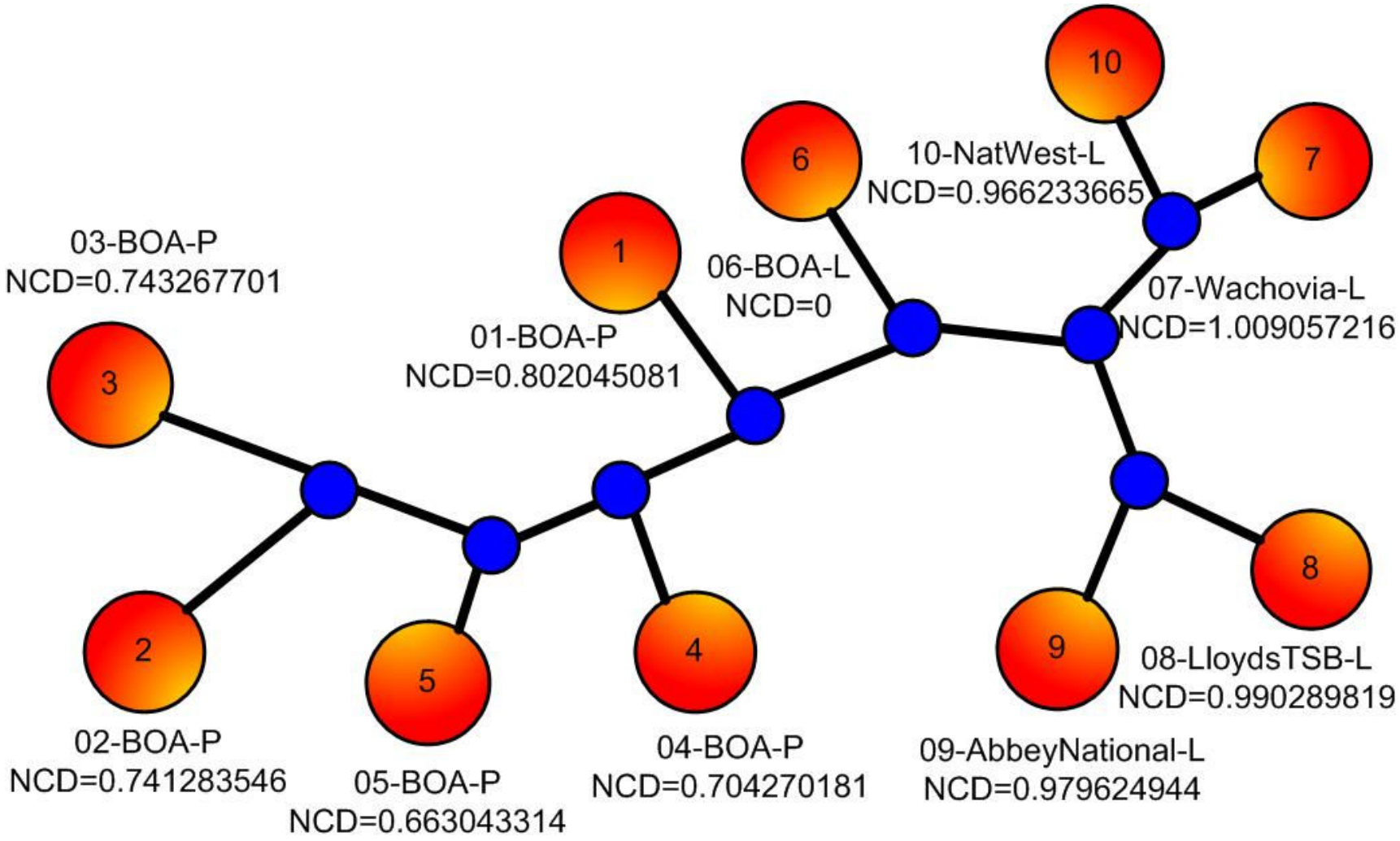

Figure 5.2 Quartet tree visualization for clustering experiment

Table 5.5 NCD Values Against BOA-L

Website	NCD value
01-BOA-P	0.802
02-BOA-P	0.741
03-BOA-P	0.743
04-BOA-P	0.704
05-BOA-P	0.663
06-BOA-L	0.183
07-Wachovia-L	1.009
08-LloydsTSB-L	0.990
09-AbbeyNational-L	0.979
10-NatWest-L	0.966

5.2.2 Interpretation of results

In Table 5.5, we present the NCD of each webpage against the legitimate Bank of America webpage. Rows 1 to 5 correspond to Phishing pages that target Bank of America, while rows 7 to 10 correspond to different, legitimate financial websites. There appears to be a considerable difference between the Phishing web pages targeting BOA and the other four financial websites. Therefore, we can say that the legitimate BOA website is more similar to the five Phishing websites targeting it than the other four websites in our experiment, and these six web pages should be clustered together. A quartet tree visualization of the NCD values is presented in Figure 5.2.

5.3 The Large Scale Experiment

The objective of this experiment is to test our proposed similarity-based Anti-Phishing technique on a reasonably large corpus of legitimate and Phishing websites. We collected 16 legitimate websites, and 20 Phishing web pages targeting each of these sites (total 320

Phishing web pages). Our initial experiments (12 Pairs and Clustering) have supported our assumption that a Phishing page will be highly similar to the legitimate website it targets. Furthermore, they also indicate that a legitimate page and the phish targeting it will be more similar to one another than to other legitimate websites. Thus, in this experiment, we will contrast two populations: one consists of the pair wise NCD between all of the legitimate sites, while the other consists of the NCD between a legitimate site and each of the Phishing web pages targeting it. The expected result of this experiment is that there should be a statistically significant difference in the means of the two populations, specifically with the mean of the latter group being lower.

5.3.1 Design

Our goal in this experiment is to examine how the NCD similarity technique would perform in a realistic, browser-level Anti-Phishing scenario. Assume that we have access to a whitelist of legitimate websites, which represent the likely targets of Phishing scams. This whitelist mainly contains the genuine information of our legitimate websites such as the up-to-date webpage captured images for NCD similarity classification and the domain name information (name and registration date) for further Phishing identification.
This is plausible because the number of brands that Phishers attack in a month is relatively constant (at less than 131 in Jan, 2008), with a small number of brands (less than 15 in Jan, 2008) accounting for over 80% of all Phishing attacks [9]. When we visit a website, we automatically execute an image capture, followed by a comparison (using the NCD similarity technique) against all websites in the whitelist. If there is a strong similarity to one of the white listed sites (i.e. the NCD is unusually low), we signal an

alert. This experiment is designed to determine whether or not a population of Phishing sites exhibits a statistically significant difference in the mean NCD against their target brand when compared to differences between different, legitimate websites. If such a difference exists, then the NCD similarity technique is viable in the Anti-Phishing scenario we have outlined.

Using the same criteria as in the previous experiments (frequency of phish targeting a legitimate site), we chose 16 legitimate websites (see Table 5.6). As discussed, the great majority of Phishing scams (more than 80%) active in any one month target as few as 15 websites. We thus chose 16 as a reasonable number of legitimate sites to "protect." These 16 sites were also the most heavily phished during our collection period, allowing us to reach our goal of capturing 20 "live" Phishing web pages for each legitimate website. "Live" phish are Phishing web pages that have not yet been taken down from their host servers; we captured these phish by visiting them as soon as we observed them in the PhishTank [55] or the Broadway PhishTracker [66] during the period May 10-July 10, 2008.

Table 5.6 Samples for the Large Scale Experiment

Group One	Group Two
01-ebay-L	01-ebay-P1 ~ ebay-P20
02-PayPal-Home-L	02-PayPal-Home-P1 ~ PayPal-Home-P20
03-PayPal-L	03-PayPal-P1 ~ PayPal-P20
04-Halifax-L	04-Halifax-P1 ~ Halifax-P20
05-NatWest-L	05-NatWest-P1 ~ NatWest-P20
06-BOA-L	06-BOA-P1 ~ BOA-P20
07-ebay(it)-L	07-ebay(it)-P1 ~ ebay(it)-P20
08-Wachovia-L	08-Wachovia-P1 ~ Wachovia-P20
09-LloydsTSB-L	09-LloydsTSB-P1 ~ LloydsTSB-P20
10-RBS-L	10-RBS-P1 ~ RBS-P20
11-AbbeyNational-L	11-AbbeyNational-P1~ AbbeyNational-P20
12-PosteItaliane(it)-L	12- PosteItaliane(it)-P1 ~ PosteItaliane(it)-P20
13-HSBC(uk)-L	13-HSBC(uk)-P1 ~ HSBC(uk)-P20
14-Cartasi-L	14-Cartasi-P1 ~ Cartasi-P20
15-WellsFargo-L	15-WellsFargo-P1 ~ WellsFargo-P20
16-eppicard-L	16-eppicard-P1 ~ eppicard-P20
Total samples count	
16	16*20=320

We designed two groups in this experiment (shown in Table 5.6). The samples for group one are the 16 legitimate websites. The samples for group two are the 320 Phishing websites targeting the legitimate websites in group one.

5.3.2 Methodology

Two populations of NCD values are generated in this experiment, using the blocksorting compressor. Firstly, we compute all possible pair wise NCD values for the websites in group one (note that the NCD is, at this time, only defined for the comparison of two objects). This population represents the expected NCD between two different, legitimate sites, and is the control group for this experiment. The computation results in a 16*16

matrix; the diagonal values should be removed (NCD(x,x)~=0), and the corresponding entries in the upper and lower triangle of the matrix are aggregated together, yielding (16*16-16)/2=120 elements in this population. We have used both the arithmetic mean and the maximum to aggregate each of the two corresponding NCD values; there is currently no guidance available on which would be the more effective choice. Our decision to use the pair wise differences between our legitimate sites seems likely to cause an over-estimate of the false positive rate in our experiments; the 16 brands all have highly similar objectives, will likely share at least some common text, and the structural layout of the pages should be more similar than would a white list site and a random site. Given the importance the anti-Phishing community places on minimizing false positives, this seems to be a reasonable approach. To form the second population, we computed the NCD between each Phishing page and the legitimate page it targets. As we have captured 20 Phishing pages for each legitimate page, this yields 320 elements in the group two populations (again, we compute the average and maximum of the two NCD values).

Our hypothesis is that the NCD values in group two are significantly less than group one. As we have more than 300 samples, we choose to employ the z-test for sample means to test this hypothesis. We repeat this experiment for both aggregating the NCD values by the arithmetic mean, and for aggregating by the maximum operation. Note that both the group one and group two populations are different in each of these repetitions, so the results are independent.

5.3.3 Interpretation of Results

The results of the z-tests are given in Table 5.7 (aggregation by average) and in Table 5.8 (aggregation by maximum). In both cases, we reject the null hypothesis with $p < 0.05$, and so we conclude that our original hypothesis – that NCD values in group two are significantly less than group one – is supported. Furthermore, it appears that the choice of using arithmetic mean or maximum to aggregate corresponding NCD values does not influence this outcome. Thus, the NCD similarity technique is a viable Anti-Phishing strategy.

Table 5.7 Results From the Z-Test for Average

Two Sample for Means		
	Legitimate vs. Legitimate	Legitimate vs. Phishing
Mean	1.005	0.745
Known Variance	0.001	0.087
Observations	120	320
z	15.577	
P(Z<=z) one-tail	< 0.001	
z Critical one-tail	1.645	

Table 5.8 Results From the Z-Test for Maximum

Two Sample for Means		
	Legitimate vs. Legitimate	Legitimate vs. Phishing
Mean	1.005	0.745
Known Variance	0.001	0.087
Observations	120	320
z	15.578	
P(Z<=z) one-tail	< 0.001	
z Critical one-tail	1.645	

5.4 Effectiveness as an Anti-Phishing Classifier

The viability of our proposed approach has been evaluated in three experiments. 24 samples (12 legitimate, 12 Phishing), 10 samples (5 Phishing, 1 matching legitimate, and 4 other legitimates), and 440 samples (120 pairings of legitimate sites, 320 Phishing) of real-world web pages are used in these three experiments, respectively. In our small-scale "12-pairs" and "clustering" experiments, the NCD technique was 100% accurate in grouping a legitimate page with Phish targeting that brand. In our large-scale test (which simulates a reasonable client-side anti-Phishing scenario), a *z*-test reveals that the NCD between a Phish and its target brand is significantly less than that between two different, legitimate sites.

The results presented in Table 5.7 and Table 5.8 show that our similarity metric is able to distinguish highly similar pages from dissimilar pages. However, since our technique seems effective in detecting (in a statistical sense) phishing pages, it is only sensible to inquire what the performance of this technique would be if it were used specifically as a classifier. For this discussion, we assume that we are implementing a simple decision threshold over the NCD values, with no other features.

Anti-phishing researchers often evaluate their algorithms using two interrelated metrics: the true positive rate (true positives divided by the sum of true positives and false negatives) and the false positive rate (false positives divided by the sum of false positives and true negatives). (Note that these measures are equivalent to sensitivity and (1-specificity) in the medical diagnostic testing literature.) However, the two measures are

not considered equally important; false positives are extremely annoying to the average user, and even a fairly low false positive rate may well lead to the abandonment of the system [29]; [67, 68]. In the machine learning literature, whenever such differential error costs are present, it is customary to analyze the performance of a classifier using the Receiver Operating Characteristic (ROC) curve [69]. The ROC curve is a plot of true positive rates against false positive rates, under a variety of "tradeoffs" between improving one metric or the other. The ROC curve allows one to visualize the capabilities of a classifier when the analyst is concerned with different costs (penalties) associated with false-negative errors versus false-positive errors, and to compare two different classifiers in the presence of differential error costs.

Our analysis in this section is a different presentation of the same results from Table 5.7 and Table 5.8. Instead of a *z*-test, we subject the NCD values to a simple threshold-based decision rule: if the NCD value is less than the threshold, we judge the page in question to be a phish targeting one of our protected pages; if it is greater, we deem the page legitimate. We vary the threshold across an adequate range to produce false positive rates from 0% to roughly 100%. The spacing of thresholds is non-uniform so as to reduce the importance of interpolations in the ROC curves. (The distribution of NCD values for legitimate sites has a much smaller variance than that for phishing pages against their targets, meaning there could be a large jump in FP rates for uniformly spaced thresholds.) We again compare aggregating NCD value pairs using the arithmetic mean or the maximum value. In addition, we also compare two different one-dimensional compression techniques: the Blocksort algorithm [38] and the LZMA algorithm [70, 71].

(See our discussion at the beginning of Chapter 5 for our rationale in choosing one-dimensional algorithms.) Plainly, these represent only two out of a great many possible choices of algorithms. At this time, we are not aware of any theoretical rationale for any given compression technique in this class to be more or less effective in computing the NCD value, and so we have simply picked two well-known techniques. These comparisons should thus be considered initial explorations of the impact of different compression techniques on an NCD-based decision threshold classifier.

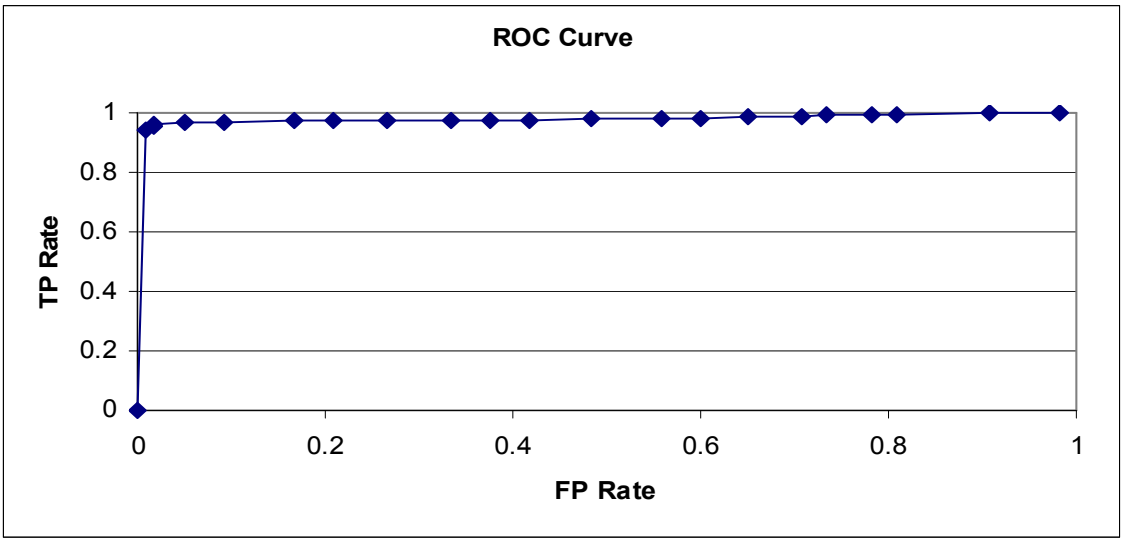

Figure 5.3 ROC curve for Blocksort, aggregated by arithmetic mean

As can be seen in Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6, an NCD-based decision-threshold classifier would work extremely well on this dataset. The classifier would achieve a true positive rate of roughly 95% with a false positive rate of less than 1.7% for all combinations of compression algorithm and aggregation technique. At the "corner" of the curves, LZMA is slightly superior (TPR = 95.6%, FPR = 0.8%), but this difference is miniscule; it amounts to one less false positive at the true positive rate of 95.6%. This compares favorably with existing anti-phishing techniques (excluding

blacklist-based approaches) such as CANTINA [72](TPR = 97%, FPR = 6%), or SpoofGuard [46] [72] [72] (TPR = 91%, FPR = 48%), and is similar to the more recent hybrid approach in [73] (TPR = 90.06%, FPR = 1.95%).

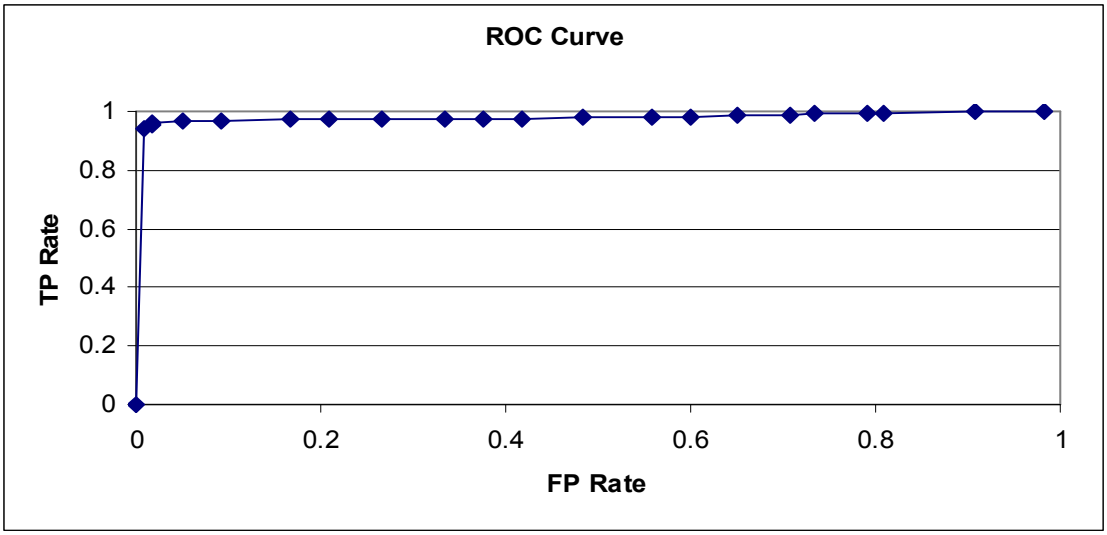

Figure 5.4 ROC curve for Blocksort, aggregated by maximum value

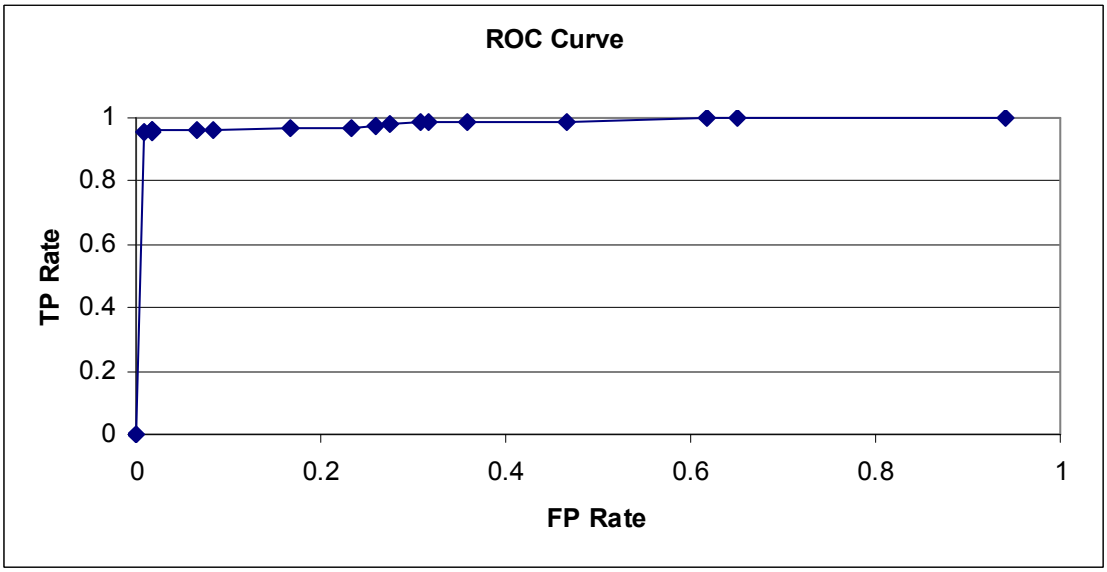

Figure 5.5 ROC curve for LZMA, aggregated by arithmetic mean

In Table 5.9, we compare our results at the "corner" of the curves with CANTINA, Spoof Guard, and the new hybrid method. We present five measures: the first is Cohen's Kappa

statistic [74], which measures the chance-corrected agreement between the actual and predicted classification. Chance-correction means that the Kappa statistic explicitly accounts for classification biases due to uneven class distributions. Matthew's Correlation Coefficient (MCC) is equivalent to Pearson's correlation coefficient for binary data [75]. Precision and Recall are normally presented together, as with TP and FP rates; precision represents the fraction of examples labeled positive that were in fact positive, while recall is again the true positive rate. Finally, the F-Measure is computed as 2*(precision*recall)/ (precision + recall). (Technically, this is the F-1 measure.) As can be seen, our similarity technique with either compressor (there was no difference between averaging & maximum value) compares favorably with the three existing techniques on all five measures. Additionally, LZMA is slightly superior to Blocksort, but only by a small amount. These results also support our earlier finding that our similarity metric effectively discriminates between similar and dissimilar websites. They also indicate that our technique is at least potentially robust against different choices of compression algorithms. This is an important practical finding, as different algorithms may have radically different computational demands (in both time and space), rendering some ineffective on mobile Internet-enabled devices. LZMA, for instance, is a fast and memory-efficient algorithm [70, 71].

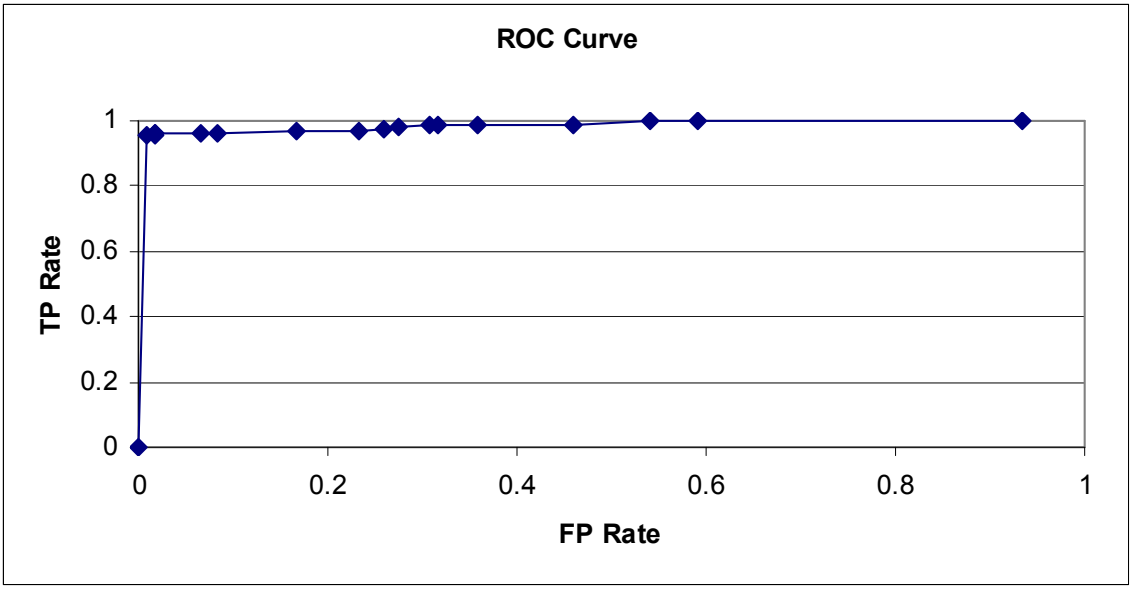

Figure 5.6 ROC curve for LZMA, aggregated by maximum value

Table 5.9 Comparison Against Existing Anti-Phishing Solutions (Without Blacklists)

	Kappa	MCC	Precision	Recall	F-Measure
Blocksort	0.9058	0.9082	0.9935	0.9531	0.9729
LZMA	0.9169	0.9193	0.9967	0.9563	0.9761
Hybrid	0.8345	0.8427	0.9904	0.9006	0.9434
CANTINA	0.91	0.9104	0.9417	0.97	0.9557
SpoofGuard	0.43	0.47	0.6547	0.91	0.7615

5.5 Robustness against Countermeasures

We have argued that our similarity technique could be robust against Phishing countermeasures because it does not use localized features. In this section, we will test this claim using obfuscations based on image processing techniques. This analysis will also indicate how our similarity technique will perform in general when a web page is

obfuscated, either deliberately or accidentally. However, the key characteristic of obfuscations employed by Phishers is that they are deliberately chosen by these adversaries. The sheer variety of possible obfuscations that an adversary could employ to alter the Phishing image has never been considered in the limited existing literature. Human experts readily recognize the difference between innocent similarity and fraud; crafting a computer-based technique to do so is another matter entirely.

