

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Detecting malicious Webpages using content based classification

Permalink

<https://escholarship.org/uc/item/8926j7mk>

Author

Bannur, Sushma Nagesh

Publication Date

2011

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Detecting malicious Webpages using content based classification

A Thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Sushma Nagesh Bannur

Committee in charge:

Professor Lawrence Saul, Co-Chair
Professor Stefan Savage, Co-Chair
Professor Charles Elkan

2011

Copyright
Sushma Nagesh Bannur, 2011
All rights reserved.

The Thesis of Sushma Nagesh Bannur is approved, and
it is acceptable in quality and form for publication on
microfilm and electronically:

Co-Chair

Co-Chair

University of California, San Diego

2011

EPIGRAPH

“The most beautiful thing we can experience is the mysterious. It is the source of all true art and all science. He to whom this emotion is a stranger, who can no longer pause to wonder and stand rapt in awe, is as good as dead: his eyes are closed.”

—Albert Einstein

TABLE OF CONTENTS

Signature Page	iii
Epigraph	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Acknowledgements	ix
Abstract of the Thesis	x
Chapter 1 Introduction	1
Chapter 2 Data	5
Chapter 3 Features	7
3.1 URL	10
3.2 Structural	11
3.3 Page link	12
3.4 Semantic	14
3.5 Visual	15
Chapter 4 Offline classification	20
4.1 Classification algorithms	20
4.1.1 Logistic regression	21
4.1.2 Linear support vector machines	22
4.2 Dimensionality reduction	23
4.2.1 Non-negative matrix factorization	23
4.3 Feature validation	24
4.4 Evaluation	25
4.4.1 Experimental design	25
4.4.2 Results	27
Chapter 5 Realtime classification	35
5.1 Classification algorithms	35
5.1.1 Perceptron	36
5.1.2 Passive-aggressive	36
5.1.3 Confidence weighted algorithm	37
5.2 Evaluation	37

	5.2.1	Experimental design	38
	5.2.2	Results	38
Chapter 6		Discussion	43
Bibliography			45

LIST OF FIGURES

Figure 2.1:	Data collection and feature extraction infrastructure.	6
Figure 3.1:	Feature size for the various feature categories for data collected over 10 days.	9
Figure 3.2:	Cumulative number of features observed over a period of about one month.	9
Figure 3.3:	Keypoints identified by SIFT, arrows indicating the locations, scales and orientations of the keypoints.	17
Figure 3.4:	Output of the SIFT matching algorithm: white lines from the logo on top-left to the one on bottom-left indicate the matching locations identified by SIFT.	18
Figure 4.1:	Classification with tf-idf, bag-of-words and topic-coefficients representation of semantic features.	27
Figure 4.2:	Classification with different visual features.	30
Figure 4.3:	ROC curve for classification using SVM for the optimal feature configuration.	32
Figure 4.4:	Classification results using SVM and LR for different feature categories.	32
Figure 5.1:	Classification on URL features using Peceptron, PA and CW.	39
Figure 5.2:	Classification on features from the body of the Webpage using Peceptron, PA and CW.	40
Figure 5.3:	Classification on combination of URL and body features using Peceptron, PA and CW.	40
Figure 5.4:	Comparison of URL, body, and combination of URL and body features using CW.	41
Figure 5.5:	Cumulative false positives (FP) and false negatives (FN) corresponding to classification results in Figure 5.4.	41

LIST OF TABLES

Table 4.1:	Top weighted words for some of the topics identified by NMF.	28
Table 4.2:	Feature size for the different visual features.	29
Table 4.3:	Classification results using SVM for combinations of features from various categories.	31
Table 4.4:	Actual and relevant feature size for different feature categories.	33
Table 4.5:	Misclassification overlap (%) between different feature categories.	34
Table 4.6:	Classification results using the optimal feature configuration for different datasets with varying ratios of positive to negative samples.	34
Table 5.1:	Actual and relevant feature size for different feature categories.	39

ACKNOWLEDGEMENTS

The best thing about working on the intersection of two topics is that you get to work with two amazing advisors instead of one. Professor Lawrence Saul and Professor Stefan Savage have been an inspiration and guiding force behind this thesis. I consider myself extremely fortunate to have the best advisors I could have asked for. Their faith in me to solve this problem, their ideas and regular feedback have helped me realize my full potential during the course of this work.

Justin Ma helped me bootstrap the project. His earlier work in this field set the foundation on which I was able to continue my research. Some of the tools he had developed also helped me with the implementation part of this project. I am thankful to him and his team at UC Berkeley for providing access to the Twitter dataset.

A lot of work in this project depended on a key set of infrastructure components being up and running. If there was one person who kept it all