Our threat model for these experiments assumes that the Phisher will attempt to evade our similarity technique by manipulating the phishing page. Note, however, that our analysis of the Phishing scam in Section 4.3 leads us to argue that changes which are noticeable to the human being lead to the failure of the scam. Thus, the changes the Phisher makes must pass unnoticed. Specifically, while the Phisher's goal is still to hijack a known brand by visually mimicking the page, they will also attempt to introduce discrepancies which are not visible to the human viewer but are significant to the system evaluating the similarity metric. Clearly, the Phisher has a wide variety of options for pursuing this objective and it is impossible to visualize all of the possibilities. Therefore, in this section, we will seek to provide some initial explorations of this possibility. The results of these explorations seem to hold over a wide range of web pages; however, in this section, the results will be given for a single (legitimate plus Phish) pair, and only for the NCD average value, for the sake of brevity. The pair was randomly selected from the set of pairs with low NCD values before obfuscation; this allows us to observe how progressively increasing the distortion impacts the NCD values. In these trials, we utilize

the LZMA compressor as it seems to perform slightly better than Blocksort in the anti-Phishing classifier, and is known to be fast and memory-efficient.

We assume that the most likely attack possibilities are to change "small" details in the image; that is, in general, the Phisher will work at the pixel level. Broadly, these types of changes can be characterized into two types:

- Non-structural Distortions (such as Luminance Changes, Contrast Changes, Chromatic Distortions, Spatial Shifts, etc)
- Structural Distortions (such as Noise Contamination, Blurring, JPEG Blocking, Wavelet Ringing, etc)

Structural Distortions are unlikely to be effective attacks as they introduce "unnatural" characteristics into the image. Images representing web pages are computer generated and tend to not suffer from structural distortions. By contrast, many non-structural distortions do not degrade image structure or quality and hence are more difficult to detect. Hence in this section, we will first explore the potential impact of introducing such non-structural distortions into phish as the detection of these types of distortions is not perfect. Non-structural distortions based upon spatial shifts will not be actively explored as the compression techniques used in this thesis are robust against spatial shifts which do not impact luminance, contrast, or chromatic characteristics. Unless otherwise stated, the legitimate image is left unaltered in these experiments.

5.5.1 Non-Structural Distortions

Our design for this experiment employs the decision-threshold classifier explored in Section 5.4. As this is only a preliminary exploration, we are only comparing a single legitimate page against one phish targeting it. We introduce controlled levels of non-structural distortion into the image, and seek to determine what level of distortion results in an NCD value exceeding the "best" decision threshold (from Figure 5.3, Figure 5.4, Figure 5.5, Figure 5.6). We then have two judges visually compare the original and obfuscated Phish. For this experiment, the non-structural distortion is a replacement of a pixel value with one of its immediate neighbors (i.e. one of the eight pixels surrounding the chosen one in a 3x3 convolution mask). We control the level of distortion by varying the fraction of pixels chosen for replacement in the image, from 1% to 99%.

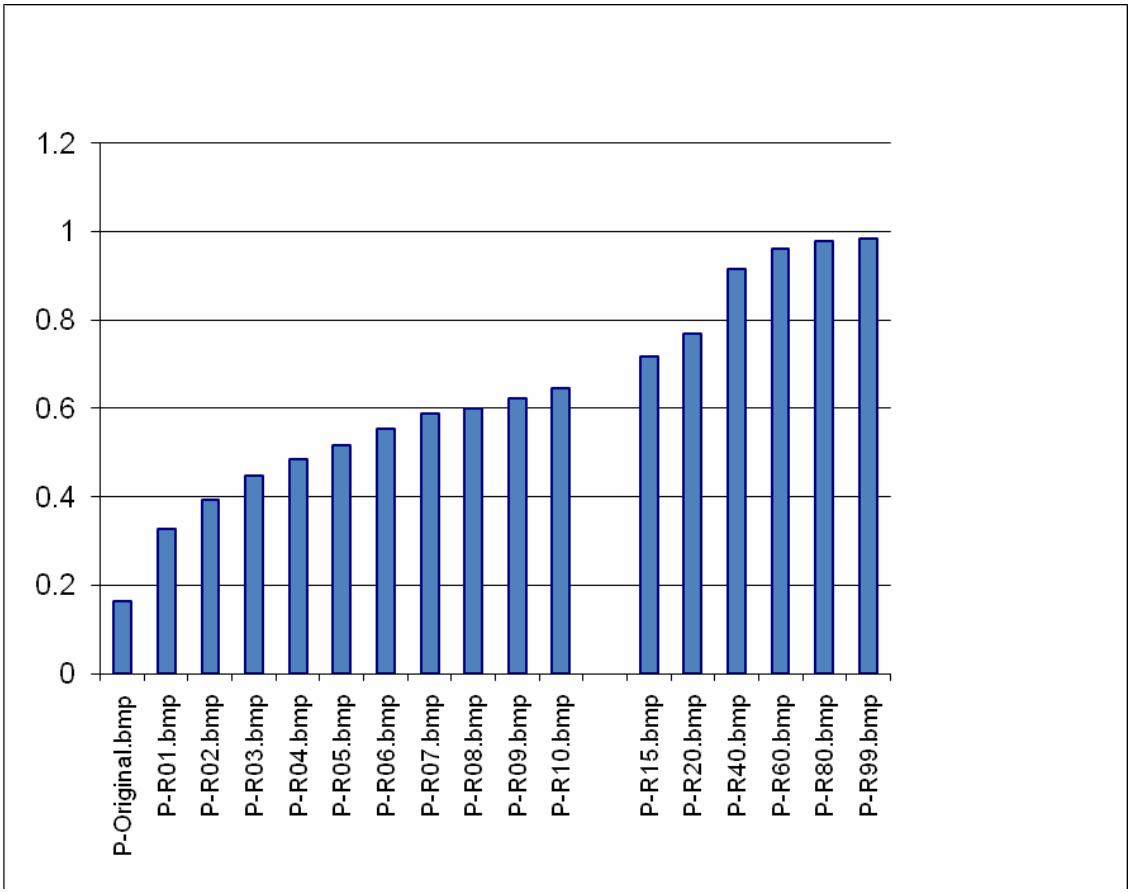

Figure 5.7 The effects of local noise on NCD values

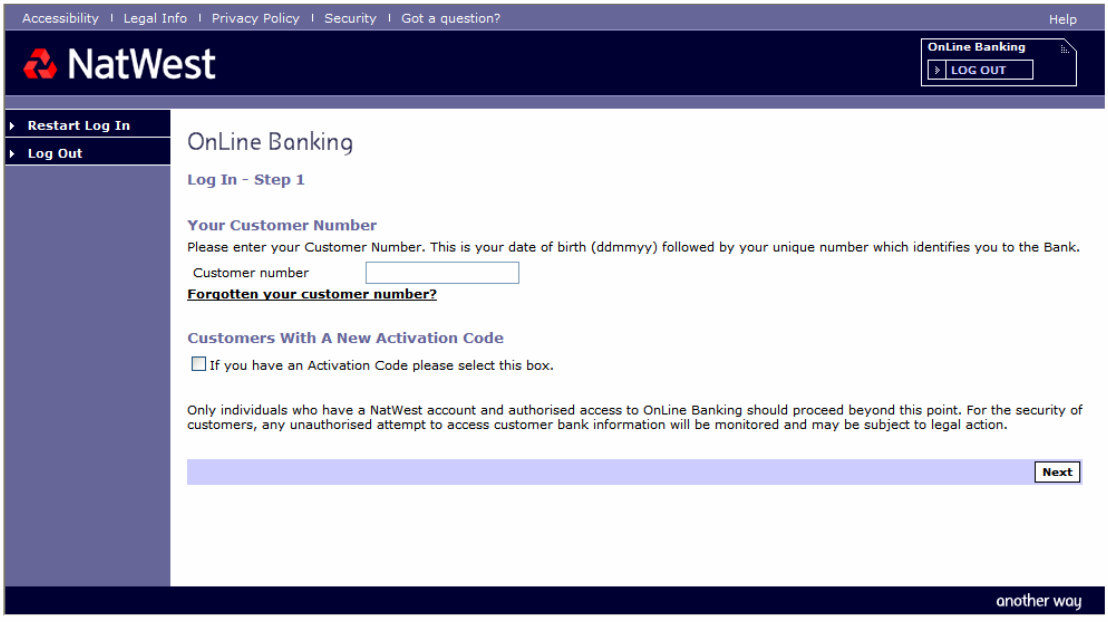

Figure 5.8 Phish before 40% of the pixels have been changed.

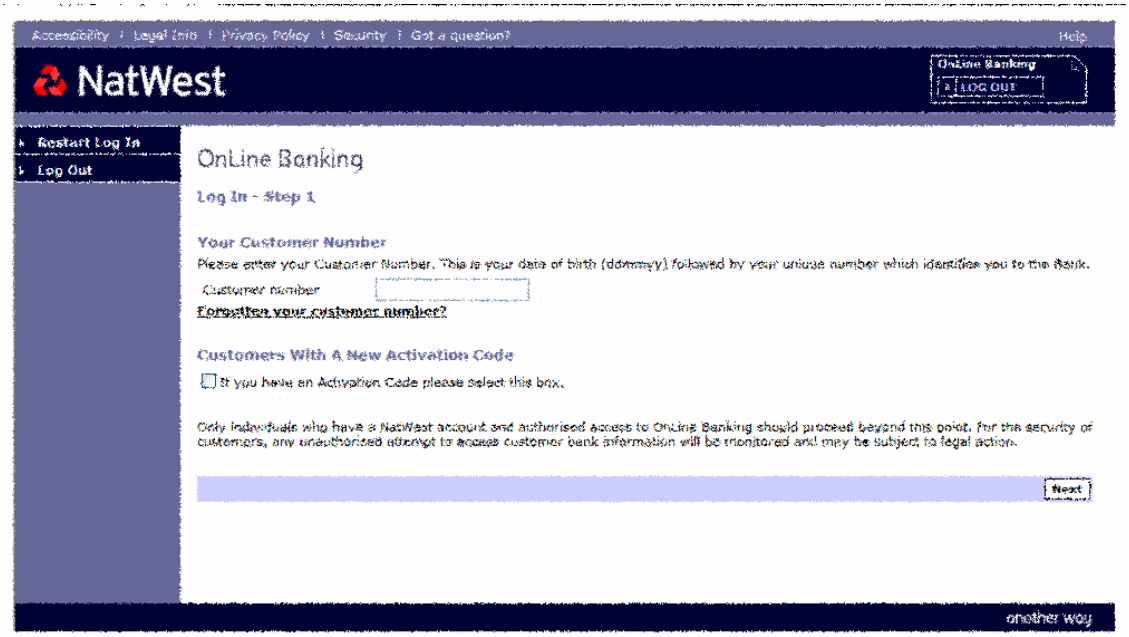

Figure 5.9 Phish after 40% of the pixels have been changed.

Figure 5.7 demonstrates a monotonic relationship between the level of the distortion and the NCD value. In the left-hand side of the figure, the NCD value rises steadily; the x-axis labels "P-Rxx" encode the level of distortion, which begins at 1%, and rises by one percentage point for each category on the left-hand portion of the plot. In the right-hand portion, the NCD value approaches and exceeds the decision threshold value. Using thresholds determined from Section 5.4, the phish is still correctly classified with 40% distortion, but becomes a false-negative error with 60% distortion. We then submit the legitimate and phish pages to our two judges, who unanimously agree that the phish no longer mimics the legitimate web page at 60% distortion – nor at 40%. In Figure 5.9, we present the phish before and after 40% distortion; we believe it is clear that the Phisher

would now have failed in their principal objective of visually mimicking the legitimate page.

5.5.2 Structural Distortions

This experiment explores the issue of detecting "pure" structural distortion. While most distortions have some structural impacts, especially at higher "noise" levels, several types of distortions are considered to be principally structural. Our experimental design is the same as in Section 5.5.1: a controlled level of distortion is introduced into one Phish targeting one legitimate page, and we compare the resulting NCD values against the decision threshold. The legitimate and Phishing pages are then presented to our two judges for comparison. In this experiment, we are introducing random noise into the image; this is done by selecting a fraction of the pixels in an image, and randomizing the RGB color values for those pixels.

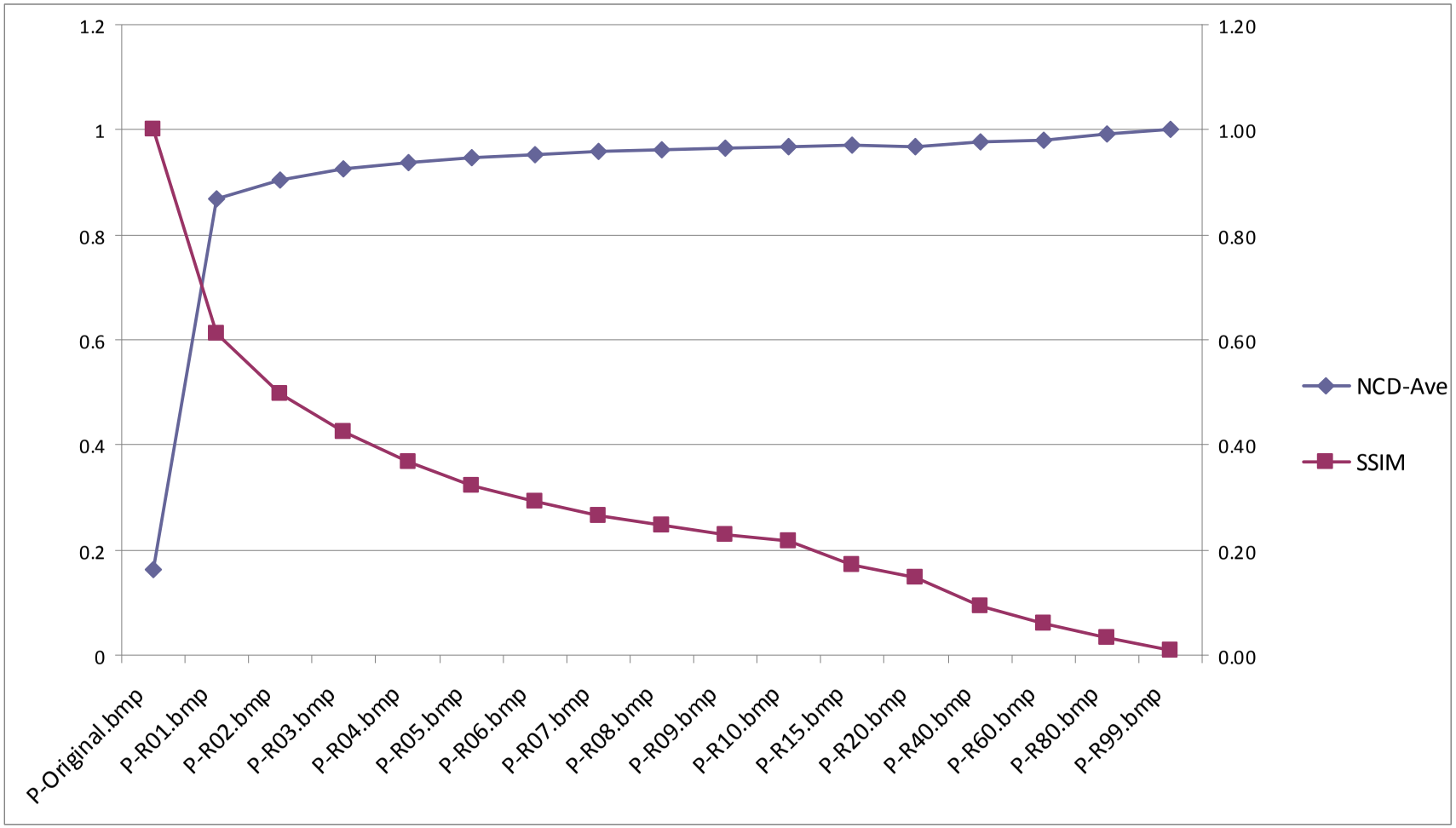

Figure 5.10 Impact of Structural Noise on NCD and SSIM values

Figure 5.10 again shows a monotonic relationship between the distortion level and the NCD value. The decision-threshold classifier is expected to return a false-negative result for distortion levels above 3%, perhaps indicating that our similarity technique is more sensitive to this type of noise than to non-structural noise. However, both judges agreed that 3% distortion made the legitimate and obfuscated phish pages distinct (see Figure 5.11). Again, the level of noise required to fool our similarity technique is great enough to be obvious to a human observer.

These results are consistent with the existing literature on the NCD metric, which generally shows it is quite robust against noise. In [76] they studied how the NCD is affected by noise in the symmetric channel model, in which a random positive integer is added to individual bytes in a file, with the outcomes constrained to the legal domain for values in that file. For instance, genome data is limited to these characters such as A, C, G, T, while text bytes could be any ASCII character, and bytes in a MIDI file can be any integer value in [0,255]. Both theoretical analysis and empirical testing showed that NCD-based clustering degrades slowly with increasing levels of noise. In [77] they examine a noise model in text corpora, in which some percentage of the words in the corpus are distorted by either replacing characters at random, or by replacing characters with asterisks. Six variations on this noise model were tested, and a cluster validity measure was computed from a dendrogram of the corpus. In this case, the NCD was very resilient when the most frequent words were distorted, but less so when words were

randomly chosen. Finally, as noted earlier, the NCD seems to be resilient against translations [62].

In Figure 5.10, we compute and plot an additional metric, the Structural SIMilarity (SSIM) metric [78]. This is an "image quality" metric, which has been empirically shown to "match" human assessments of image quality in a number of experiments, e.g. [79]. We provide this metric to sketch a possible use of our similarity technique as a feature in a robust anti-Phishing classifier. We believe this is the most appropriate usage of our similarity technique in a realistic anti-phishing scenario; the NCD feature is highly discriminative, and a well-designed anti-phishing system should incorporate other complementary features to help defeat Phisher counter-measures. In this experiment, the SSIM values also change dramatically as the distortion level rises.

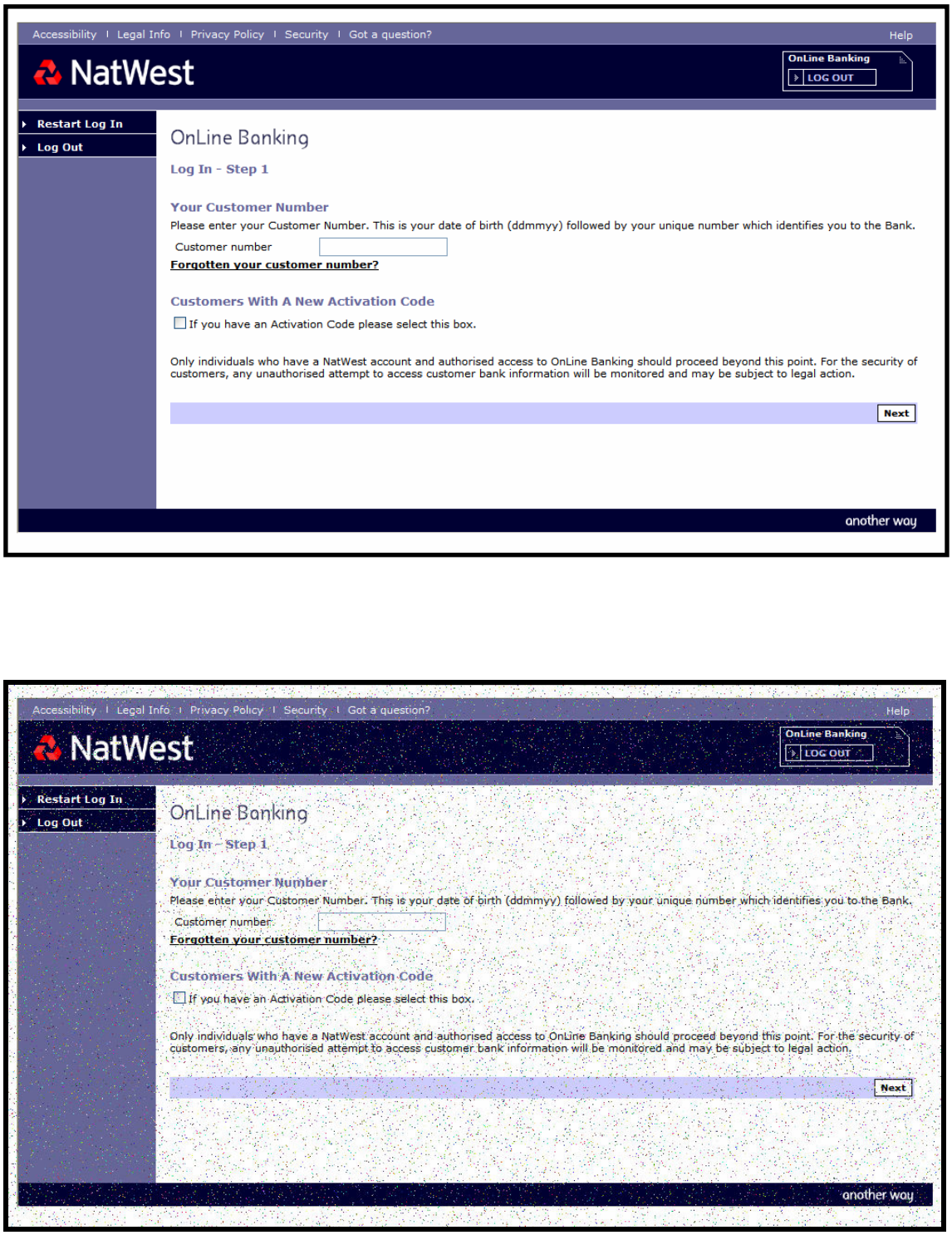

Figure 5.11 Phish before and after 3% structural distortion

We selected the SSIM metric because it is effective in detecting both structural and non-structural distortions. Many non-structural distortions do not degrade image structure or quality and hence are more difficult to detect. However, SSIM is known to be effective at the detection of global luminance shifts and contrast stretching [80]; in addition, a chromatic-variant of SSIM [81] has been shown to be effective at detecting many chromatic, non-structural, distortions. However, as both the legitimate and Phishing pages can reasonably be considered montages of inter-related but independent images, it is unclear how successfully SSIM will be at detecting non-structural distortions. Further empirical analysis will be required to determine if SSIM remains effective in this context.

SSIM, like NCD, is a full reference technique and as the number of brands being phished increases a risk exists that its discrimination performance will not scale. Under these circumstances, it may be necessary to adopt a no-reference image assessment approach (e.g. [82] [83] [84]). It should not be inferred from these experiments that the visual similarity metric is impervious to Phisher countermeasures. It is believed that all Phishing page detection techniques have limitations, and the visual similarity metric is likely to have limitations which the determined Phisher can expose by undertaking some form of image manipulation on their phishing page. However, just as the Phisher can manipulate their page, the authors can manipulate the similarity detection approach to counteract these manipulations. For example, extending the similarity approach to include image reconstruction [85] and seam carving [86] components represents interesting “defensive” possibilities.

5.6 Other related works

In the anti-Phishing literature, [5, 87] are the most closely related research results to the experiments reported in this thesis. In [87], they analyzes the similarity of web pages by comparing HTML tags in the pages. By extracting and comparing regular sub-graphs from the DOM tree representation they construct similarity metrics using webpage structure. Their approaches experienced significant false positive rates; for the identification of 200 Phishing web pages, the approaches experienced false positive rates of 16.90% and 30.29% for the isomorphic subtree identification algorithm and simple tags comparison approach, respectively. Moreover, Phishers can avoid this mechanism by using a combination of images to create a Phishing website that is visually recognized as the legitimate website (as we did with eBay in Figure 2.3). Due to the huge difference in webpage structure, this synthetic Phishing webpage easily evades the similarity metric in [87].

Another closely related approach is presented in [5]. They first convert the webpage into low resolution images, and then extract features from this image (dominant color category and the corresponding coordinate). The Earth Mover's Distance approach (EMD) is then employed to create a feature-based similarity metric. In their evaluation, they demonstrated 8 Phishing web pages (collected from the authors own email accounts) could be successfully identified within a collection of legitimate web sites. This approach again appears vulnerable to obvious countermeasures.

In [88], they use three features which are text pieces/style, images embedded in the pages and the overall visual appearance of the page to identify the Phishing webpages. They compare 41 real Phishing webpages against their corresponding target legitimate webpages. Their overall were FPR=0% and FNR=7.4% (two Phishing pages were missed). Again, their attempt to identify Phishing sites via a feature-based method appears vulnerable to common countermeasures.

Chapter 6 Review of Compression Algorithms

6.1 Introduction

Our experiments in Chapter 5 have demonstrated that our similarity technique is a viable approach to constructing an anti-Phishing mechanism. We now turn our attention to optimizing this system. The literature on the NCD technique plainly indicates that the choice of compressor used is the most important "parameter" in the algorithm, and so must be optimized first. As our input samples are images, we have the choice to compress our subject in a byte-stream or data (one dimensional) fashion or in a rectangle-image (two dimensional) fashion. This two dimensional characteristic makes image compression algorithms different from data compression algorithms. Images can also be compressed in a lossy fashion due to the limits of human vision; this potentially provides higher levels of compression. We will review several popular compression algorithms in this chapter, which will be used in the experiments in the following chapter. We divide our discussion below into data (1-D) and image (2-D) compression algorithms. The development of new compression algorithms is an ongoing field of research, and it is obviously impossible to test every available compression technique. For our purposes, given that our usage of the compressors was (almost certainly) not contemplated during their design, we focus only on well-known, widely-used compression algorithms. These techniques will at least have the advantage of well-understood properties, observed across a huge number and variety of compression tasks. The generalizability of newer and less-used techniques outside of their design specification is at least more questionable.

6.2 Data compression

Data compressors commonly use either statistical, dictionary or block sorting methods to find and remove redundancy from the input stream. Then the compressed data can be decompressed back to the same representation as the original input.

6.2.3 Statistical methods

Statistical methods assign fixed-size codes to the symbols that appear frequently in the input stream. In other words, it is an algorithm based on the probability of occurrence of the symbols. Statistical compression methods commonly have two phases:

1) The modeling phases in which probabilities are assigned to each symbol. A statistical model will count the frequency of occurrence of each symbol in the input stream up to the current point.

2) The coding phase, which encodes the symbol according to its observed probability.

The statistical model for compression can be either static or dynamic. Fixed probabilities are used in the static model. Usually it is effective when compressing certain known, specific types of objects. For example, in the typical English written text object, the letters “E, A, I, R” appear very often. On the contrary, the letters “Z, X, J, Q” are relatively rare. By using the letter frequency table [89] as our static model, we can compress the English text object effectively. The dynamic or adaptive model modifies the statistical model according to the symbol stream observed from the compressor input. For example, assume 112 symbols have been processed by the compressor, of which seven are the symbol “m”. If the next input symbol is “m”, the probability 7/112 is assigned to

it; and the count for “m” is increased by one. The next time “m” shows up in the input stream, the probability will be 8/T, where T is the total number of input symbols at that time point. (The dynamic model is also called an adaptive coder.)

Most of the statistical models are established by either the frequency approach or context approach. The frequency approach is to assign the probability by the symbol’s frequency of occurrence. Shorter codes are assigned to frequently seen symbols. The context approach assigns probabilities which are dependent on the context of a symbol. The context refers to the past text (the symbols already seen in the input stream). For example, the current symbol “K” has probability 0.05 in the past input. According to the past context probability model, there is a high probability to observe the symbol “I” next. In this case, the context based model predicts “I” as the next symbol. If the next symbol is really “I”, it is assigned large probability. If it is not, a small probability will be assigned to the symbol that was observed. The probability model is then updated.

The performance of this compression technique is excellent when the occurrence rate of symbols stays approximately the same throughout the input stream. Usually this method is widely applied to lossless compressors. The idea can be illustrated with a simple Run-Length Encoding (RLE) [70] compression technique. If a symbol K consecutively occurs *n* times in the input stream, the *n* occurrences are encoded as nK. For example, if we have an input stream “KKKKKK”, we encode this as “6K”, thus reducing or eliminating redundancy in the input stream.

PPMD [70] , a variant of PPM [90] (Prediction by Partial Matching), may be the most popular statistical data compression algorithm. PPM uses adaptive statistical data compression based on context modeling. The predicted probability for each symbol is adaptively updated from the frequency count. PPMD estimates the probability of occurrence of a new symbol (originally probability = 0) based upon the ratio of the frequency count. The ratio is calculated by that specific symbol's count divided by the total number of previous observed/processed symbols.

6.2.4 Dictionary methods

Dictionary-based methods select strings of symbols and encode each string as a token in a search buffer (the dictionary). The dictionary can be static or dynamic and maps input symbol sequences to single tokens. There are various dictionary based compression algorithms, all of which are relatives of the LZ (Lempel–Ziv) 77 algorithm [91]. LZ77 uses the previously processed part of the input stream as the dictionary. The encoder uses a technique called sliding windows to scan the unseen input stream for a match from the dictionary. As shown in Figure 6.1, the window is separated into two parts. The left part of the window is called the search buffer, which includes symbols that have already been input and encoded. The right part is called the look-ahead buffer, which contains the text to be encoded. At the very beginning of the compression, nothing exists in both of the buffers. Then the window shifts from left to right, and the first input string enters the look-ahead buffer. As nothing exists in the search buffer (dictionary), no match is found. The window continues to shift until the look-ahead buffer is full. Now the scanned input string enters the search buffer and becomes part of the dictionary. In a practical situation,

the size of the search buffer is very large when compared to the look-ahead buffer (which is usually ~10 bytes).

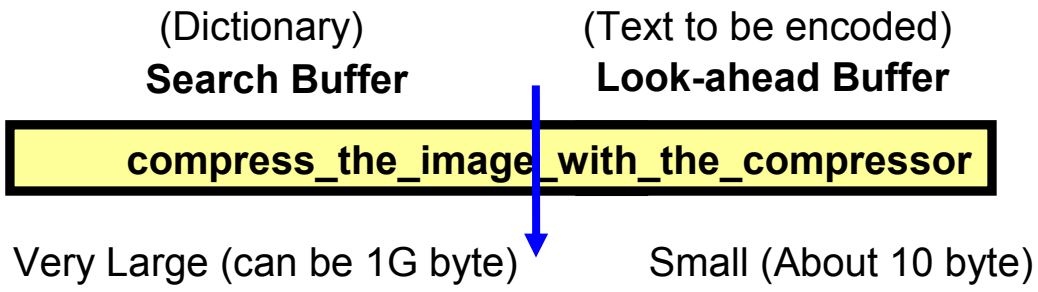

Figure 6.1 The LZ77 Dictionary search

Consider Figure 6.2, in which LZ77 sees the input stream "FGKPPKFKUKPPK PPKPPKU". It has three components in the output stream – the offset, length and next symbol respectively. When the window slides one symbol, there is nothing in the search buffer (dictionary) resulting in (0, 0, F); see Figure 6.3. Similarly (0, 0, G) (0, 0, K) (0, 0, P) are the next three "mismatch" outputs. So far, "FGKP" exist in the dictionary; the next input is P which is a match in the dictionary. The "matched" output can be represented as (1, 1, K) which refers to the offset or distance 1 for the match in the dictionary, length 1 for the match and the next symbol K to be in the dictionary. The next character results in another match (6, 1, K) ; followed by the "mismatched" result (0,0,U). By finding a match at offset 7 for 4 symbols in the dictionary, we can acquire the output (7, 4, P). Finally (3, 5, U) is output using a specific characteristic of LZ77. For the last six input symbols, initially a match with the length 3 (PKP) is found in the dictionary. In this case, the next two symbols are "PK" which exists in the match "PKP". Consequently, LZ77 extends the length of its match from 3 to 5 which result in the output (3, 5, U). This special design is to deal with repetitive input such as PPPPPPPP. With this extendable

characteristic, we can reduce the searching time in the dictionary dramatically. The whole process can be seen in Figure 6.4.

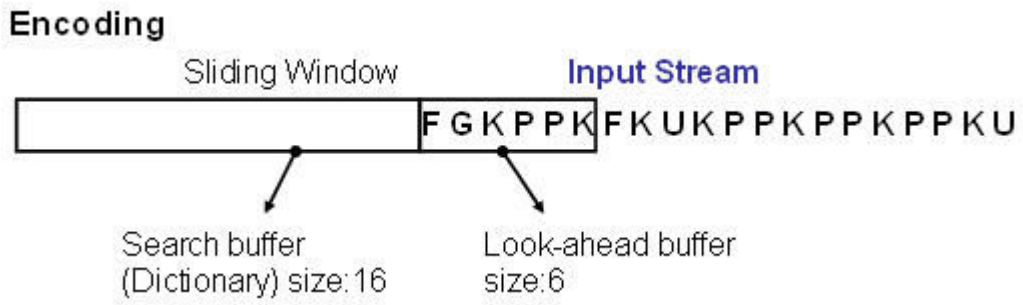

Figure 6.2 The encoding of LZ77

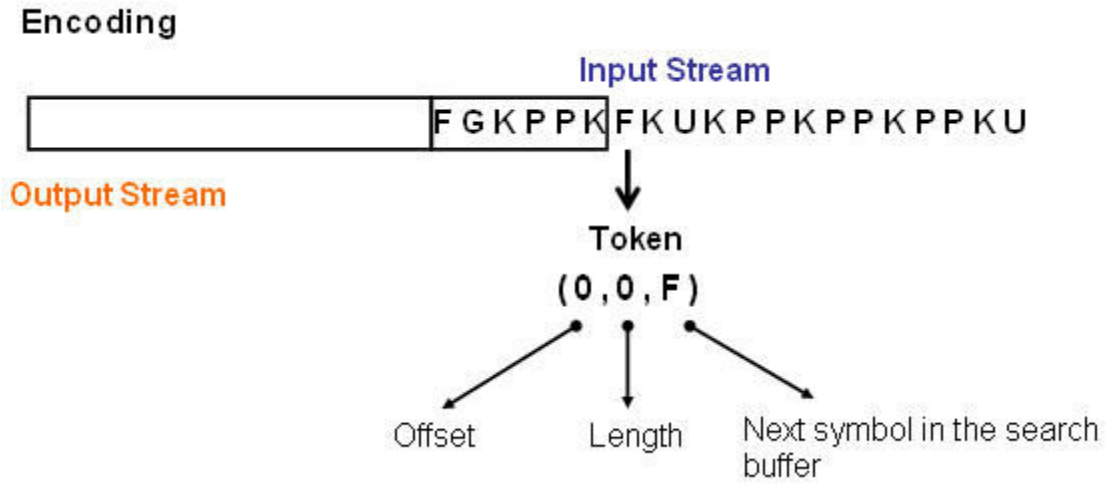

Figure 6.3 The encoding output of LZ77

Figure 6.4 The encoding procedure of LZ77

The decoding process is much easier. First, by the first two components (offset and length) in the output data, we know the status information (match or mismatch) of the dictionary comparison. If we have (0, 0) we get a mismatch. If the first two components in the output are anything else, we get a match. The first component refers to the match offset or the distance in the dictionary and the second component refers to the length for the match. In our case, the first four output (0,0,F), (0,0,G), (0,0,K), (0,0,P) refer to four mismatches. Therefore, the decoded input should be FGKP. For (1, 1, K) refers to the fact that we get a match and its offset and length are 1 and 1 respectively. And the K is the next symbol in the search buffer. Consequently, the token (1, 1, K) is decoded as PK. We can decode (6,1,K) as FK, (0,0,U) as U, (7,4,P) as KPPKP and (3,5,U) as PKPPKU. The details can be seen in Figure 6.5.

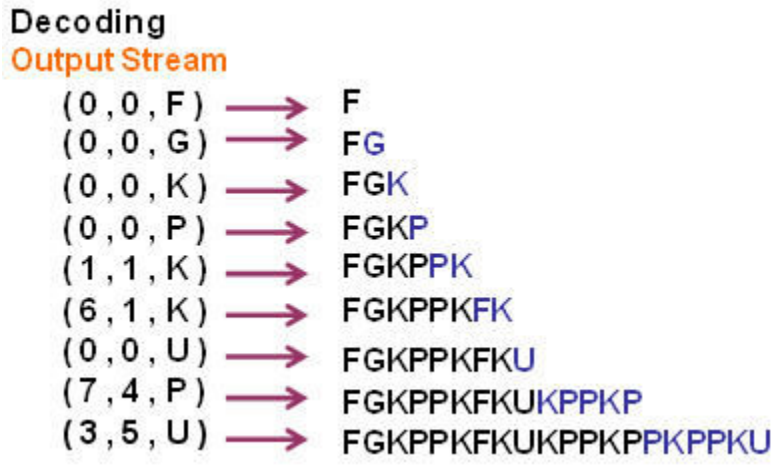

Figure 6.5 The decoding process of LZ77

6.2.4.1 Deflate

Deflate is a lossless data compression algorithm based on the variation of LZ77[91] combined with Huffman encoding. The compression is accomplished via duplicate string elimination and bit reduction. The elimination starts with finding the match in the input stream then replacing the duplicate symbols with pointers based upon the frequency of use. The finding/matching strategy is the same as the sliding window technique used in LZ77. Bit reduction is done by using dynamic Huffman encoding which generates an optimized Huffman tree for each block in the input stream. Huffman encoding [92] tries to replace each symbol to a better representation based on its occurrence frequency.

6.2.4.2 LZMA

LZMA (Lempel-Ziv-Markov chain-Algorithm) is variation of LZ77 [91] and uses adaptive binary range coding [93]. LZMA provides fast decompression speed, high compression ratio, and low memory requirements. Just like LZ77, LZMA uses a sliding-window buffer, using the previously-scanned input stream as the dictionary for the next input. After the search buffer has found the longest string that matches the look-ahead buffer, it writes the output into the compressed stream. Eventually, the individual outputs are encoded by adaptive arithmetic binary range coding to achieve bit reduction. This encoding transforms a string of characters into a fixed number of bits (binary) per character based on its frequency of occurrence in the previous buffer. More frequent tokens in the input stream are assigned fewer bits for storage.

LZMA is similar to Deflate, but uses adaptive range coding instead of Huffman coding. Range coding is very similar to the arithmetic coding [94]. The only difference between them is that range coding using a huge range number instead of the number fraction of arithmetic coding for their coding output. In many aspects, arithmetic/range coding is superior to the Huffman coding [95]. Unlike Huffman coding, which separates the input stream into block symbols and replaces each component with a code, arithmetic range coding encodes the entire message into a single fraction (between 0 and 1).

6.2.5 Block sorting

Block sorting [38] (also called the Burrows–Wheeler transform) is a lossless data compression algorithm which applies a reversible transformation (reordering) to a block

of input text. The total input stream is first divided into a number of multi-symbol blocks. Then these units are processed by a reversible transformation for reordering to form a block containing the same characters for earlier compression. The reordering process tends to group the same characters together and sort them into a lexicographical order which improves the probability and efficiency of finding a character. The main benefit for this lexicographical reordering action is to simplify finding redundancy (because the block is already in lexicographical order). This improvement can make a significant contribution when augmenting a local adaptive algorithm such as Huffman or arithmetic coding.

6.2.5.1 Blocksorting with the statistical method

We used the Blocksorting compression method from CompLearn 0.9.7 [96]. This specialized block sorting method first uses the same technique for the lexicographical reordering. When it finishes the sorting process, it uses a dynamic state lookup table to encode the input symbols by their occurrence probability. By keeping statistics on state transition frequencies through this state table that adapt to the input stream, we can encode the reordered input stream more efficiently.

6.2.5.2 Blocksorting with Huffman coding (Bzip2)

Bzip2 is a block-sorting compressor in which frequently-recurring character sequences are encoded by a move-to-front transform and Huffman coding [97].

6.3 Image compression

Similar to data compression, the main objective of image compression is to reduce the file size for an image. By definition, a digital image is a rectangular array of pixels arranged in rows and columns. Depending on different applications across various fields, there are two kinds of image compression techniques – lossless and lossy. Lossless image compression is like data compression, in that its goal is to recover an image that is identical to the original input after decompression. Lossy image compression focuses on further reducing the file size by allowing the creation of some imperceptible differences. Losing information in exchange for better compression is a key characteristic for lossy methods. When the compressed result is recovered or decompressed, the result is not the same as the original input stream. If the loss is rather small but acquire lots of improvement for compression, we can deem it as an acceptable tradeoff. For example, digital photos and web images use lossy compression for smaller file size and faster Internet transmission. The loss is "small" in the sense that human vision is essentially unable to detect the difference, meaning that there is no practical degradation in performance – at least in this use case.

6.3.1 Lossless image compression

Just like data compression, lossless image compression attempts to remove redundancy present in image signals. This redundancy is proportional to the amount of correlation among the image data samples. Compressing an image using RLE may help illustrate lossless image compression. When we randomly pick a pixel in the image, it is very possible that the next to-be-compressed pixel has the same bitmap which creates high

correlation (spatial redundancy). Hence, the compressor scans the image row by row to find spatial redundancy.

6.3.1.1 GIF [98]

GIF (Graphics Interchange format) file format was developed to replace the earlier RLE format which was black and white only. It is widely used to on the World Wide Web. It is popular for web graphics because of its optional interlacing feature, which can partially download the image. With this, a user can abort the download if they so choose, saving Internet transmission time. This file format employs a variant of the Lempel-Ziv-Welch (LZW) lossless data compression technique to reduce the file size and avoid the loss of visual quality. LZW is a dictionary based compression algorithm which uses a dynamic, growing dictionary. Unlike some dictionary methods, the dictionary initially contains the single character strings for all kinds of the possible input character. For example, in the case of 1 byte symbols, the first 256 entries in the dictionary are used to store their predefined corresponding symbol. Due this initialization of the dictionary, the first input character can be always found in the dictionary.

The algorithm scans through the input stream, successively finding longer substrings until it can not find any longer one in the dictionary. Once it can not find a longer substring, the index (in the dictionary) for that specific string matched part is sent to output and this new string (including the last character) is added into the dictionary. The last input character is then used as the next starting point to scan for substrings. Therefore, longer strings can be added into the dictionary and be available for the next encoding process. Input streams contain repeated context patterns are well suited for LZW compression

algorithm. With the initialization of this flexible dictionary, it gets better compression performance as the input stream grows. Importantly, the GIF format scans the input stream row by row, so it can only discover the correlations or redundancy of pixels in the rows but not between them. Or we can say GIF can not find the redundancy of pixels in columns.

6.3.1.2 PNG [99]

The PNG (Portable Network Graphics) file format employs the dictionary-based, lossless data compression Deflate method to encode the difference between pixels. Basically, there are two steps for the compression:

1) It converts pixel values to numbers by a process called delta filtering which calculates a "predicted" value for each pixel. This prediction is done by evaluating the pixel value based on the neighboring pixels. Then this pixel will be replaced with a predicted difference value which is the offset of this original pixel and its predicted value.

2) Then the Deflate compression algorithm (see Section 6.2.4.1) is applied to encode these predicted differences.

Just like the characteristic of the GIF scanning pattern, filtering is done row by row on each image. The correlations or redundancy of pixels between rows are not able to be found or discovered by this process.

6.3.1.3 JPG (Lossless)

Although JPG is famous for its lossy compression, it also provides a lossless mode which uses a completely different technique in the same compression package. The JPEG

lossless mode uses a simple predictive coding model called DPCM (Differential Pulse Code Modulation) [100]. It predicts (based on previous samples) the nearest neighboring pixels and codes its prediction error. DPCM encodes differences between the predicted pixels but does not encode each pixel separately. The encoding method can be either Huffman or arithmetic coding. Usually arithmetic coding is applied for better compression [95].

6.3.1.4 JP2 (Lossless)

JPEG 2000 [101] is a wavelet-based image compression algorithm. For the normal RGB color image for example, it is divided into three components. Each component is partitioned into rectangular called tiles which are compressed individually. First, the sub-bands of wavelet coefficients are computed by a wavelet transform. Wavelet analysis is an approach to identify and isolate the frequencies that represent the signal in that specific time interval. The tiles now have been transformed into the form of wavelet coefficients. The coefficients are quantized by the user specified bit rate. The quantized coefficients then are encoded arithmetically by the block based designated encoder. The bits generated from the coding process are in blocks called packets. These packets are the components to be rearranged for construction of the bit stream. The bit stream is organized in several layers which provide high-resolution image information for decoding. The reversible integer wavelet transform is used in the first step and the sizes of all quantization steps are set to 1 for lossless compression.

6.3.2 Lossy image compression

Lossy image compression is done by removing “irrelevant” or “unimportant” information based on human perception. A lossy image encoder must delete the information whose absence would not be noticed by humans. Therefore, the compression algorithms have to account for the physiology of human vision. Consequently, the point is to decide which part of information or details of image would be “unnoticed” by the viewer or “irrelevant” to a human. Usually the brightness of neighboring pixels in an image is highly correlated. The pixel and its neighbor may have different colors, however, their brightness are normally similar. Accordingly, we can convert RGB pixel components into another three-component representation, YCbCr. Y, or “luminance”, represents the brightness of the pixel, and the other two CbCr represent its color. Human perception is more sensitive to changes in brightness than changes in color. Accordingly, some loss of the Cb and Cr color component information during the compressing process becomes imperceptible to humans. This compression distortion is the core concept of lossy image compression to reduce image file size.

Another issue for the lossy image compression is the image compression ratio, also called the visual fidelity issue. Human perception is very subjective from person to person; therefore, people may have different tolerances for the quality of the image. To perform the lossy image compression, users have the choice to choose what image quality they can accept. However, the subjective image quality is a trade-off for data compression. The better image compression you can achieve, the poorer image quality you may acquire.

When we perform the lossy image compression, its ratio must be adjustable to fit different needs/requirements.

6.3.2.1 JPG (Lossy)

JPEG [102] is a lossy compression method which is parameterized to allow users to decide on the tradeoff between storage size and image quality. First, color images are transformed from the RGB (Red, Green, and Blue) additive color model into YCbCr (luminance, chrominance) alternative image format. The loss of chrominance information is accomplished by down sampling which creates low resolution pixels from the original image. Then the DCT (Discrete Cosine Transform) is applied to each 8*8 pixel data unit. This generates a quantization coefficient (a sum of cosine functions at specific frequencies) which will be compressed by the hybrid compression algorithm of RLE and Huffman encoding [70]. DCT transforms the data unit in terms of a sum of cosine functions which oscillates at different frequencies. Small high-frequency components can be discarded in this case to acquire better compression.

6.3.2.2 JP2 (Lossy)

JPEG 2000 also provides an option to manipulate the tradeoff between file size and image quality. Basically, the JPEG 2000 lossy compression algorithm is performed by the following four steps [103]:

1. A wavelet transform of the image.
2. The embedded encoding of block subdivisions of that wavelet transforms.

3. Optimal truncation of transformed values for optimizing PSNR (Peak Signal-to-Noise Ratio) values.
4. Encoding the raw binary output generated by step 3 using a sophisticated arithmetic encoding.

In the lossy mode, either a reversible or a nonreversible integer wavelet transform can be used for the computation of the sub-bands of the wavelet coefficients. However, to achieve the better compression, the nonreversible is normally used. JPEG 2000 compression is based on a fundamental idea which is "compress once, decompress many ways"[101]. Its encoder simply uses the maximum image quality Q and maximum resolution R to perform the compression. The decoder has the option to choose the image quality up to or including Q and the image resolution less than or equal to R. Another important feature of its decompression is it can also decompress parts of the user designated image or just exact parts of the compressed stream and reconstruct it to another new compressed stream. This gives the user the freedom to crop and transform the image without any decompression process which save time and file space. For example, the rotation of the image (90 or 180) can be done directly in the compressed stream without decompressing the image.

Chapter 7 Compression Algorithms for Anti-Phishing

7.1 Introduction

In [32], they describe the NCD as a parameter-free, universal similarity distance which can utilize nearly any compressor. They demonstrate NCD with various compressors (gzip, bzip2, PPMZ) can acquire an excellent clustering result for Euthrian Orders in Genomics and Phylogeny field, and argue that "NCD can minorize all similarity metrics based on features that are captured by the reference compressor involved". In [104], [105], [106], they cluster protein sequence, general textual documents, and chain letters using PPMZ, gzip, and bzip2, respectively. Moreover, in [107], they performed all their experimental subjects only with the CompLearn [96] default compressor. In [108], the PBM image file format is utilized as the sample for their image clustering experiments. Only the Bzip2 compression algorithm is applied in their tests. The reason for this selection is described as: "We chose the Bzip2 algorithm because it is a block compression algorithm which ensures that the compression of the concatenation of two objects will not vary appreciably with the ordering of the objects in the concatenation." These articles appear to uncritically accept the argument proposed by [32] that clustering based on NCD is robust across application fields and compressors. On the other hand, there is a modest literature examining the characteristics of different compressors within the NCD algorithm. In [109], an image registration technique based on the NCD explored both image (JPEG 2000) and data (bzip2) compressors; the image compressor was found to be more sensitive to translations, and thus more appropriate than data compression. In

[110] modified LZ78 and PPM compression algorithms are used to cluster grayscale image samples; specifically, the dictionary elements of LZ78 were used to form a new kernel function for support vector machines.

Our application is to identify Phishing web pages by detecting a visually similar image amongst the webpage screenshot images of our protected sites. Therefore, an accurate (high TPR, low FPR) NCD classification result directly leads to successful Phishing webpage identification in our algorithm. As the impact of different compressors on the NCD clustering result in this domain is still relatively unexplored, the issue we want to explore here is:

Do different compressing algorithms have any effect on the result of NCD classification method for our subject?

An empirical evaluation of the impact of different compressing algorithms can help us to explore this unknown territory. According to the acquired result, we will select the best compressor for further analysis and evaluation in our anti-Phishing application.

7.2 Empirical evaluation

The objective of this evaluation is to employ several different compression algorithms over exactly the same samples to generate two different NCD populations which represent the comparison outcome of visually similar images and dissimilar images,

respectively. By analyzing the results, we are able to find the most appropriate compression algorithm for our application.

7.2.3 Design

We designed the experiment with two groups:

- 16 legitimate webpage images; and
- 20 respective Phishing webpage images group for this purpose.

Then we use these two groups to generate two NCD populations (the population of NCD values of visually dissimilar websites, i.e. the legitimate against legitimate-LL webpage image population; and the population of visually similar websites, i.e. the legitimate against Phishing-LP webpage image population) for clustering purpose. These two populations are:

- The LL NCD population: total 120 NCD values-(16*16-16)/2=120 comparing legitimate against legitimate webpage images; and
- The LP NCD population: 320 NCD values-20*16=320 comparing legitimate webpages against Phishing webpages targeting them.

We can estimate the compression algorithms' impact from their ability to separate these two populations. Nine compression algorithms plus two additional parameterizations (0.5 and lossless for each of the two lossy compressors) will be examined in the experiments.

7.2.3.1 The collected samples for two groups

Although the requirement of the samples for these experiments is almost the same with the samples we collected in Section 5.3.1, we still made the decision to recollect our test data. The reasons for doing this are:

1) The Phishing websites adapt to existing Anti-Phishing mechanism-it is important to keep our Phish "fresh".
2) To examine if our experimental results vary over time.

We recollected 16 different legitimate web pages and 320 Phishing web pages targeting them as the samples in this experiment (Table 7.1).We chose legitimate websites in Table 7.1 based on the frequency with which Phishing sites attempt to imitate them. The Phish were captured in the PhishTank [55]. In addition, three Italian, one French and one Spanish website were added to the group to increase the language or regional diversity of the sample. There are a total of 336 samples are collected for the experiment.
The Phishing samples are collected during the period of August 6-October 13, 2009 from the PhishTank website which gathers the link of Phishing websites from the real world. All the samples were collected by the purchased version of "Convert HTML to Image Ver.1.1"[111] which uses a virtual browser to capture the image of the website from the PhishTank links into the BMP file format..

Table 7.1 The two groups of collected samples for the Experiment

Group **One**	Group **Two**
01-Abbey-L	01-Abbey-P1 ~ Abbey-P20
02-Alliance(uk)-L	02-Alliance(uk)-P1 ~ Alliance(uk)-P20
03-BOA-L	03-BOA-P1 ~ BOA-P20
04-CartaSi-L	04-CartaSi(it)-P1 ~ CartaSi(it)-P20
05-ebay(it)-L	05-ebay(it)-P1 ~ ebay(it)-P20
06-paypal(fr)-L	06-paypal(fr)-P1 ~ paypal(fr)-P20
07-Tibia-L	07-Tibia-P1 ~ Tibia-P20
08-Halifax-L	08-Halifax-P1 ~ Halifax-P20
09-Lloydstsb-L	09-Lloydstsb(uk)-P1 ~ Lloydstsb(uk)-P20
10-Bradesco(br)-L	10-Bradesco(br) ~ Bradesco(br)-P20
11-PosteItaliane(it)-L	11-PosteItaliane(it)-P1 ~ PosteItaliane(it)-P20
12-Chase-L	12-Chase-P1 ~ Chase-P20
13-facebook-L	13-facebook-P1 ~ facebook-P20
14-cahoot(uk)-L	14-cahoot(uk)-P1 ~ cahoot(uk)-P20
15-Warcraft-L	15-Warcraft-P1 ~ Warcraft-P20
16-WellsFargo-L	16-WellsFargo-P1 ~ WellsFargo-P20
Total samples count	
16	**16*20=320**

7.2.3.2 Compressors for the test

As shown in Table 7.2, nine compressors and two options (ratio=0.5 and lossless) for JPG and JPEG2000 compressors are utilized during the test. (JPG and JP2 each for two settings)

Table 7.2 The selected compression algorithms

Compressors
1. Block sorting
2. LZMA
3. PPMD
4. Bip2
5. Deflate
6. GIF
7. PNG
8. JPG-(0.5)
9. JPG-(Lossless)
10 JP2-(0.5)
11 JP2-(Lossless)

7.2.3.3 The image quality for lossy JPG and JP2

When we apply the lossy image compression algorithm to our NCD calculation, the first issue we should consider is the image quality ratio. We use MOS (Mean opinion score) [112] which is a numerical indication of the quality of received objects after compression (as perceived by human subjects), specifically MOS=3. In most situations, a user can not tell the difference between the original webpage captured image and the modified MOS=3 webpage image. In [113] it was determined that commonly setting quality ratio=0.5 for both JPG and JP2 results in MOS=3.

7.2.3.4 The calculation of compressed result *C* (*xy*)

For the data based compressors such as Block sorting, LZMA, PPMD, Bip2 and Deflate, which simply deem our image objects *x* and *y* as two binary streams, and the compressed result $C(xy)$ is computed by concatenating the object *x* and *y* first and applying our selected data compressor to it. The only difference is the sequence of concatenation for $C(xy)$ and $C(yx)$. In Section 5.3.3, we have found the choice of using arithmetic mean or

maximum to aggregate corresponding NCD values-NCD(x,y) and NCD(y,x) or C (xy) and C (yx) does not influence our outcome statistically. Therefore, we use the average value of them.

The situation becomes different when we calculate NCD values by using image based compressors such as GIF, PNG, JPG and JP2 which deem our compared image objects x and y as a two-dimensional square object. Therefore, there are two ways (horizontal and vertical scan directions) to concatenate the image objects. This yields the following equations:

$$NCD(x,y)average-horizontal=\frac{NCD(x,y)horizontal+NCD(y,x)horizontal}{2} \tag{7.1}$$

$$NCD(x,y)average-vertical=\frac{NCD(x,y)vertical+NCD(y,x)vertical}{2} \tag{7.2}$$

$$NCD(x,y)aggregated=\frac{NCD(x,y)average-horizontal+NCD(x,y)average-vertical}{2} \tag{7.3}$$

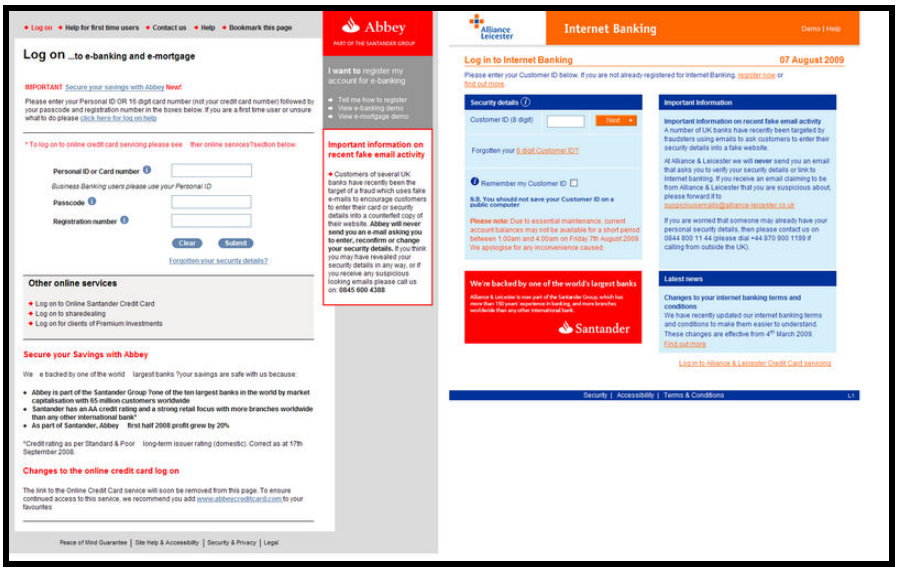

Figure 7.1 The horizontal (left-right) concatenation for C(*xy*)

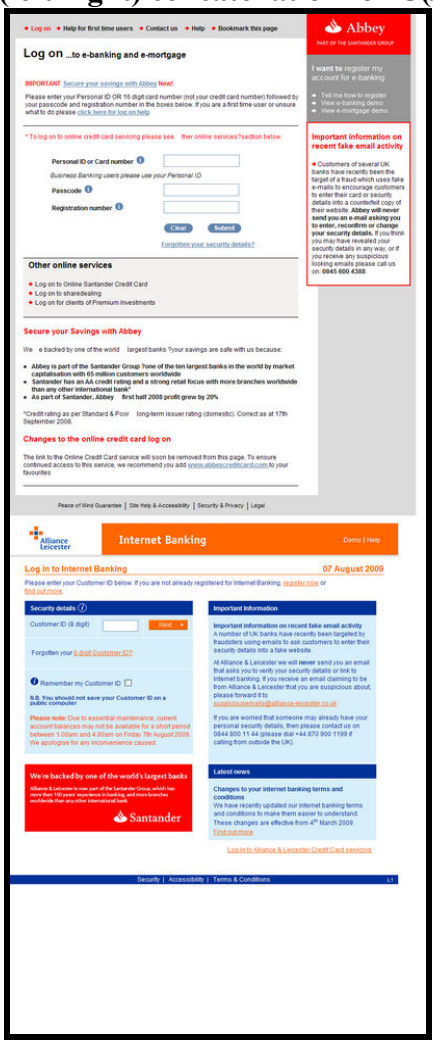

Figure 7.2 The vertical (top-bottom) concatenation for C(*xy*)

7.2.4 Methodology

Following the same methods as in Section 5.3.2, we create two populations: the LL population of 120 dissimilar webpages, and the LP population of 320 similar webpages. We compute NCD values across these datasets for eleven compressors in this experiment.

As in Chapter 5, no optimal method exists for analyzing our experimental data; hence, we will use several statistical methods (*Z*-test, effect size, and ROC curves) for the comparison of the various compression algorithms. First, we use a *Z*-test to examine if our null hypothesis can be rejected for the two populations generated for our selected different compression algorithms. Second, the effect size [114] will give us the idea of quantification of the difference between our two populations for compressors. Finally, the ROC curve will be used to check the identification performance which means the true positive rate (sensitivity) and false positive rate (100-specificity) of our different compressors. By the comparison results for our nine selected compression algorithms, we can tell if the difference made for various compression algorithms.

In the ROC curve, we focus on two indicators in the graph for the classification performance evaluation which are the corner point and AUC. The corner point (on the upper left) of the ROC curve is the representation of the highest accuracy which has the highest TPR and the lowest FPR. At the corner point, the higher TPR and the lower FPR, the better classification ability will be. AUC (Area Under Curve) can be interpreted as the probability that a randomly selected sample from the "positive" group has a test value larger than a randomly selected sample from the "negative" group. When there is no

difference, AUC=0.5. The greater AUC value, the better identification performance will be.

7.3 Interpretation and the evaluation result

7.3.1 Null Hypothesis

From the result given in the Table 7.3, we can reject every null hypothesis with $P<0.05$. Consequently, the hypothesis we made- that NCD values in the dissimilar population (LL population) are significantly different (or higher) from the similar population (LP population) - is supported.

Table 7.3 The z-test result

Z-test Compressor	*z* statistic	***P*** Significance
1. Blocksorting	31.052	<0.000001
2. LZMA	**34.654**	**<0.000001**
3. PPMD	18.436	<0.000001
4. Bzip2	13.894	<0.000001
5. Deflate	4.468	0.000004
6. GIF	14.156	<0.000001
7. PNG	22.504	<0.000001
8. JPG-(0.5)	13.122	<0.000001
9. JPG-Lossless	11.864	<0.000001
10.JP2-(0.5)	3.309	0.000467
11.JP2-(Lossless)	2.941	0.001635

7.3.2 Magnitude of Effect

The greater value of the effect size, the better ability of the similarity classification will be. From the result given in the Table 7.4, we have found LZMA has the largest effect size. JPG-0.5 and Blocksorting have almost the second and the third highest values in the test respectively.

Table 7.4 The result of effect size

	LP Group			LL Group				
Compressors	Means	Num*	SD*	Means	Num*	SD*	SD (Pooled)	Effect size
1. Blocksorting	0.5301828	320	0.2850288	1.0271639	120	0.0165406	0.2856484	1.7398349
2. LZMA	0.4131593	320	0.3159146	0.9930296	120	0.0071033	0.3164372	**1.8324971**
3. PPMD	0.7989296	320	0.2068179	1.0125373	120	0.0083009	0.2072023	1.0309135
4. Bzip2	0.8508287	320	0.2072941	1.0134066	120	0.0177794	0.2078986	0.7820056
5. Deflate	0.9986314	320	0.0017288	0.9992272	120	0.0010065	0.0018377	0.3242234
6. GIF	0.9734981	320	0.0737602	1.0650414	120	0.0545724	0.0810618	1.1293038
7. PNG	0.8129697	320	0.1600499	1.0151954	120	0.0091596	0.1603969	1.2607826
8. JPG-(0.5)	0.99665053	320	0.003269921	1.0085781	120	0.0097542	0.006806536	1.7523711
9. JPG-Lossless	1.00002474	320	0.003056542	1.0120284	120	0.0109238	0.0073502	1.6331203
10.JP2-(0.5)	1.00120784	320	0.002113665	1.0019264	120	0.0019955	0.0024437	0.2940486
11.JP2- Lossless	1.00166554	320	0.00223995	1.0023484	120	0.0021416	0.0025980	0.2628289

* Num stands for number of samples; SD stands for Standard Deviation

7.3.3 ROC Curve

From the result given in the Table 7.5, we have found LZMA and PPMD compressor have the highest TPR and lowest FPR. From the result given in the Table 7.6, we have found Blocksorting, LZMA and PPMD have the highest AUC value. From the ROC curve plot shown in Figure 7.3 and Figure 7.4, we have found the graph for Blocksorting, LZMA and PPMD compressors provide the best performance.

Table 7.5 TPR FPR at the corner point of the ROC curve for different compression algorithm

	1.Block sorting	**2.LZMA**	**3.PPMD**	4.Bzip2	5.Deflate	6.GIF	**7.PNG**	8.JPG-0.5	**9.JPG-Lossless**	10.JP2-0.5	11.JP2-Lossless
TPR	**99.69%**	**99.99%>>**	**99.99%>>**	66.87%	45.63%	87.81%	**99.69%**	78.44%	**97.19%**	62.5%	51.56%
FPR	**<<0.01%**	**<<0.01%**	**<<0.01%**	2.5%	20.83%	11.67%	**2.5%**	16.67%	**27.5%**	37.5%	26.67%

Table 7.6 AUC of the ROC curve for different compression algorithm

	1.Block sorting	**2.LZMA**	**3.PPMD**	4.Bzip2	5.Deflate	6.GIF	7.PNG	8.JPG-0.5	9.JPG-Lossless	10.JP2-0.5	11.JP2-Lossless
AUC	**1.000**	**1.000**	**1.000**	0.868	0.604	0.933	0.999	0.902	0.922	0.625	0.613

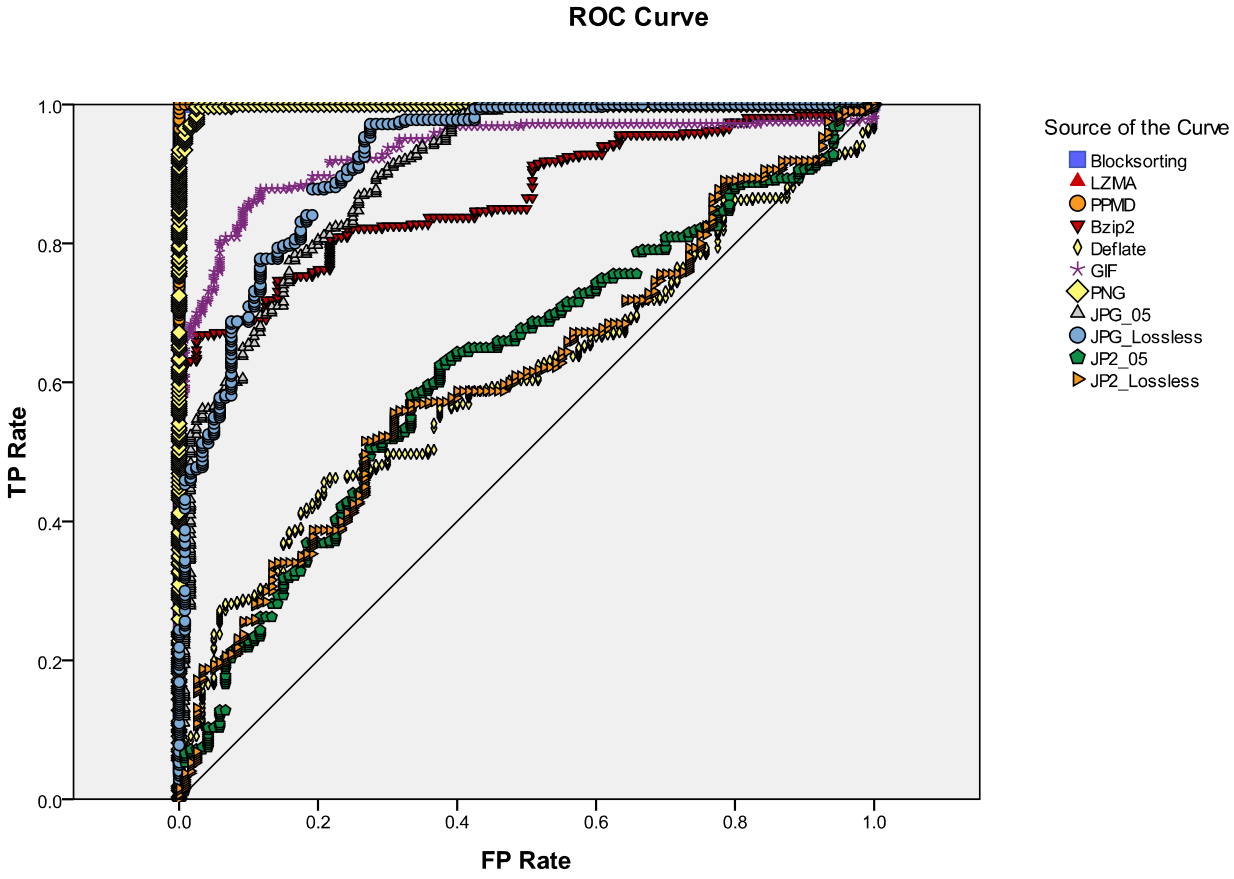

Figure 7.3 The ROC curve

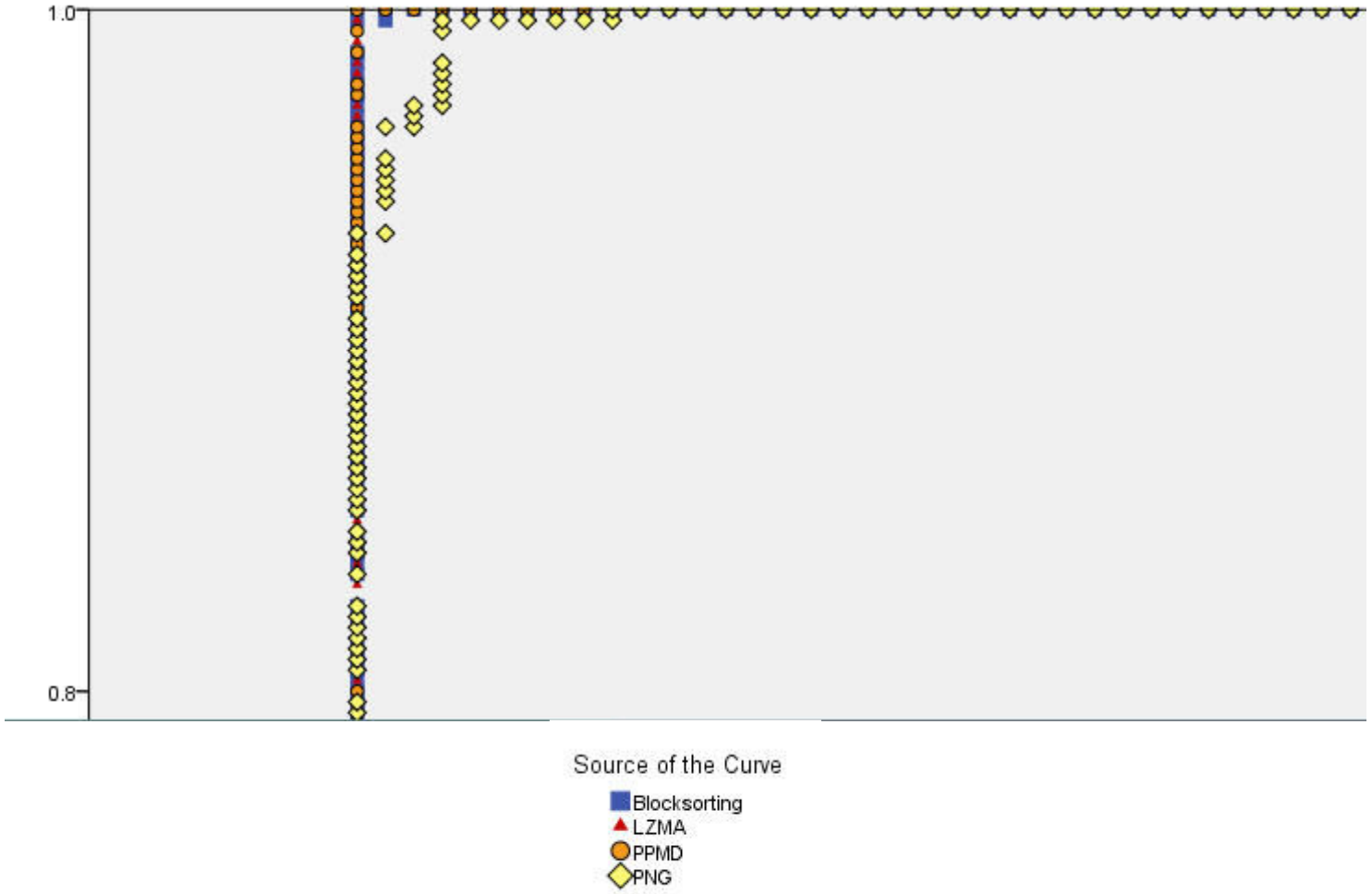

Figure 7.4 Five times larger at the corner point

7.4 Discussion

In this section, we will discuss the problems created by some particular comparison samples or web images we collected.

7.4.1 Obfuscation by advertisement banners

Some real world Phishing websites created unexpected difficulties for our identification mechanism. For example, in Figure 7.5 and Figure 7.6, (Facebook and one Phish targeting it, respectively) the two website images differ in the two advertising banners on the Phish site. Their similarity comparison result (the NCD(x,y) value calculated by LZMA compressor) is 0.986, which is far higher than other "Phishing-facebook" subjects(0.224~0.718). One could argue that this particular Phishing webpage, with an advertisement banner on top added by the free hosting website, is not so similar to the legitimate webpage anymore, and the user should have the ability to tell the difference or identify it as a Phishing or suspicious webpage. However, according to [29], 23% of users do not look at browser-based cues such as the address bar, status bar and the security indicators; and from a Gestalt-viewpoint, we would accept this result. Hence, future mechanisms need to incorporate this issue into their decision making strategy. Hopefully, this issue can be tackled by considering simply the different "sizes" of any pair of images; however, currently such an investigation is beyond the scope of this thesis.

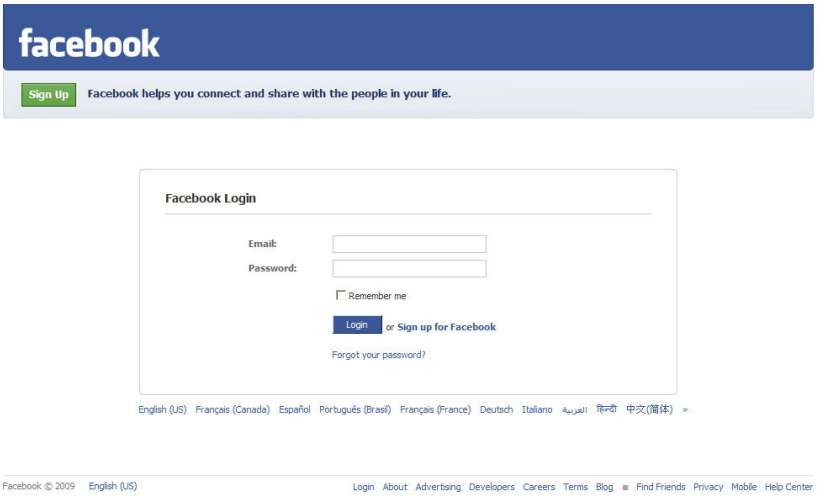

Figure 7.5 The legitimate webpage from facebook [115]

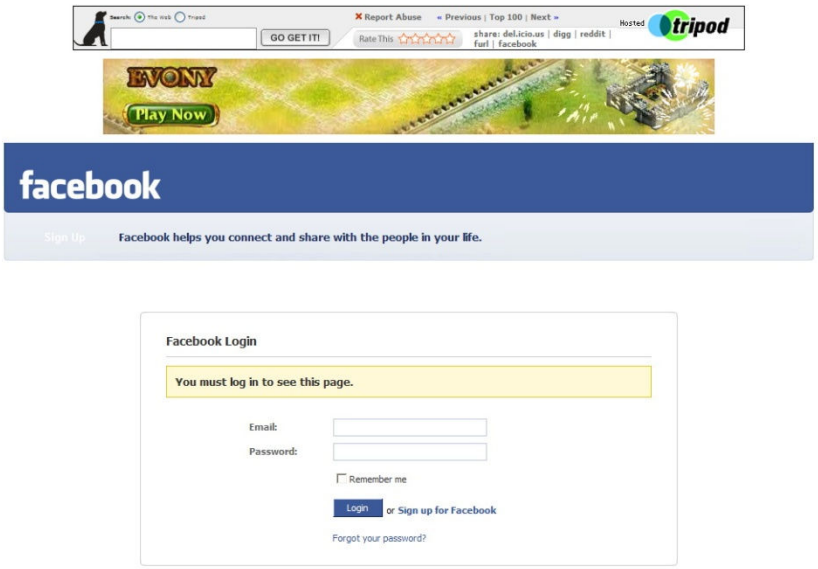

Figure 7.6 The Phishing webpage targeting facebook

7.4.2 Obfuscation by dynamic web DOM tree components

For our similarity identification mechanism, there should be only one captured image to represent our protected website. Nevertheless, our targeted websites such as Bradesco(br) and Chase we collected in Table 7.1 have dynamic web DOM tree components which generate different captured images for the same website through time. This kind of dynamic webpage components (commonly implemented via Adobe Flash [116]) are getting more and more popular for advertisements and information updates. Figure 7.7, Figure 7.8 and Figure 7.9 are captured at 1, 5 s and 10 seconds respectively after the webpage is fully loaded (the web browser will trigger the Document Complete event after the webpage completes the loading procedure). The dynamic web DOM tree components are marked with purple outlines in these figures. These dynamic components will obviously complicate our NCD comparison, by altering the information content of the page over short stretches of time.

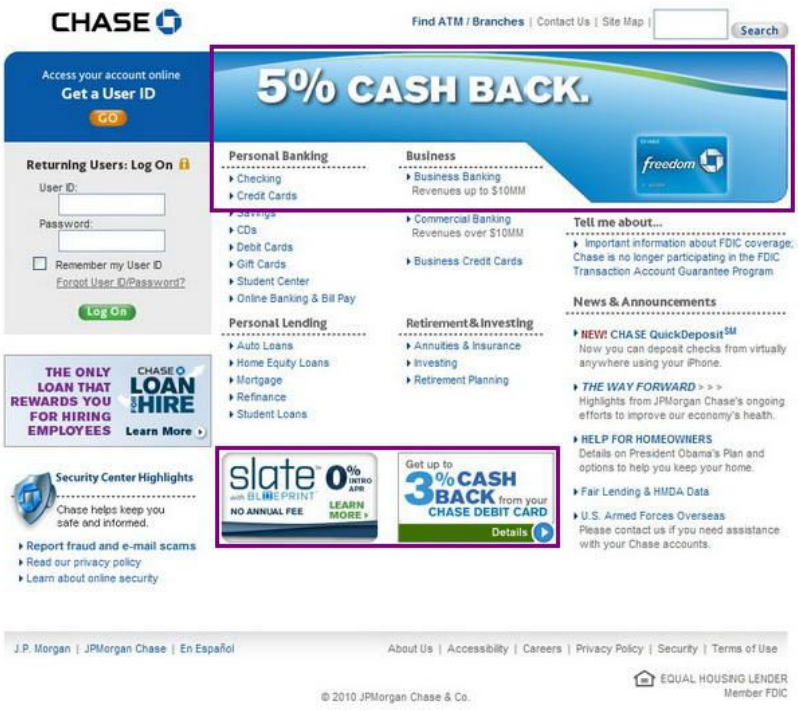

Figure 7.7 The legitimate website images with the dynamic web DOM tree components captured at 1 second

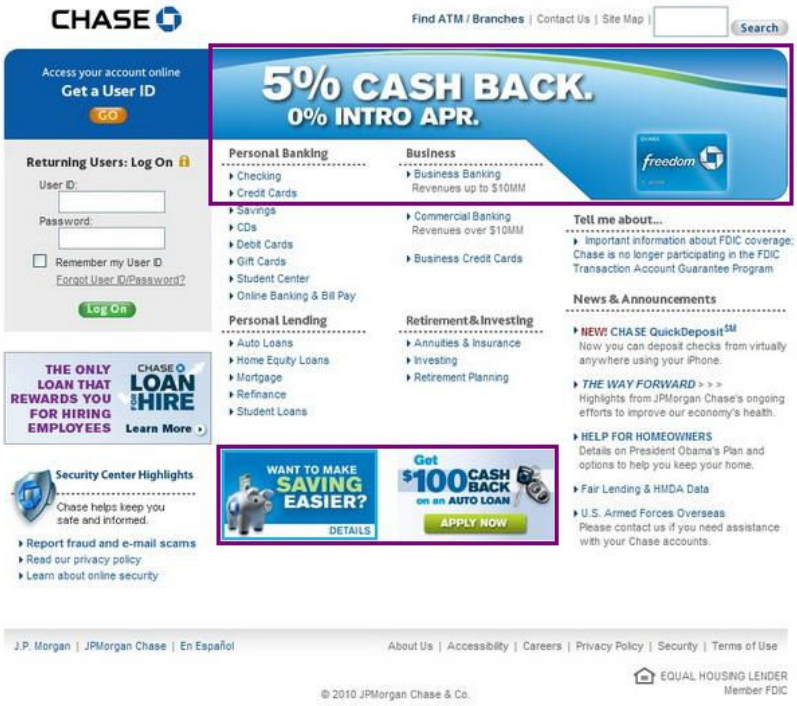

Figure 7.8 The legitimate website images with the dynamic web DOM tree components captured at 5 seconds

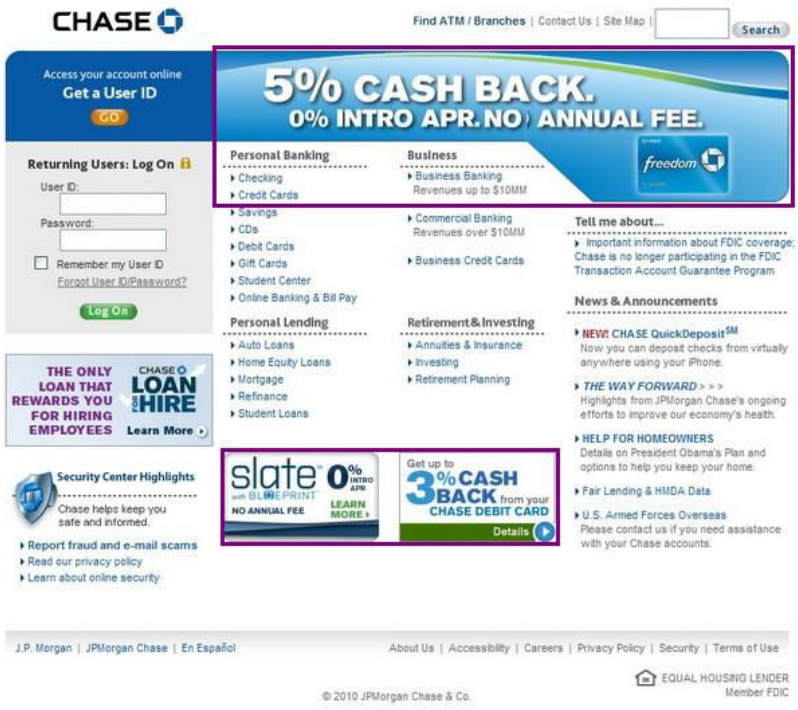

Figure 7.9 The legitimate website images with the dynamic web DOM tree components captured at 10 seconds

7.5 Conclusion and future works

The contribution from this experiment is:

1. We've found that NCD classification performance varies significantly for different compression algorithms; and
2. We also have found that LZMA compression algorithm has the best clustering experimental results specifically for our application samples-the captured webpage images.

However, the LZMA compressor is designed for data compression purposes. Hence, we should explore the possibility of customizing the compressor to improve the identification accuracy. We also need an approach that offers some help in solving the banner and the dynamic component obfuscations.

Chapter 8 The Real World Scenario

8.1 Introduction

In [29, 49] they demonstrated that neither the security indicators nor the toolbars can provide enough warnings to prevent the innocent users from being deceived by the Phishing websites. One of the reasons for this is the users tend to ignore the warning sign. Just like the famous fable "The Boy Who Cried Wolf", it is obvious that users tend to ignore warnings when the system issues too many false alerts. High false positive rate (misidentify the ordinary legitimate webpage as the Phishing one) significantly damages user trust. Consequently, human factors should be a major concern for the developing systems. In our case, how to simulate a human user's web browsing activity to challenge our mechanism's FP Rate is a main concern. It is hard to predict what kind of web site users are going to browse. Therefore, it is crucial to use statistical data to assist us to simulate the human web browsing activities for examining the viability of our proposed system.

We choose to use website popularity as a proxy for "common browsing behavior." Alexa.com [10] is a well-known source of website popularity ratings. By selecting the top 110 popular real-world websites from the Alexa.com rankings, we hope to simulate user's possible browsing activity to examine the robustness of our anti-Phishing mechanism. We will simulate a realistic scenario where the website is visited, and a screenshot of the rendered webpage is captured. This is then passed to our anti-Phishing mechanism. This will provide a realistic test of our proposed system.

8.2 The Empirical evaluation

The objective of this evaluation is to feed the most popular websites to our NCD white-list based classification system to examine the discriminative performance of the system.

8.2.1 Design

We designed the experiment with the 16 legitimate websites we used in Table 7.1 Group one as our protected websites (the white-list). Then , as shown in Table 8.1, 110 popular/top ranking websites were selected from Alexa top global websites list [117] on 04/19/2010 as the web sites fed to the NCD classification system. The LZMA compression algorithm will be used for this experiment in that it has the better classification ability according to the result from our former test in Section 7.3. Because each protected website will be pair-wise compared against 110 Alexa websites, there will be 16*110=1760 NCD values generated. Then these calculated values are collected to form the “LA population”.

Table 8.1 Alexa top 110 global websites

Number	URL	Number	URL
Alexa-001	http://www.google.com	Alexa-031	http://www.imageshack.us
Alexa-002	http://www.blogger.com	Alexa-032	http://www.livejournal.com
Alexa-003	http://www.baidu.com	Alexa-033	http://www.megaupload.com
Alexa-004	http://www.twitter.com	Alexa-034	http://www.kaixin001.com
Alexa-005	http://www.wordpress.com	Alexa-035	http://www.filesress.com
Alexa-006	http://www.myspace.com	Alexa-036	http://www.hotfile.com
Alexa-007	http://www.microsoft.com	Alexa-037	http://www.cpxinteractive.com
Alexa-008	http://www.bing.com	Alexa-038	http://www.renren.com
Alexa-009	http://www.yandex.ru	Alexa-039	http://www.mixi.jp
Alexa-010	http://www.linkedin.com	Alexa-040	http://www.hao123.com
Alexa-011	http://www.flickr.com	Alexa-041	http://www.sogou.com
Alexa-012	http://www.mail.ru	Alexa-042	http://www.thepiratebay.org
Alexa-013	http://www.fc2.com	Alexa-043	http://www.odnoklassniki.ru
Alexa-014	http://www.rapidshare.com	Alexa-044	http://www.xtendmedia.com
Alexa-015	http://www.craigslist.org	Alexa-045	http://www.optmd.com
Alexa-016	http://www.livejasmin.com	Alexa-046	http://www.adultfriend.com
Alexa-017	http://www.vkontakte.ru	Alexa-047	http://www.rediff.com
Alexa-018	http://www.soso.com	Alexa-048	http://www.clicksor.com
Alexa-019	http://www.mozilla.com	Alexa-049	http://www.partypoker.com
Alexa-020	http://www.doubleclick.com	Alexa-050	http://www.ning.com
Alexa-021	http://www.apple.com	Alexa-051	http://www.mywebsearch.com
Alexa-022	http://www.orkut.com.br	Alexa-052	http://www.badoo.com
Alexa-023	http://www.ask.com	Alexa-053	http://www.cnzz.com
Alexa-024	http://www.adobe.com	Alexa-054	http://www.filestube.com
Alexa-025	http://www.youporn.com	Alexa-055	http://www.download.com
Alexa-026	http://www.mediafire.com	Alexa-056	http://www.zedo.com
Alexa-027	http://www.about.com	Alexa-057	http://www.twitpic.com
Alexa-028	http://www.cnet.com	Alexa-058	http://www.netflix.com
Alexa-029	http://www.hi5.com	Alexa-059	http://www.depositfiles.com
Alexa-030	http://www.4shared.com	Alexa-060	http://www.nasza-klasa.pl

Number	URL	Number	URL
Alexa-061	http://www.bit.ly	Alexa-091	http://www.ameba.jp
Alexa-062	http://www.tinypic.com	Alexa-092	http://www.xing.com
Alexa-063	http://www.tagged.com	Alexa-093	http://www.118114.cn
Alexa-064	http://www.free.fr	Alexa-094	http://www.naver.com
Alexa-065	http://www.tumblr.com	Alexa-095	http://www.gamespot.com
Alexa-066	http://www.zynga.com	Alexa-096	http://www.51.la
Alexa-067	http://www.sourceforge.net	Alexa-097	http://www.leboncoin.fr
Alexa-068	http://www.wikimedia.org	Alexa-098	http://www.narod.ru
Alexa-069	http://www.fastbrower.com	Alexa-099	http://www.youdao.com
Alexa-070	http://www.domainols.com	Alexa-100	http://www.scribd.com
Alexa-071	http://www.vk.com	Alexa-101	http://www.avg.com
Alexa-072	http://www.typepad.com	Alexa-102	http://www.megaporn.com
Alexa-073	http://www.torrentz.com	Alexa-103	http://www.z5x.net
Alexa-074	http://www.reference.com	Alexa-104	http://www.skype.com
Alexa-075	http://www.dell.com	Alexa-105	http://www.formspring.me
Alexa-076	http://www.archive.org	Alexa-106	http://www.ucoz.ru
Alexa-077	http://www.getpersos.com	Alexa-107	http://www.leo.org
Alexa-078	http://www.2ch.net	Alexa-108	http://www.liveinternet.ru
Alexa-079	http://www.blogfa.com	Alexa-109	http://www.58.com
Alexa-080	http://www.digitalpoint.com	Alexa-110	http://www.clickbank.com
Alexa-081	http://www.stumbleupon.com		
Alexa-082	http://www.hp.com		
Alexa-083	http://www.linkbucks.com		
Alexa-084	http://www.libero.it		
Alexa-085	http://www.angege.com		
Alexa-086	http://www.alipay.com		
Alexa-087	http://www.gougou.com		
Alexa-088	http://www.126.com		
Alexa-089	http://www.istockphoto.com		
Alexa-090	http://www.imagevenue.com		

8.2.2 Methodology

To identify our LA population feeds are a Phish or Legitimate website population; we need another tested population as a reference group. We selected the LP population which we generated in Section 7.2.4. Then we use the same statistical methods (*Z*-test, effect size, and ROC curves) to examine/evaluate the difference between these two populations. The Z-test examines if there is a statistically significant difference for these

two populations. The effect size will quantify any difference found between our two populations. The ROC curve compares the two populations again using a decision-threshold classifier.

8.2.3 Interpretation and the evaluation result

8.2.3.1 Null Hypothesis

From the result given in the Table 8.2, we can reject every null hypothesis with $P<0.05$. Consequently, the hypothesis we made- that NCD values in the Alexa population (LA population) are significantly different from the Phishing population (LP population) - is supported.

Table 8.2 The Z-test result

Z-test Compression Algorithm	*z* statistic	***P*** Significance
LZMA	10.420	<0.000001

8.2.3.2 Magnitude of Effect

The greater value of the effect size, the better ability of the similarity classification will be. From the result given in the Table 8.3, we have found the effect size is very big.

Table 8.3 The result of effect size

Compression Algorithm	LP Population Means	Num	SD (Standard Deviation)	LA Population Means	Num	SD (Standard Deviation)	SD (pooled)	Effect size
LZMA	0.4131593	320	0.3159146	0.995077	1760	0.005837	0.123894	4.696893558

8.2.3.3 ROC Curve

From the result given in the Table 8.4, we have found the TPR is high and FPR is very low. From the result given in the Table 8.5, we have found the very high AUC value. From the ROC curve plot shown in Figure 8.1 and Figure 8.2, we have found the graph shown the identification performance is great.

Table 8.4 TPR FPR at the corner point of the ROC curve

	NCD
TPR	**99.99% >>**
FPR	**<< 0.01%**

Table 8.5 AUC result

	NCD
AUC	1.000

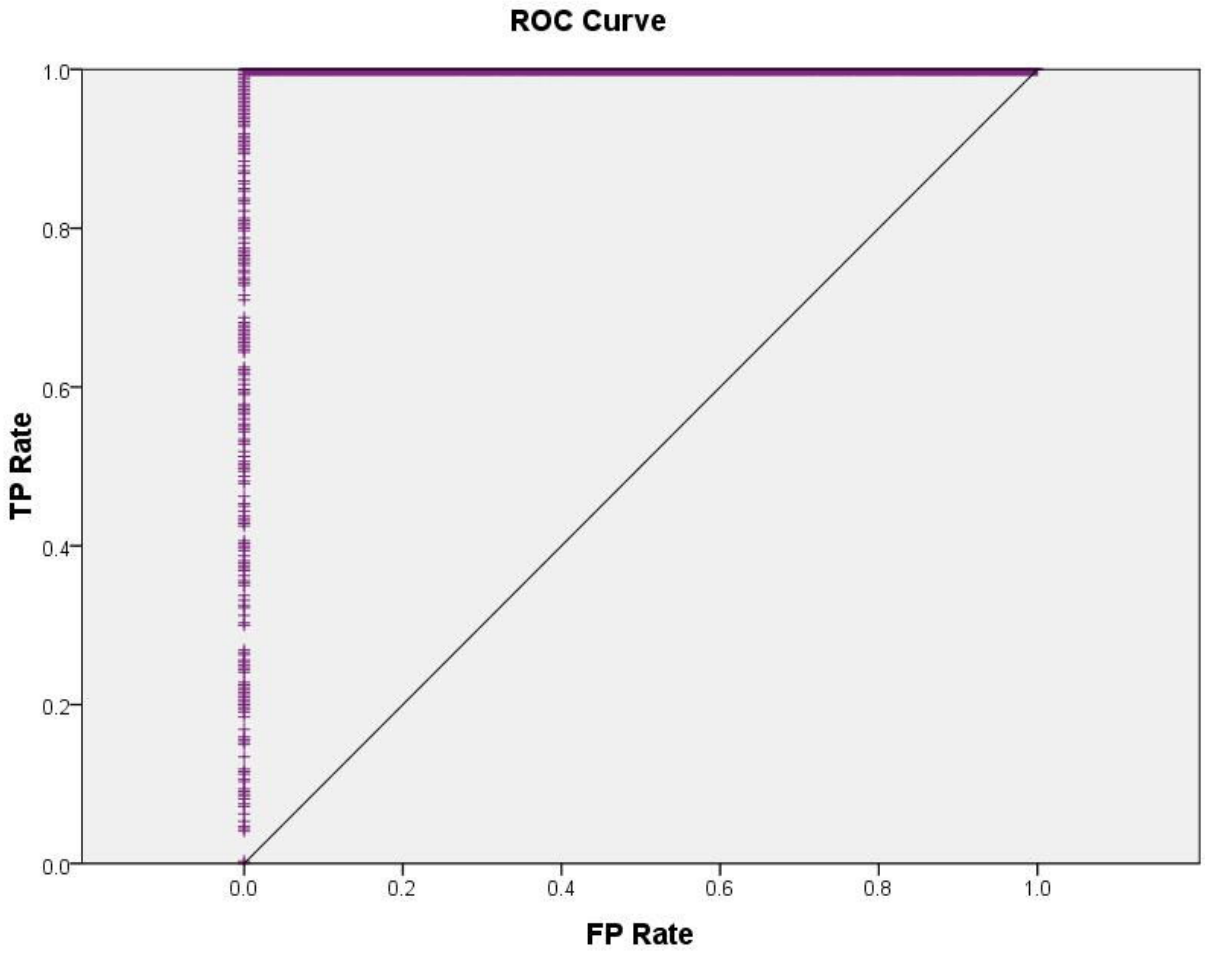

Figure 8.1 The ROC curve

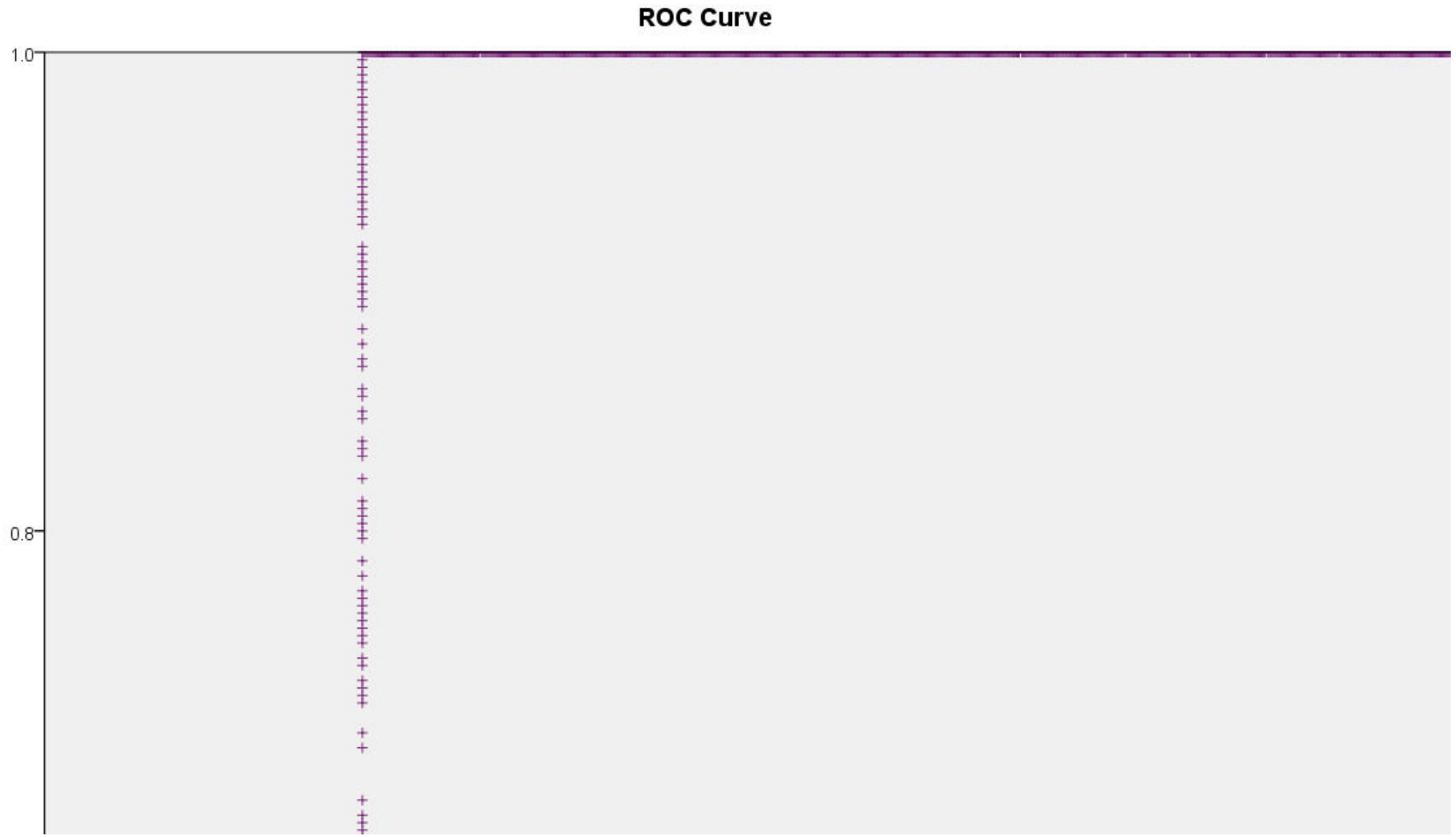

Figure 8.2 Five times larger at the corner point

8.3 A Phishing Classifier Based on NCD

We believe that an actual anti-Phishing mechanism will not consist merely of a decision-threshold classifier using only NCD values. Instead, we expect that some classification algorithm will be employed, which likely fuses the NCD feature with other useful information, forming a complete and robust anti-Phishing mechanism. In this section we investigate how such a system would work, using only the observed NCD values for websites. As we have not previously examined the performance of our system using a more complex classifier, we will examine both the LL vs. LP dataset of Section 7.2.4, and the LA vs. LP dataset of Section 8.2. For brevity, these are denoted as the "440" and "430" datasets in the following sections (based on the number of websites in each).

8.3.1 Design

Our experimental design simulates a browser help object / extension that implement our anti-Phishing mechanism. We assume that we have been provided with a whitelist of protected sites (this will be the same 16 protected sites as in Section 8.2). We assume that the browser helper object uses the NCD technique to compare each visited website against the sites in the whitelist. The minimum NCD value for the visited site against the whitelist is determined, and passed as the NCD feature value to a classifier. (NCD values must be computed for every whitelist site when a new website is visited. Our classifier only accepts the minimum NCD value, as this is the whitelist site most likely to be seen as similar to the website under examination.) If the classifier predicts that this site is a Phish, we raise an alert.

8.3.2 Methodology

The features employed in our experiments were the NCD values for both orderings of the images in the concatenation (i.e. AB and BA), as well as the average of these two. For the Alexa dataset, we compare the website against only the whitelist site having the lowest NCD (AB) value. The learning algorithms tested were the C4.5 decision tree generator (J48 implementation in WEKA [118]); the Ripper rule-induction algorithm (JRip implementation in WEKA); a Logistic regression classifier; and support vector machines (trained using WEKA's SMO implementation). In our experiments, we perform a tenfold cross-validation for each parameterization of the learning algorithms (changing the random seed each time), and obtain the average and sample standard deviation of the TPR and FPR.

8.3.3 Interpretation of results

ROC curves were generated on both the cross-comparison of whitelist sites (440 data points) and the top 110 most-popular websites as ranked by Alexa.com (430 data points). The means of TPR and FPR were plotted following the method in [119]; note that an alternative method of aggregating ROC curves for a cross-validation was proposed in [120],[121]. Parameterizations for the algorithms were varied to produce different (FPR, TPR) pairs as suggested in [120]; this corresponds to the experimental setting where FPR is uncontrollable by the analyst. Classifier parameters varied were as follows:

- C4.5: we vary the minimum number of examples *m* and the confidence *c* per node; these control the pruning operation in C4.5.

- Ripper: we vary the minimum total instance weight per rule *n* and the number of optimizations *o*.
- Logistic: we vary the ridge parameter *r*.
- SMO: We choose the radial basis function kernel, and vary its width *g*. The complexity parameter *C* is held constant at 1 and all other parameters are left at default values.

All classifiers performed extremely well on the two datasets; to the point where producing error bars to represent the standard deviations of TPR and FPR in the ROC curves is visually futile. This also means that the ROC curves would overlap heavily (almost completely) if plotted on the same graph. We therefore plot the ROC curve for each classifier individually. Figure 8.3, Figure 8.4, Figure 8.5 and Figure 8.6 present the ROC curves for the 440 (LL-vs.-LP) dataset, while Figure 8.7, Figure 8.8, Figure 8.9 and Figure 8.10 and present the ROC curves for the 430 (LA-vs.LP) dataset.

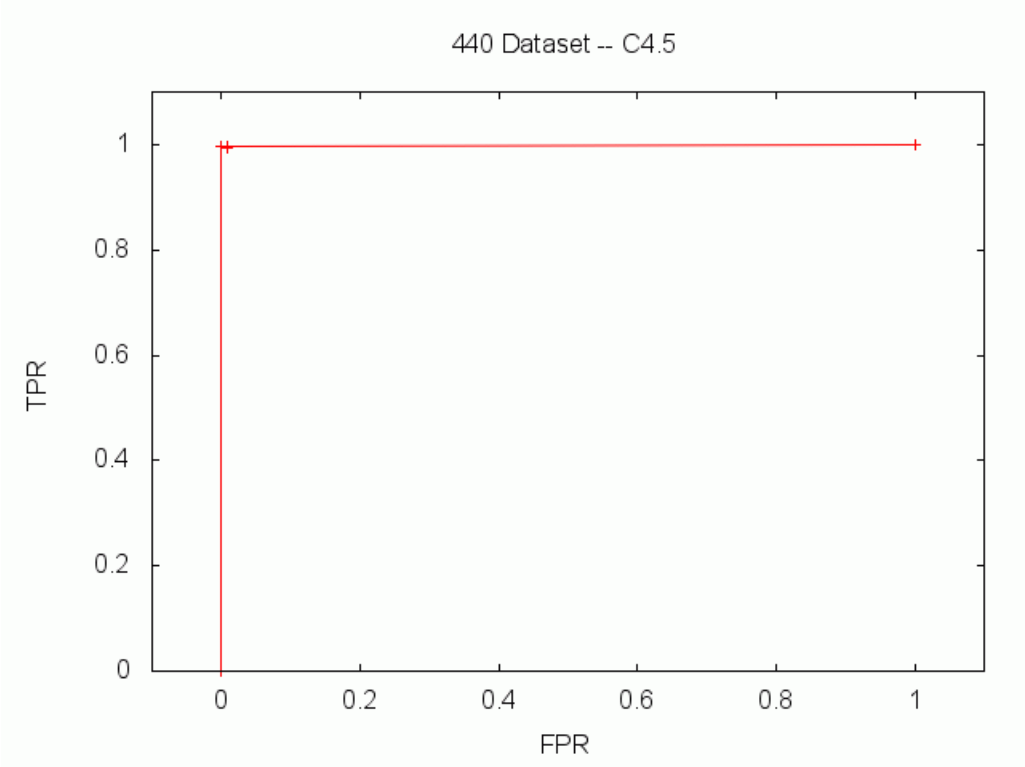

Figure 8.3 C4.5 Classifier, LL vs. LP data

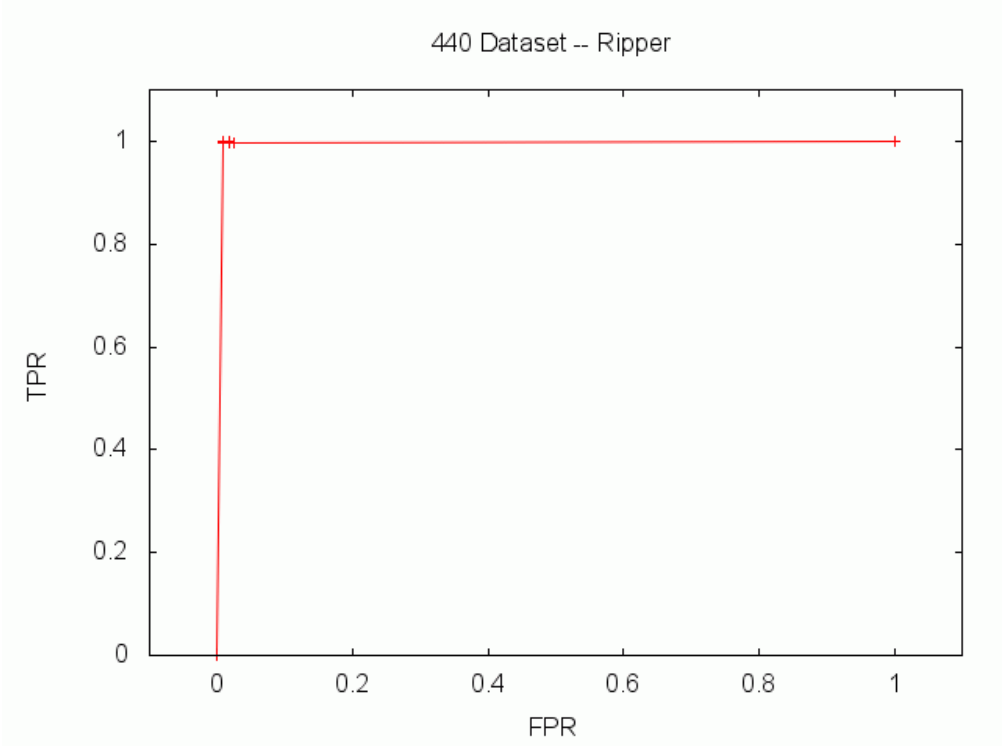

Figure 8.4 Ripper Classifier, LL vs. LP data

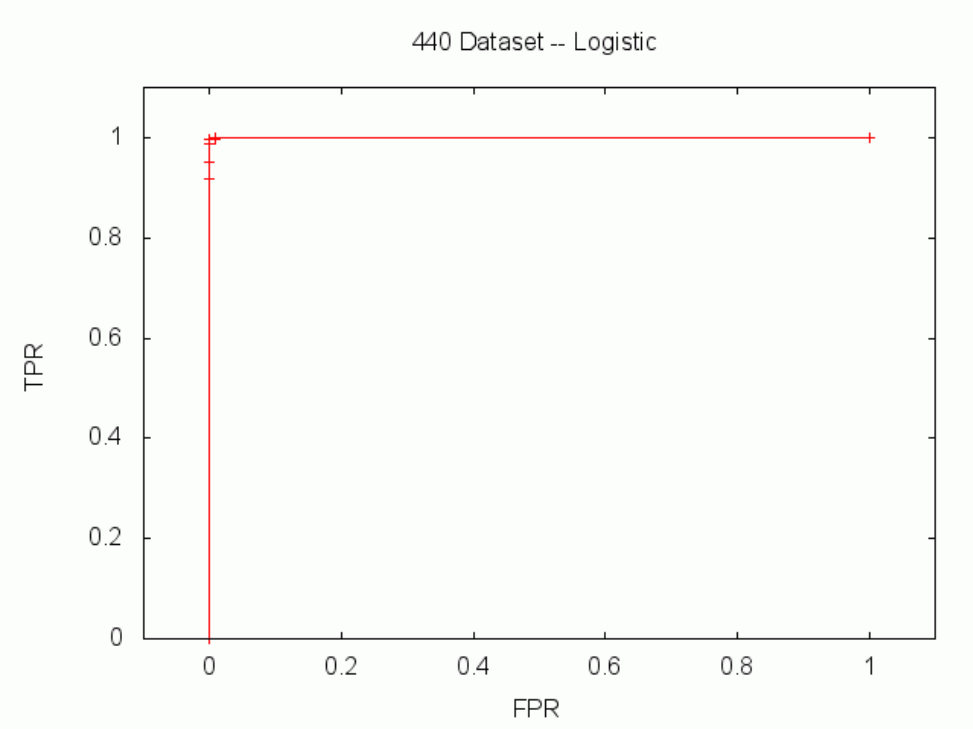

Figure 8.5 Logistic Classifier, LL vs. LP data

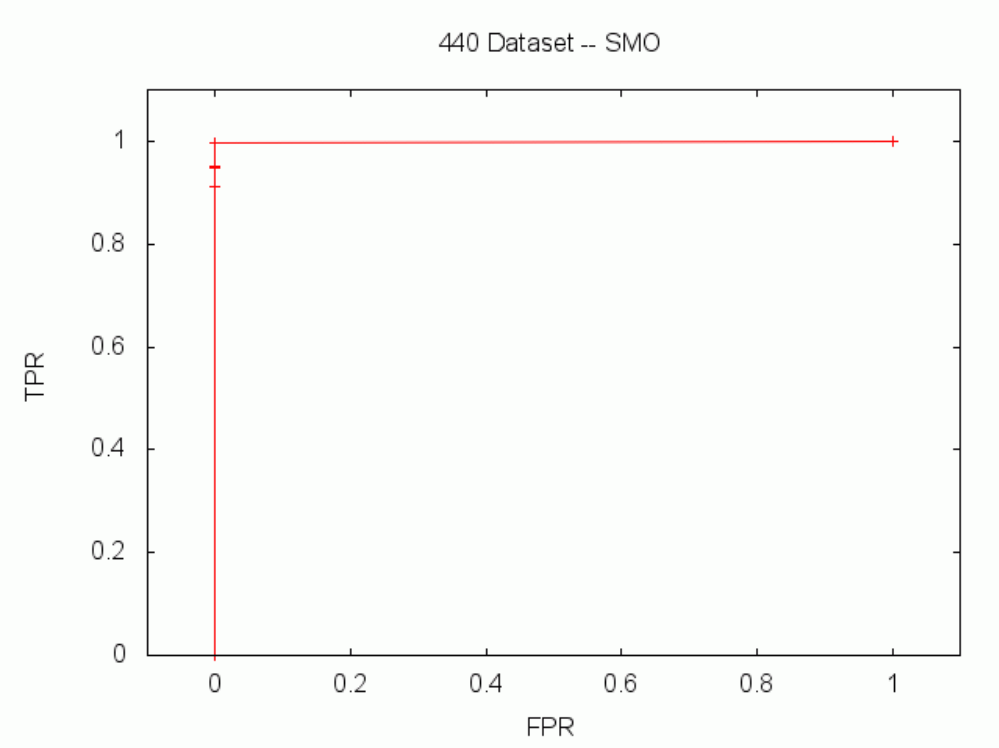

Figure 8.6 SMO classifier, LL vs. LP data

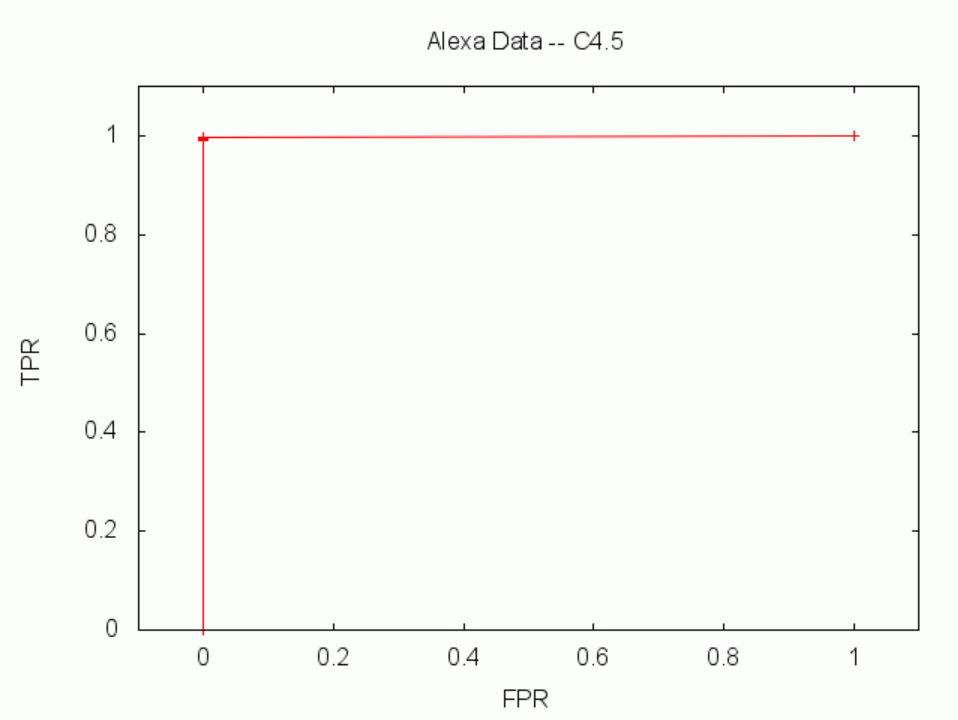

Figure 8.7 C4.5 Classifier, Alexa (LA vs. LP) data

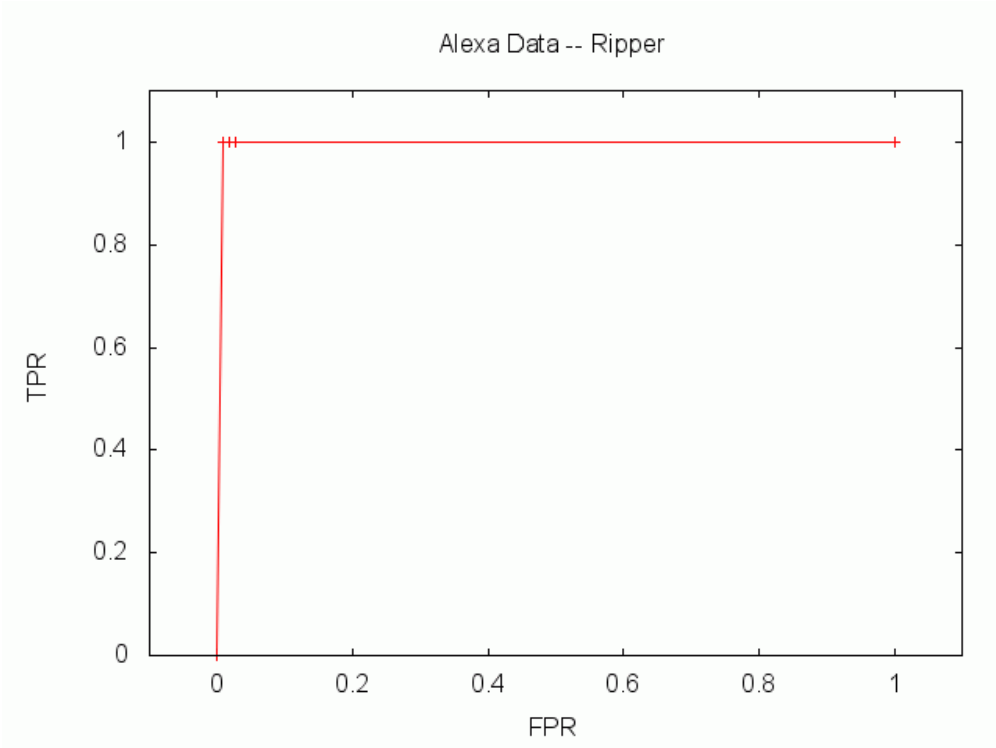

Figure 8.8 Ripper Classifier, Alexa (LA vs. LP) data

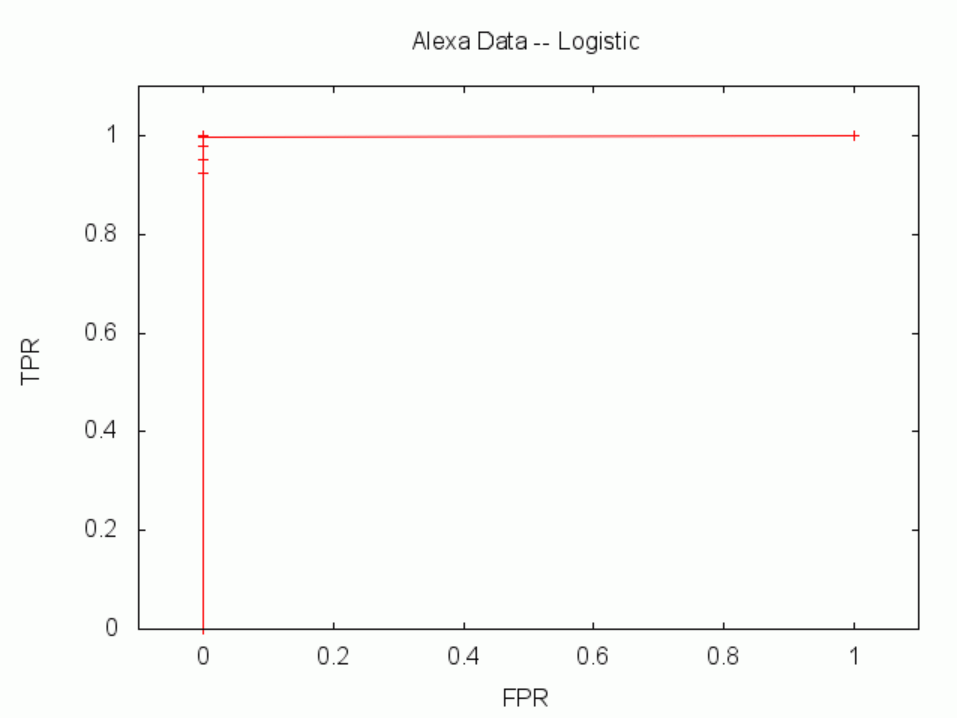

Figure 8.9 Logistic Classifier, Alexa (LA vs. LP) data

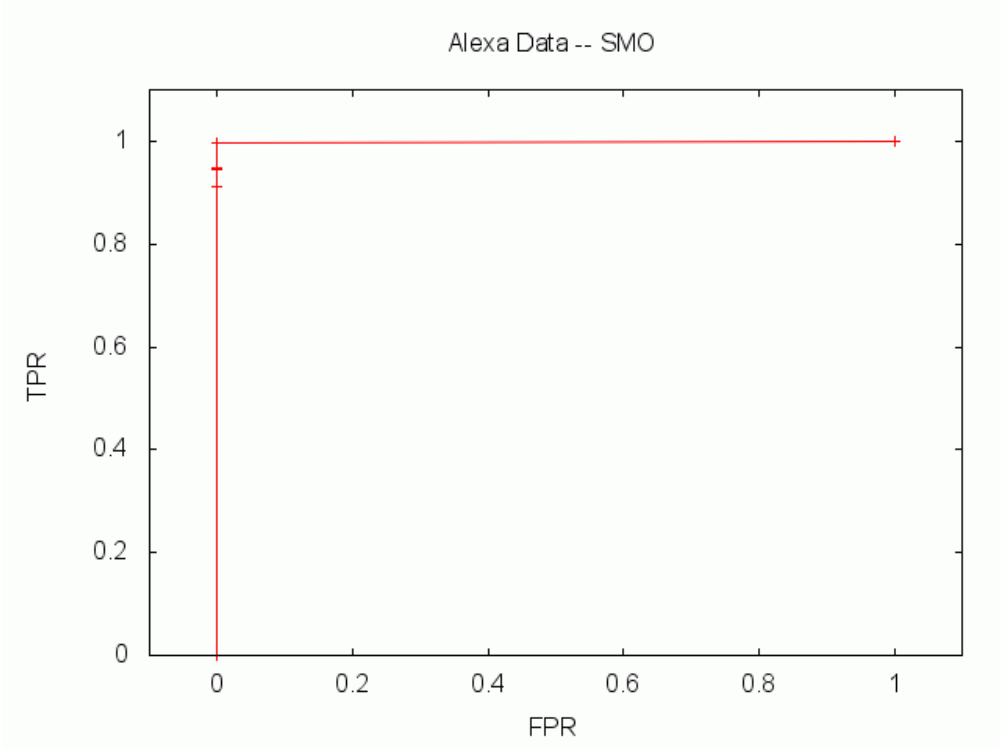

Figure 8.10 SMO Classifier, Alexa (LA vs. LP) data

As the figures above do not show the standard deviation for TPR and FPR, we provide them in the following tables. Table 8.6, Table 8.7, Table 8.8 and Table 8.9 are the average and sample standard deviations of FPR and TPR for each parameterization of an algorithm for the whitelist cross-comparison (LL vs. LP) dataset, and Table 8.10, Table 8.11, Table 8.12 and Table 8.13 are the same information for the Alexa (LA vs. LP) dataset.

Table 8.6 TPR, FPR for C4.5 Parameterizations, LL vs. LP data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996528	0.010417	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.99375	0.019764	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352

0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.99375	0.019764	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352

Table 8.7 TPR, FPR for Ripper Parameterizations, LL vs. LP data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.996875	0.009882	0.008333	0.026352
1	0	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
1	0	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
1	0	0.008333	0.026352
1	0	0.008333	0.026352

0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
1	0	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
1	0	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.016667	0.052705
0.996875	0.009882	0.016667	0.052705
1	0	0.016667	0.035136
0.996875	0.009882	0.016667	0.052705
1	0	0.016667	0.035136
1	0	0.016667	0.035136
1	0	0.016667	0.035136
0.996875	0.009882	0.025	0.056246

Table 8.8 TPR, FPR for Logistic Parameterizations, LL vs. LP data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.9875	0.03019	0	0
0.91875	0.044683	0	0
0.95	0.03019	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352
0.996875	0.009882	0.008333	0.026352

1	0	0.008333	0.026352
0.379365	0.48332	0.333333	0.48795
0.411111	0.499471	0.377778	0.485777
0.455556	0.505787	0.388889	0.494681
0.468889	0.49545	0.457143	0.507093
0.333333	0.48795	0.457778	0.488514
0.390476	0.496244	0.522222	0.507353
0.468254	0.498107	0.52381	0.508432
1	0	1	0

Table 8.9 TPR, FPR for SMO Parameterizations, LL vs. LP data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.9125	0.038415	0	0
0.946875	0.039115	0	0
0.95	0.026352	0	0
0.996875	0.009882	0	0
0.95	0.039528	0	0
0.996875	0.009882	0	0
1	0	1	0
1	0	1	0
1	0	1	0

Table 8.10 TPR, FPR for C4.5 Parameterizations, Alexa data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.996875	0.009882	0	0
0.99375	0.019764	0	0
0.99375	0.019764	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0

0.990625	0.029646	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.99375	0.019764	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.99375	0.019764	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.99375	0.019764	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0

Table 8.11 TPR, FPR for Ripper Parameterizations, Alexa data

AvgTPR	StdTPR	AvgFPR	StdFPR
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.009091	0.028748
1	0	0.018182	0.057496
1	0	0.018182	0.057496
1	0	0.018182	0.057496
1	0	0.027273	0.086244

Table 8.12 TPR, FPR for Logistic Parameterizations, Alexa data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.996875	0.009882	0	0
0.978125	0.025727	0	0
0.95	0.033593	0	0
0.925	0.078229	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
1	0	0	0
1	0	0	0
0.996875	0.009882	0	0

Table 8.13 TPR, FPR for SMO Parameterizations, Alexa data

AvgTPR	StdTPR	AvgFPR	StdFPR
0.9125	0.062152	0	0
0.94375	0.032275	0	0
0.946875	0.029646	0	0
0.996875	0.009882	0	0
0.996875	0.009882	0	0
1	0	1	0
1	0	1	0
1	0	1	0

8.4 Discussion

The main contribution from these experiments is that we have demonstrated the effectiveness and robustness of our NCD based identification system using a realistic set of popular websites. The low observed false positives rate (~0.8%) shows our mechanism will hopefully be trusted by users. We have demonstrated the discriminative power of our method using statistical tests, and confirmed those results using a variety of classification algorithms. Interestingly, the relatively simple Logistic regression classifier seems to

(very slightly) outperform decision trees and support vector machines. In the future, we hope to employ distributed computer system to challenge the robustness of our identification mechanism against a very large corpus of sites (Alexa.com [10] provides a list of the top 1,000,000 sites, for instance).

Chapter 9 Refinement for LZMA Compression Algorithm

9.1 Introduction

In the previous chapters, we found that LZMA is the best performing compression algorithm when clustering visually similar webpage images. However, is there any room for improvement? Can we modify the LZMA algorithm for our specific application purposes? The answer to this question must start from understanding how LZMA works.

9.2 Regular LZMA compression algorithm process in details

Basically, the LZMA algorithm has two major steps. The first part is the attempt to find matching symbols. This is done by the modified LZ77 sliding window technique – to be referred to as the LZ (Lempel–Ziv) part. The second part is the encoding of the output from the first part to make the output stream much smaller. The encoding operation is mainly done based on a Markov-chain [122] process which means the next output depends only on the current output. The probability of its occurrence is used to connect the relationship between these two outputs. An adaptive binary based range coding machine will be used during this process – to be referred to as the MA (Markov-chain Algorithm) part.

9.2.1 The LZ part

The LZ part starts from the modified LZ77 [91] searching method we have already described in Section 6.2.4. However, the author of LZMA has found the output format for LZ77 has a serious problem of wasting number space for unmatched output.

9.2.1.1 The redundancy of LZ77 mismatch format

When LZ77 cannot find a match in the dictionary, it generates the mismatch output in the format of (0, 0, next symbol). When lots of mismatch results are found, as shown in Figure 9.1, the first two zero digits (0, 0) have significant redundancies.

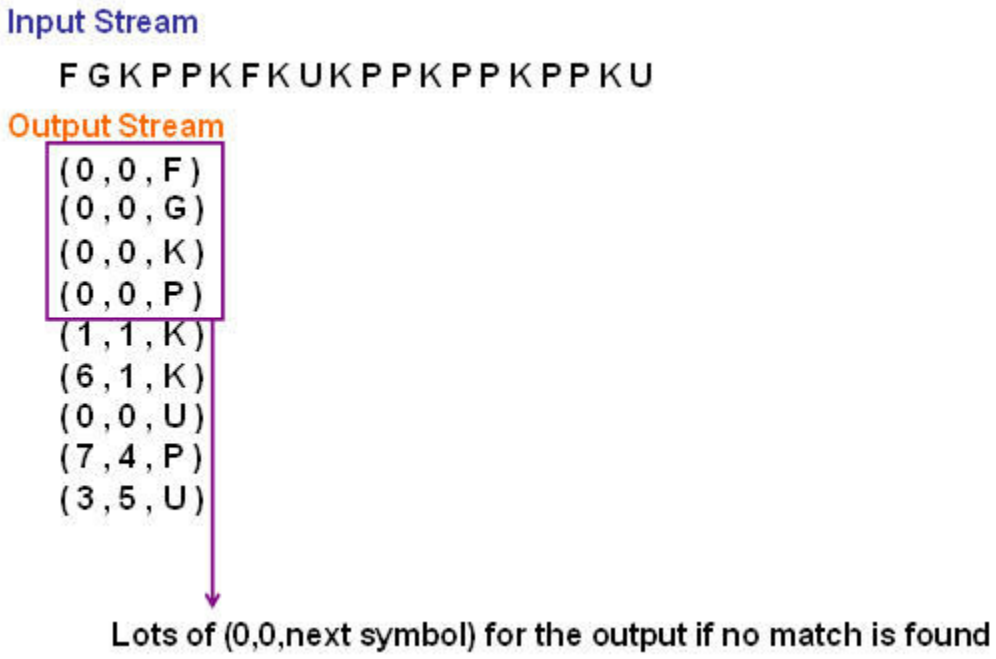

Figure 9.1 The redundancy of (0, 0) for LZ77

9.2.1.2 The LZ output format for LZMA

To remove this redundancy, LZMA changes its output format into a more concise one to minimize the size of the output stream. For the mismatch part, LZMA only places an unmatched character into the output which takes one character space instead of three for LZ77.

As shown in Figure 9.2, the searching method is the same as LZ77, which separates its sliding window into two parts, the search buffer (the dictionary) and the look-ahead buffer.

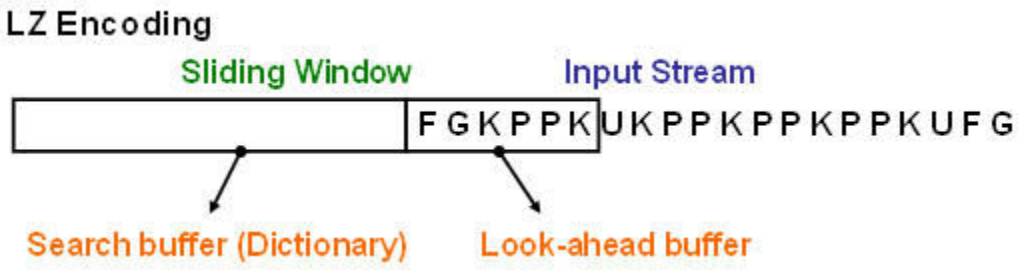

Figure 9.2 The LZ Encoding

There are three kinds of token after matching; these outputs will be sent to the adaptive range encoder (the MA part) separately. The first token is the literal (a value in the interval 0~255 with the Unicode format) if nothing matches the look-ahead buffer. The second token is the (length, offset) pair if a match is found. As with LZ77, length is the longest matching substring starting from the offset position in the search buffer. The third token is the pair of (length, index). As the large size of the search buffer causes large offset values, a four-entry array which contains the information of the most four recent distances that have been found is kept. In many cases, this offset history array can reduce the number of offset or distance dramatically. If the distance of the current match equals one of the four distances in this array, a short two-bit index will be written instead of the actual large distance. This offset history array design is especially effective in reducing the storage size for the consecutive repeat input characters or input stream. These three kinds of output tokens can be seen in Figure 9.3.

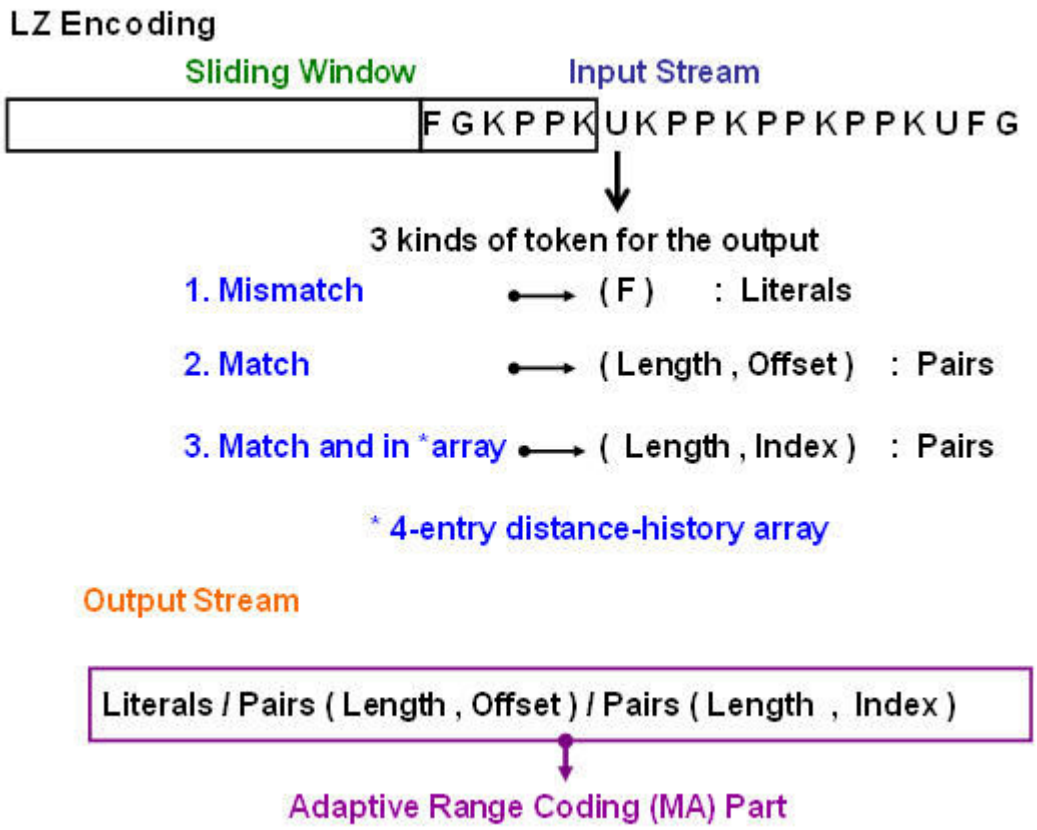

Figure 9.3 The MA part

9.2.2 The MA part

The MA part is called adaptive binary range coding. It is similar to Context-based Adaptive Binary Arithmetic Coding or CABAC in [123]. The difference between arithmetic coding and range coding is the representation of the output. Arithmetic Coding uses the probability which is less than or equal to one during the process. The range coding uses a fixed range instead of the probability for encoding [94]. The three kinds of tokens after the searching process from the LZ part will be sent to range encoder separately. The literals will be transformed into binary before the encoding process. The

other two kinds of pair token outputs (length, offset) and (length, index) are sent to the encoder directly because their output formats are already in the binary form.

Range coding is based on Markov chains which use a large integer range and a frequency or probability estimation for the character to encode all the input characters of the message into one single number. The frequency algorithm will be used to reduce the current range integer down to another smaller range which corresponds to the next character to be encoded. Then the next character will be encoded by the same procedure. For decoding, the decoder must acquire the same frequency algorithm in advance as the encoder used. By using the same probability estimation, the decoder can recover the original message in the reverse way.

Here is a simplified example to show how the range coding works. Assume we have a 4-character input stream "PPQA" for our example demonstration. The initial range is [0, 10000) and assume the initial frequency algorithm is P: .50; Q: .30; A: .20. Therefore, the first three sub ranges are divided as:

P: [0, 5000), Q: [5000, 8000), A: [8000, 10000).

Our first input character is P, so we can reduce our initial range to [0, 5000). This makes the second three sub ranges:
PP: [0, 2500), PQ: [2500, 4000), PA: [4000, 5000).

The second input P reduces the range to [0, 2500). The third three sub ranges are

PPP: [0, 1250), PPQ: [1250, 2000), PPA: [2000, 2500)

The third input Q reduces our range to [1250, 2000), and so The fourth three sub ranges are:

PPQP: [1250, 1625), PPQQ: [1625, 1850), PPQA: [1850, 2000)

The range coding output can be any number between 1850~1999 which also provides some degree of error tolerance. In a normal situation, we will use the integer in the midpoint; or1925 in this situation. The decoding procedure is to perform the encoding process in the reverse way. The visual representation can be seen as follows in Figure 9.4.

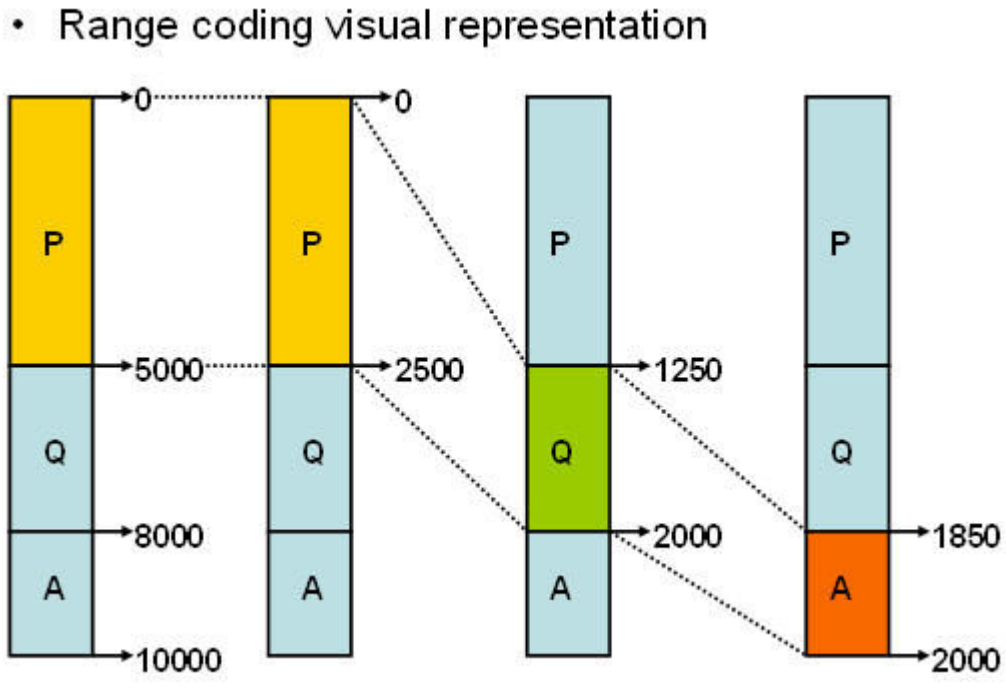

Figure 9.4 The range encoding

The concept of the adaptive range coding was introduced in [93]. The frequency algorithm or the probability is fixed at the beginning of the encoding process, then changes depending on the occurrence of that specific character in the input stream. This means the region sizes are continually adjusted depending upon the input during the operation and adapts itself directly to the system and the environment.

Figure 9.5 provides a visualization of the complete LZMA compression process.

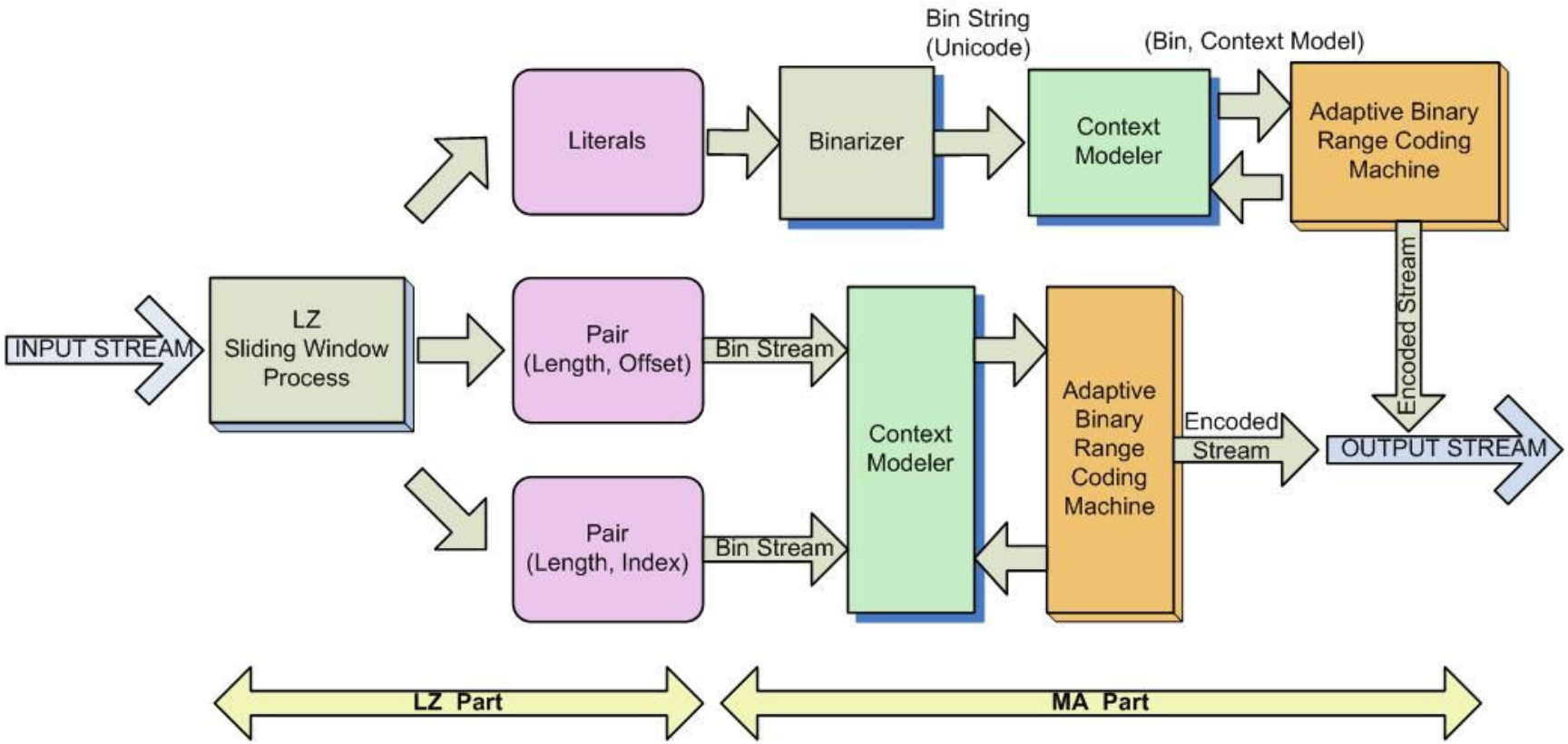

Figure 9.5 The Regular LZMA Compression Diagram

9.3 The Refined LZMA Compression Algorithm

After understanding the LZMA compression algorithm process in detail, we started to consider the possibility to modify the algorithm to our specific needs-clustering by similarity based on the NCD metric.

9.3.1 The Concept

The concept basically comes from the idea of lossy image compressors which lose some part of the image that is "redundant" for human perception. Recall that this is possible due to the division of the image into two specific parts which are the luminance intensity *Y*(highly obvious to the human) and presence of blue *Cb* and red *Cr* (not so obvious to the human) based on the physiology of the eye. The lossy compressor separates the sensitive part and insensitive part for advanced compression process. Thus, is it possible to separate our object into two specific parts - the similar and dissimilar part - to improve our similarity detection still further? By analyzing the characteristic of our data compression algorithms, we might have the chance to find the answer.

9.3.2 The meaning of literals and pairs

The dictionary compression algorithm is specialized in searching and categorizing matched and unmatched terms from its dictionary. If we define or set the dictionary size as the same of the object size, then the dictionary can be deemed as the whole object itself. Therefore, the match or mismatch in the dictionary can be seen in another form: the similar and dissimilar parts of the object. Consequently, if we apply the dictionary compression searching method to the object, by setting the dictionary size as the same or

larger than the object, it is definitely possible to separate the similar and dissimilar part in the form of match or mismatch.

The literals and pairs of the dictionary from the LZ part of LZMA mean dissimilar and similar parts of the object when the dictionary size is set to the object file size. As we mentioned in the former section, when LZMA algorithm can not find the match in the dictionary, it will send this mismatched symbol called literal to the output stream. On the contrary, when a match is found, it will send the matched outputs (two kinds of tokens: either the (length, offset) or (length, index)) to the output stream. When we set the dictionary size to the object size, the dynamic sliding dictionary becomes the object itself, the literals or the mismatched outputs become the dissimilarity of the object to itself, and the pairs or the matched output becomes the similarity of the object to itself.

9.3.3 The literals and pairs outputs for the range coding

Basically speaking, the range coder of LZMA (MA part) processes the literals and pairs outputs from LZ part separately. The range coder encodes the literals output via its context generated modeler (frequency algorithm). The pairs output pass through another context modeler which focuses on length-offset and length-index pairs. Consequently, there won't be any effect on the pairs outputs if we separate the results (the compressed file size) for literals and pairs and evaluate them independently.

9.3.4 The Refined LZMA compression algorithm and diagram

We customize our compressor to evaluate the similarity of the object by only including the pairs output from the original LZMA compression algorithm. A refined LZMA (pairs

only) compression algorithm process diagram (as shown in Figure 9.6) illustrates the idea of this modification.

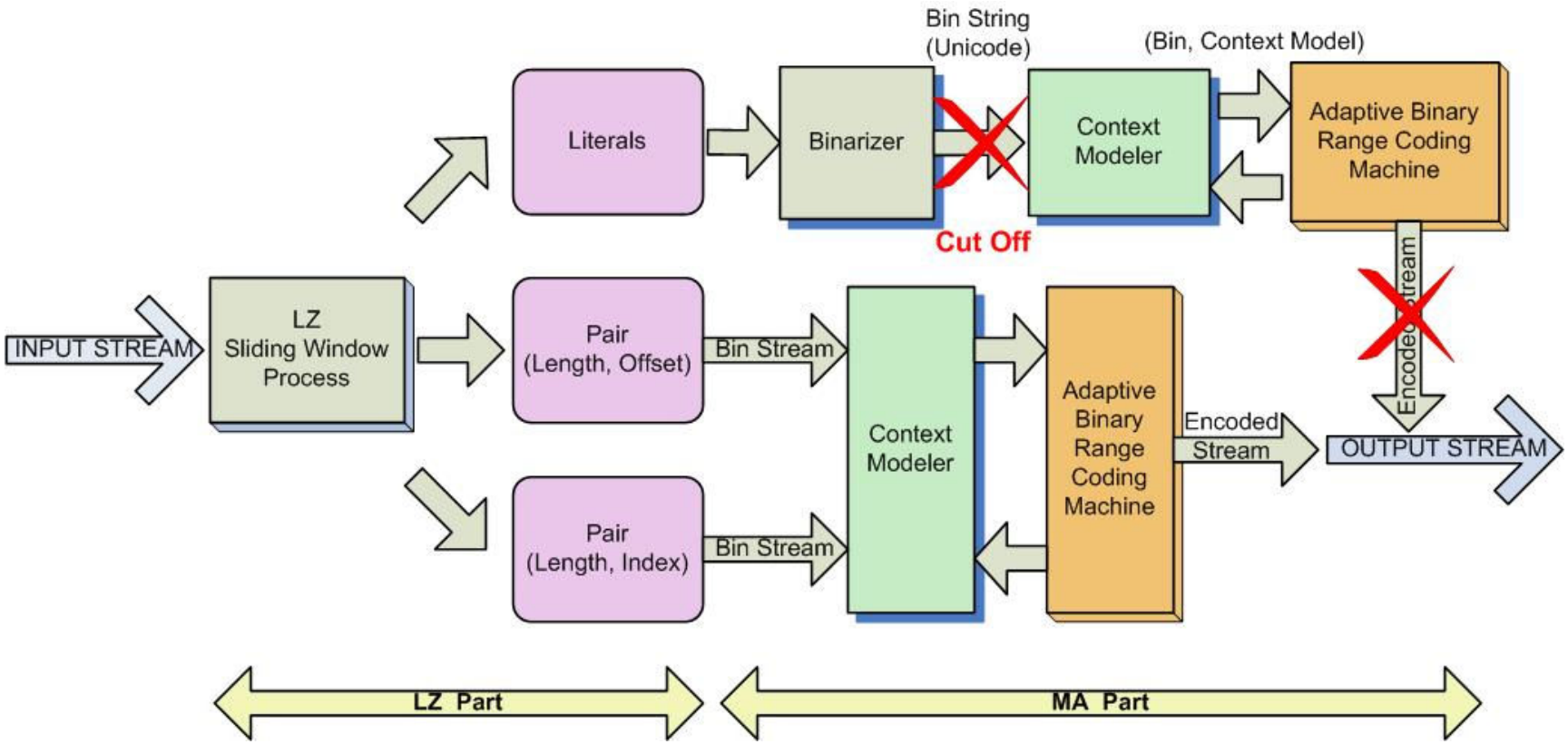

Figure 9.6 The Refined LZMA compression diagram

9.4 The Empirical evaluation

The objective of this experiment is to ask if there is any clustering improvement made by the redefined compression algorithm. Using the same data, design and methodology as in Chapter 7, we will compare NCD using the usual LZMA against NCD using the altered process.

9.5 Interpretation and the evaluation result

The only fact we can confirm from the statistical *Z*-test is the null hypothesis will be rejected or not. From the result given in the Table 9.1, for our new refined LZMA compression algorithm (Pairs NCD), we can reject the null hypothesis with $P<0.05$.

Table 9.1 The Z-test

Z-test		
Compression Algorithm	***z*** statistic	***P*** Significance
01-Regular LZMA-NCD	34.654	<0.000001
02-Refined LZMA (Pairs)-NCD	**35.137**	**<0.000001**

We use effect size for the numerical quantification of the difference between the dissimilar population-LL population and 320 for the similar population-LP population. From the result given in the Table 9.2, we have found the effect size for our refined LZMA compression algorithm (Pairs NCD) is greater than the regular LZMA algorithm.

Table 9.2 The effect size result

	LL Population			LP Population				
Compression Algorithm	Means	Num	SD (Standard Deviation)	Means	Num	SD (Standard Deviation)	SD (pooled)	Effect size
01-Regular LZMA-NCD	0.99303	120	0.0071	0.41316	320	0.31591	0.316437	1.832497087
02-Refined LZMA (Pairs)-NCD	**0.994215**	**120**	**0.013569**	**0.394982**	**320**	**0.313224**	**0.313819**	**1.90948525**

9.5.1 ROC Curve

From the result given in the Table 9.1, we have found both the regular LZMA and refined LZMA compressor have very high TPR and low FPR. From the result given in the Table 9.2, we also have found both regular LZMA and refined LZMA compression algorithm have the AUC value of 1. From the ROC curve plot shown in Figure 9.7, the graph for regular LZMA and refined LZMA compression algorithm, both graphs are near perfect; hence conclusions are difficult to draw. However, from the plot shown in Figure 9.8 which is enlarged five times at the corner point of the Figure 9.7, we can view that the refined LZMA compressor has slightly better (the most upper left corner) performance than the regular LZMA.

Table 9.3 TPR FPR at the corner point of the ROC curve

	01-Regular LZMA-NCD	**02-Refined LZMA (Pairs)-NCD**
TPR	**99.99%>>**	**99.99%>>**
FPR	**<<0.01%**	**<<0.01%**

Table 9.4 AUC result

	01-Regular LZMA-NCD	**02-Refined LZMA (Pairs)-NCD**
AUC	1.000	1.000

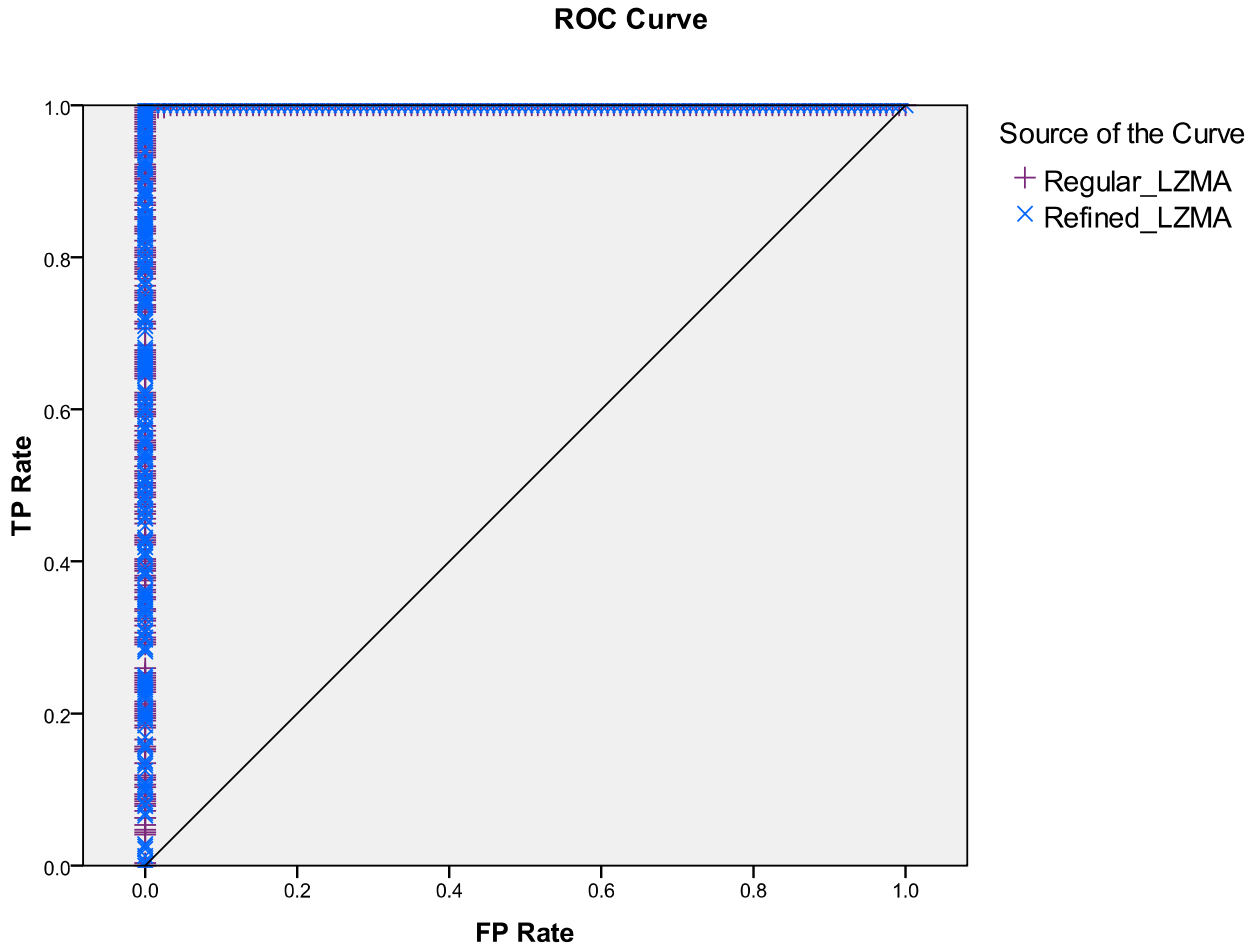

Figure 9.7 The ROC Curve

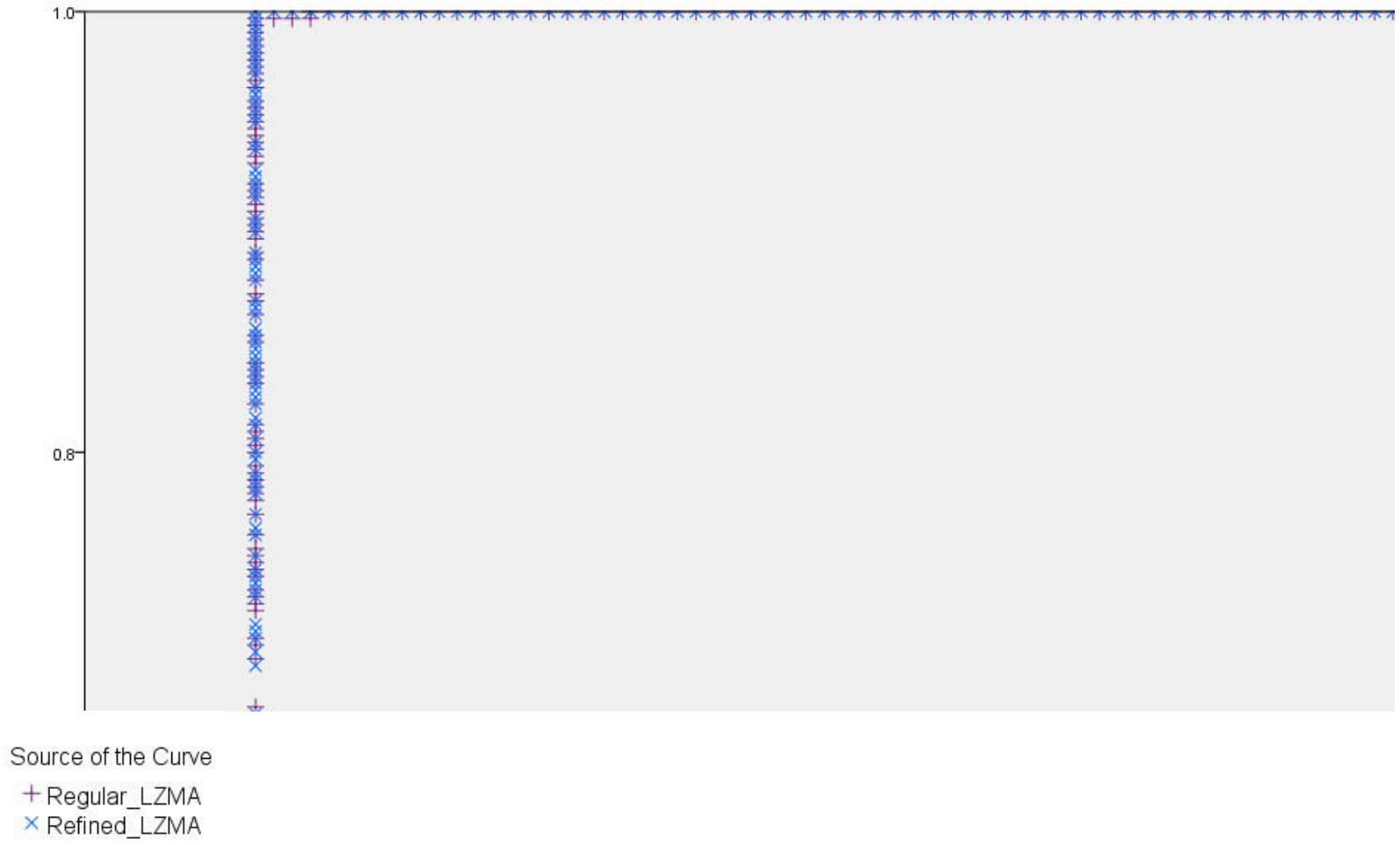

Figure 9.8 Five times larger at the corner point

9.6 Discussion, Conclusions and Future Work

Although our existing NCD clustering mechanism based on regular LZMA compressor has already achieved a remarkable identification performance (TPR >> 99.99%, FPR << 0.01%), from the result shown in Section 9.5, we still have made some improvements for image similarity classification by using our refined LZMA compression algorithm. We have demonstrated that the refined LZMA is a viable alternative to the traditional algorithm.

In future work, we examine the ability of our refined LZMA algorithm to solve the obfuscation problems we identified in Section 7.4 (advertising banners, dynamic DOM tree components). Our refined method provides us the opportunity to extract only the similar components of the webpage but ignore the dissimilar components; by taking only the similarity as a significant feature for objects comparison/classification, our new proposed compression method potentially can be a promising solution to these difficult conundrums.

Chapter 10 Conclusion and future work

10.1 Visually Similar Webpage Identification

Web page similarity detection is widely used by many popular applications. However, the existing methods cannot always successfully identify web pages that humans would perceive as similar. We propose a robust way to evaluate the similarity of web pages from the viewpoint of their reader. The concepts of Gestalt theory and supersignals provide us with a theoretical rationale for the conjecture that web pages must be treated as indivisible entities (i.e. a whole) to be congruent to human perceptions.

10.2 Visual similarity for Phishing detection

We use the domain of Anti-Phishing technology to derive test scenarios for our experiments, as visual similarity between a Phishing page and its target is an essential part of the Phishing scam. In a series of experiments, we have demonstrated that that we are able to consistently discriminate between similar and dissimilar web pages. In a large scale, real-world case study, we have showed that our approach is highly effective at detecting similar web pages, in particular for Anti-Phishing applications. By optimizing the choice of compressors for our NCD system, we also acquire the near perfect classification results. We hope based on these initial results, a novel and robust Anti-Phishing system can be developed in the future; this will need to address issues including the sensitivity and specificity of the NCD technique.

Another issue should be discussed is our mechanism can not prevent some of the specific class of Phishing attack such as the MITM attack or the “man in the middle” attack under the system “hacked” situation. Most of this kind of attacks are involved in the security system breach such as DNS poisoning [124] of personal computer or even the ISP (Internet Service Provider) server. Although we can still identify the Phishing website as the suspicious webpage due to its visual similarity, we might fail to identify it as a Phishing one in that false identification results from the other Phishing clues such as wrong domain name verification caused by the domain name poisoning. In this case, the main identification failure is caused by the malfunction of the user system but not the defect of our mechanism.

10.3 Real world scenario test

We also tested our system using the most popular Web pages to examine its practicality for the real world situation. The low false positives rate shows that when users are browsing websites the chance for our mechanism to identify legitimate websites as a Phishing website is low. This is very important in that such an inconvenience will decrease the confidence of user in any Anti-Phishing tools[29]. Hopefully, our proposed system will be acceptable to the user population.

10.4 Refined compression algorithm

We modified the original LZMA and applied it to our classification application. A better experimental result shows the refined LZMA is a viable alternative to the traditional algorithm. This provides us the opportunity to solve some of the obfuscation problems

cause by the complicated structure of web sites or countermeasures created by Phishers who want to defeat our system.

10.5 Additional Possible Countermeasures

In the future, we will undertake a more complete evaluation of the robustness of our NCD similarity technique against countermeasures Phishers could employ. While we think that it will be difficult (at least effort consuming) for Phishers to evade our NCD similarity technique, we believe it is still important to seek empirical evidence to support this statement. We will attempt to quantify the robustness of the NCD similarity technique using a variation of mutation testing. A mutation will be defined as any operation that alters a webpage to avoid our similarity identification (such as the one in Figure 7.6). These mutations include changes in image color, object coordinates (i.e. rearranging DOM elements), icon resolution, and textual contents (i.e. garbage text, out-of-context phrases, etc.). In mutation testing, the tester attempts to find a test suite that "kills" (i.e. detects) each mutation; in this approach multiple variants of a single program are developed, each containing one or more mutations. The effectiveness of a test suite is measured by the fraction of "mutant" programs killed. In this application, we will be attempting to quantify the effectiveness of a single test (NCD similarity) on a population of mutants. We will also seek an understanding of what mutations (if any) are able to evade this test, and hence of the limitations of the NCD similarity technique. The possible "counter counter-measures" to these "counter measure-mutations" is by using our refined compression algorithm to specifically extract the similarity part from a protected webpage to form a unique signature for our NCD based similarity identification.

10.6 Other Phishing clues

Another avenue of future work is to consider other characteristics of the website that can be identified as a Phishing website. In [72], they mentioned clues such as 1) Age of domain: many Phishing websites only registered their domains for a few days; 2) Known images: a webpage contains its brand logos but is not using its brand domain, then it probably is a Phishing webpage; 3) Suspicious URL and links: a page's URL contains an (@) or (-) in the URL or link for its webpage; 4) IP Address: domain name is an IP address; 5) Dots in URL: Phishing webpages tend to use many dots (.) in their URLs; 6) Forms: the HTML content for the <input> tags contains the labels with sensitive keywords such as "credit card" or "password". None of the listed features on its own can be the mighty silver bullet to "kill" the Phish. Different malicious strategies are developed by the Phishers just to avoid their Phishing webpage to form a specific feature as the "clues" for Phishing detection.

In [29], after performing the a serious tests for human factors about Phishing, they suggest "the standard security indicators (which widely deployed in nowadays Anti-Phishing tools) are not effective for a substantial fraction of users, and an alternative approaches are needed". This alternative method must have the ability to accurately identify the Phishing web pages via its own intelligent decision making mechanism instead of the list. Furthermore, to increase the accuracy of the identification, it has to be heuristic to prevent the "mutation" of Phishing websites. In [125], they mentioned both famous Internet browsers-IE and Firefox put the heuristic functions into their mainly black-list based system for better identification concern. An embedded heuristic Anti-

Phishing mechanism is an absolute necessity in the trend of the Anti-Phishing tool development.

Our proposed method which identifies the Phishing websites via its visual similarity provides a new approach to this area. Although we have proofed it is robust to accurately identify the Phishing websites, it is still important to consider adding other Phishing features to assist in Phish identification. We believe by combining other reliable Phishing website features, we can develop a hybrid heuristic Anti-Phishing mechanism based on our NCD similarity clustering core technology with accurate Phishing webpage identification results in the near future.

This page is intentionally left blank

Bibliography

1. Wertheimer, M., *Gestalt Theory*. 1938, New York: Brace and Co.
2. Dorner, D., ed. *The Logic of Failure*. 1997, Addison Wesley. 39-41.
3. R. Fergus, L.F.-F., P. Perona, A. Zisserman, *Learning Object Categories from Google's Image Search*, in *10th IEEE International Conference on Computer Vision (ICCV'05)*. 2005: Beijing, China p. 1816-1823.
4. Ching-Tung Wu Kwang-Ting Cheng, Q.Z., Yi-Leh Wu *Using visual features for anti-spam filtering*. in *Image Processing, 2005. ICIP 2005*. 2005.
5. AY Fu, L.W., X Deng, *Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD)*. IEEE Transactions on Dependable and Secure Computing, 2006. **3 no.4**.
6. McCall, T., *Gartner 2007 Phishing Survey Report* 2007 Gartner.
7. APWG, *http://www.antiphishing.org/*. 2010.
8. APWG, *Global Phishing Survey: Trends and Domain Name Use in 2H2009*. 2010.
9. AntiPhishingWorkingGroup, *Phishing Attack Trends Report - January 2008*. 2008.
10. Alexa. 2010; Available from: http://www.alexa.com/.
11. Henzinger, M., *Finding near-duplicate web pages: a large-scale evaluation of algorithms*, in *SIGIR Special Interest Group on Information Retrieval* 2006: Seattle, Washington, USA
12. A.Broder, S., Glassman,M.Manasse, and G.Zweig, *Syntactic Clustering of the Web*, in *6th International World Wide Web Conference*. 1997, 393-404.
13. Charikar, M.S. *Similarity estimation techniques from rounding algorithms*. in *Annual ACM Symposium on Theory of Computing Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* 2002. Montreal, Quebec, Canada.
14. Manber, U., *Finding similar files in a large file system*, in *USENIX Winter 1994 Technical Conference*. 1994.
15. Heintze, N. *Scalable Document Fingerprinting*. in *USENIX Workshop on Electronic Commerce*. 1996. Oakland,CA,USA.
16. TH Haveliwala, A.G., D Klein, P Indyk, *Evaluating strategies for similarity search on the web*, in *Proceedings of the 11th international conference on World Wide Web*. 2002: Honolulu, Hawaii, USA
17. D Cai, S.Y., JR Wen, WY Ma, *Extracting Content Structure for Web Pages Based on Visual Representation.* APWeb 2003,LNCS 2642, 2003: p. 406-217.
18. D Shen, Z.C., Q Yang, HJ Zeng, B Zhang, Y Lu, *Web-page classification through summarization*, in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. 2004: Sheffield, United Kingdom
19. J Dean, M.H., *Finding related pages in the World Wide Web.* Computer Networks, 1999. **31**: p. 1467-1479.
20. Y Wang, M.K., *Evaluating contents-link coupled web page clustering for web search results*, in *Proceedings of the eleventh international conference on Information and knowledge management*. 2002: McLean, Virginia, USA

21. Xiaoguang Qi, B.D.D., *Web page classification: Features and algorithms.* ACM Computing Surveys (CSUR) 2009. **41**(2).
22. FireFox3. *http://www.mozilla.com/en-US/firefox/*. 2010.
23. N.Heintze, *Scalable Document Fingerprinting*, in *2 nd USENIX Workshop on Electronic Commerce*. 1996.
24. Kalviainen, M., *The Role of Sign Elements in holistic product meaning*, in *SeFun International Seminar*. 2007.
25. Graham, L., *Gestalt Theory in Interactive Media Design.* Humanities & Social Sciences, 2008. **2**(1).
26. E.Gordon, I., *Theories of visual perception*. 3rd ed. 2004: Psychology Press. p18-19.
27. Kepes, *Language of vision*. 1944: Paul Theobald.
28. Arien Mack, I.R., *Inattentional Blindness*. 1998: MIT Press.
29. Rachna Dhamija, J.D.T., *Why phishing works* in *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2006.
30. Chaitin, *Algorithmic Information Theory*. 1987, Cambridge: Cambridge University Press.
31. M.Li, P.M.B.V., *An Intorduction to Kolmogorov Complexity and its Applications*. 2nd ed. 1997, New York: SpringerVerlag.
32. Rudi Cilibrasi, P.V.a., *Clustering by Compression.* IEEE Transactions on Information Theory, 2005. **51**(4).
33. M.Li, X.C., X.Li,B.Ma, and P.M.B Vitanyi, *The similarity metric.* IEEE Trans.Inf.Theory, 2004. **50**(12): p. 3250-3264.
34. Feldt, R., et al. *Searching for Cognitively Diverse Tests: Towards Universal Test Diversity Metrics*. in *Proc. IEEE Int. C. Software Testing Verification and Validation Wkshp*. 2008. Lillehammer, Norway.
35. Cernian, A., D. Carstoiu, and A. Olteanu. *Clustering Heterogeneous Web Data Using Clustering by Compression. Cluster Validity*. in *Proc. Int. Symp. Symbolic and Numeric Algorithms for Scientific Computing*. 2008.
36. Delany, S.J. and D. Bridge, *Textual case-based reasoning for spam filtering: a comparison of feature-based and feature-free approaches.* Artificial Intelligence Review, 2006. **26**: p. 75-87.
37. Lempel, J.Z.A., *A universal algorithm for sequential data compression.* IEEE Transactions on Information Theory, 1977: p. 23(3):337-343.
38. M. Burrows, D.J.W., *A block-sorting lossless data compression algorithm*. 1994, System Research Center.
39. I.Witten, T.B.J.C., *Data compression using adaptive coding and partial string matching.* IEEE Transactions on Communications, 1984: p. 32(4):396-402.
40. Litan, A., *Number of Phishing Attacks on U.S. Consumers Increased 40 Percent in 2008*. 2009, Gartner.
41. Lemos, R., *Script kiddies: The Net's cybergangs*, in *ZDNet*. 2000.
42. Emigh, A., *Online Identity Theft: Phishing Technology, Chokepoints and Countermeasures*. 2005, Radix Labs.
43. Thunderbird. 2008; Available from: http://www.mozillamessaging.com/en-US/thunderbird/.

44. I Fette, N.S., A Tomasic, *Learning to Detect Phishing Emails*, in *WWW 2007*. 2007: Banff,Canada.
45. IE8. 2010; Available from: http://www.microsoft.com/windows/products/winfamily/ie/default.mspx.
46. Neil Chou, R.L., Yuka Teraguchi, John C. Mitchell. *Client-side defense against web-based identity theft*. in *Network and Distributed System Security Symposium (NDSS)*. 2004.
47. Netcraft. 2009; Available from: http://news.netcraft.com/.
48. Daniel Andresen, T.Y., Omer Egecioglu,Oscar H. Ibarra,Terence R. Smith. *Scalability issues for high performance digital libraries on the World Wide Web*. in *In Proceedings of IEEE ADL '96, Forum on Research and Technology Advances in Digital Libraries*. 1996. Washington D.C
49. Min Wu, R.C.M., Simson L. Garfinkel, *Do Security Toolbars Actually Prevent Phishing Attacks?* , in *Conference on Human Factors in Computing Systems Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006: Montréal, Québec, Canada.
50. MacKay, W.E. *Triggers and barriers to customizing software*. in *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*. 1991. New Orleans, Louisiana, United States
51. R Dhamija, J.T., *The Battle Against Phishing: Dynamic Security Skins*, in *Proceedings of the 2005 symposium on Usable privacy and security*. 2005: Pittsburgh,PA,USA.
52. PassMark. 2008; Available from: http://authentication.wordpress.com/.
53. YahooSignInSeal. 2010; Available from: https://protect.login.yahoo.com/.
54. Chuan Yue, H.W., *BogusBiter: A transparent protection against phishing attacks.* ACM Transactions on Internet Technology (TOIT), 2010. **10**(2).
55. PhishTank. 2008; Available from: http://www.phishtank.com/phish_archive.php.
56. Cebrian, M., M. Alfonseca, and A. Ortega, *Common pitfalls using the normalized compression distance: what to watch out for in a compressor.* Communications in Information and Systems, 2005. **5**(4): p. 367-384.
57. Hescott, B. and D. Koulomzin, *On Clustering Images Using Compression*. 2007, Computer Science Department, Boston University: Boston, MA, USA.
58. Li, M. and Y. Zhu, *Image classification via LZ78 based string kernel: a comparative study.* Lecture Notes in Computer Science, 2006. **3918**: p. 704-712.
59. Lan, Y. and R. Harvey. *Image classification using compression distance*. in *Proc. 2nd Int. C. Vision, Video and Graphics*. 2005. Edinburgh, U.K.
60. Batista, L.V., M.M. Meira, and N.L. Canalcanti Jr. *Texture classification using local and global histogram equalization and the Lempel–Ziv–Welch algorithm*. in *Proc. 5th Int. C. Hybrid Intelligent Systems* 2005. Rio de Janeiro, Brazil
61. Macedonas, A., et al., *Dictionary based color image retrieval.* Journal of Visual Communication and Image Representation, 2008. **19**: p. 464-470.
62. Bardera, A., et al. *Compression-based Image Registration*. in *Proc. IEEE Int. Symp. Information Theory*. 2006. Seattle, WA, USA.
63. Ofuonye, E., et al., *Prevalence and Classification of Web Page Defects.* Online Information Review, In Press.
64. PhishTank. 2009; Available from: http://www.phishtank.com/phish_archive.php.

65. K Strimmer, A.v.H., *Quartet Puzzling: A Quartet Maximum-Likelihood Method for Reconstructing Tree Topologies.* Molecular Biology and Evolution, 1996. **13 no. 7**: p. 964-969.
66. BroadwayPhishTrack. *http://www.dslreports.com/phishtrack*. 2008.
67. Yih, W., J. Goodman, and G. Hulten. *Learning at Low False Positive Rates*. in *Proc. 3rd Conf. Email and AntiSpam*. 2006. Mountain View, CA, USA.
68. Florencio, D. and C. Herley, *Stopping a Phishing Attack, Even When the Victims Ignore Warnings*. 2005, Microsoft Research: Redmond, WA, USA.
69. F Provost, T.F., R Kohavi. *The case against accuracy estimation for comparing induction algorithms*. in *Proceedings of the Fifteenth International Conference Machine Learning*. 1998. Madison, WI, USA.
70. Salomon, D., ed. *Data compression: The Complete Reference*. 4 ed. 2007.
71. Panlov, I. *7Z Format / 7-zip application*. 2009 Retrieved May 29; Available from: http://www.7-zip.org.
72. Yue Zhang, J.I.H., Lorrie F. Cranor. *Cantina: a content-based approach to detecting phishing web sites*. in *Proceedings of the 16th international conference on World Wide Web*. 2007. Banff, Alberta, Canada
73. Guang Xiang, J.I.H. *A hybrid phish detection approach by identity discovery and keywords retrieval*. in *Proceedings of the 18th international conference on World wide web*. 2009. Madrid, Spain
74. Liam Rourke, T.A., D. Randy Garrison,Walter Archer *Methodological issues in the content analysis of computer conference transcripts.* International Journal of Artificial Intelligence in Education (IJAIED), 2001. **12**: p. 8-22.
75. Pierre Baldi, S.B., Yves Chauvin,Claus A. F. Andersen,Henrik Nielsen, *Assessing the accuracy of prediction algorithms for classification: an overview* Bioinformatics, 2000. **16**(5): p. 412-424.
76. Cebrián, M., M. Alfonseca, and A. Ortega, *The Normalized Compression Distance Is Resistant to Noise.* IEEE Trans.Inf.Theory, 2007. **53**(5): p. 1895-1900.
77. Granados, A., et al. *Evaluating the Impact of Information Distortion on Normalized Compression Distance*. in *Proc. 2nd Int. Castle Mtng Coding Theory and Applications*. 2008. Castillo de la Mota, Medina del Campo, Spain.
78. Zhou Wang, A.C.B., Hamid R. Sheikh,Eero P. Simoncelli, *Image quality assessment: From error visibility to structural similarity.* IEEE TRANSACTIONS ON IMAGE PROCESSING, 2004. **13**(4).
79. HR Sheikh, M.S., AC Bovik, *A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms.* IEEE Transaction on Image Process, 2006. **15**(11): p. 3440-3451.
80. Laung-Terng Wang, N.E.H., Edwin H. Porter, John J.Zasio. *SSIM: a software levelized compiled-code simulator*. in *Annual ACM IEEE Design Automation Conference archive Proceedings of the 24th ACM/IEEE Design Automation Conference*. 1987.
81. Alexander Toet, M.P.L., *A new universal colour image fidelity metric* Display, 2003. **24**.
82. Sheikh, H.R., Bovik, A.C.,Cormack, L., *No-reference quality assessment using natural scene statistics: JPEG2000.* IEEE Transactions on Image Processing, 2005. **14**(11): p. 1918-1927.

83. R. Venkatesh Babua, S.S., Andrew Perkisc, *No-reference JPEG-image quality assessment using GAP-RBF* Signal Processing, 2007. **87**(6): p. 1493-1503.
84. Tomás Brandãoa, M.P.Q., *No-reference image quality assessment based on DCT domain statistics* Signal Processing, 2008. **88**(4): p. 822-833.
85. H. M. Quiney, K.A.N., A. G. Peele *Iterative image reconstruction algorithms using wave-front intensity and phase variation.* Optics Letters, 2005. **30**(13): p. 1638-1640.
86. Shai Avidan, A.S., *Seam Carving for Content-Aware Image Resizing.* ACM Transactions on Graphics, 2007. **26**(3).
87. APE Rosiello, E.K., C Kruegel, F Ferrandi, *A Layout-Similarity-Based Approach for Detecting Phishing Pages*, in *Security and Privacy in Communications Networks and the Workshops*. 2007: Nice,France.
88. Eric Medvet, E.K., Christopher Kruegel *Visual-similarity-based phishing detection*. in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*. 2008. Istanbul, Turkey
89. Beker, H., *Cipher Systems: The Protection of Communications*. 1982: Wiley-Interscience.
90. J.G. Cleary, a.I.H.W., *Data compression using adaptive coding and partial string matching.* IEEE Transactions on Communications, 1984. **32**(4): p. 396-402.
91. Acob Ziv, A.L., *A universal algorithm for sequential data compression.* IEEE Transactions on Information Theory, 1977. **23**(3): p. 337-343.
92. Huffman, D.A. *A Method for the Construction of Minimum-Redundancy Codes*. in *Proceedings of the Inssitute of Radio Engineer* 1952.
93. Bruce E. Rosen, J.M.G., Jacques J. Vidal. *Adaptive range coding*. in *Proceedings of the 1990 conference on Advances in neural information processing systems*. 1990. Denver, Colorado, United States
94. Martin, G.N.N., *Range encoding: an algorithm for removing redundancy from a digitised message.*, in *Video & Data Recording Conference*. 1979: Southampton, England.
95. IH Witten, R.N., JG Cleary, *Arithmetic coding for data compression.* Communications of the ACM, 1987. **30**(6): p. 520 - 540
96. ComplearnToolkit. 2003; Available from: http://www.complearn.org/.
97. Bzip2. 2010; Available from: http://www.bzip.org/.
98. CompuServe, H.R.B.C. *Graphics Interchange Format : A standard defining a mechanism for the storage and transmission of raster-based graphics information*. 1987; Available from: http://www.w3.org/Graphics/GIF/spec-gif87.txt.
99. Beckett, D., *The replacement for the GIF format*, in *PC Magazine*. 1995.
100. J, O.N., *Differential pulse-code modulation (PCM) with entropy coding.* IEEE Transactions on Information Theory, 1976. **22**(2): p. 169-174
101. D. S. Taubman, M.W.M., *JPEG2000 -Image Compression Fundamentals, Standards and Practice*. 2001: Kluwer Academic.
102. William B. Pennebaker, J.L.M., *Jpeg: Still Image Data Compression Standard*. 1992.
103. JS Walker, Y.C., TM Elgindi. *Comparison of the JPEG2000 lossy image compression algorithm with WDR-based algorithms*. in *IEEE International Conference on Image Processing*. 2006.

104. A Kocsor, A.K.-F., L Kaján, S Pongor, *Application of compression-based distance measures to protein sequence classification: a methodological study* Bioinformatics, 2006. **22**(4): p. 407-412.
105. GP Telles, R.M., FV Paulovich, *Normalized compression distance for visual analysis of document collections.* Computers & Graphics, 2007. **31**(3): p. 327-337.
106. C.H.Bennett, M.L., B.Ma, *Chain letters and evolutionary histories*, in *Scientific American*. 2003. p. 76-81.
107. Manuel Cebrián, M.A., *The normalized compression distance is resistant to noise.* IEEE Transactions on Information Theory, 2007. **53**(5): p. 1895-1900.
108. Benjamin Hescott, D.K., *On Clustering Images Using Compression*. 2007: Boston, MA, USA.
109. Anton Bardera, M.F., Imma Boada, Mateu Sbert, *Compression-based Image Registration*, in *ISIT 2006*. 2006: Seattle,WA,U.S.A.
110. Ming Li, Y.Z., *Image Classification Via LZ78 Based String Kernel: A Comparative Study*, in *Lecture Notes in Computer Science*. 2006, Springer Berlin / Heidelberg.
111. *Convert HTML to Image*. 2009; Available from: http://www.converthtmltoimage.com/.
112. M Carnec, P.L.C., D Barba, *An image quality assessment method based on perception of structural information*, in *International Conference on Image Analysis and Recognition*. 2003.
113. Farzad Ebrahimi, M.C., Stefan Winkler. *JPEG vs. JPEG2000: An Objective Comparison of Image Encoding Quality*. in *SPIE*. 2004.
114. Coe, R., *It's the effect size, stupid: What effect size is and why it is important*, in *Annual Conference of the British Educational Research Association annual conference*. 2002.
115. Facebook. 2010; Available from: http://www.facebook.com/.
116. Adobe. *Flash*. 2010; Available from: http://www.adobe.ca/flashplatform/.
117. AlexaTop500Sites. 2010; Available from: http://www.alexa.com/topsites.
118. WEKA. 2010; Available from: http://www.cs.waikato.ac.nz/~ml/weka/.
119. Nitesh V. Chawla, K.W.B., Lawrence O. Hall, W. Philip Kegelmeyer, *SMOTE: synthetic minority over-sampling technique.* Journal of Articial Intelligence Research, 2002. **16**(1): p. 321-357.
120. Fawcett, T., *An introduction to ROC analysis* Pattern Recognition Letters, 2006. **27**(8): p. 861-874.
121. Foster Provost, T.F., Ron Kohavi. *The case against accuracy estimation for comparing induction algorithms*. in *Proceedings of the Fifteenth International Conference on Machine Learning*. 1998. Madison, Wisconsin,USA.
122. W. R. Gilks, W.R.G., Sylvia Richardson, D. J. Spiegelhalter, *Markov chain Monte Carlo in practice*. 1996: Chapman&Hall/CRC.
123. Detlev Marpe, H.S., Thomas Wiegand, *Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression* IEEE Transactions on Circuits and Systems for Video Technology, 2003. **13**(7).
124. Stewart, J., *DNS Cache Poisoning – The Next Generation*. 2002, LURHQ Corporation Technical Report.

125. Christian Ludl, S.M., Engin Kirda,Christopher Kruegel. *On the Effectiveness of Techniques to Detect Phishing Sites*. in *Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. 2007. Lucerne, Switzerland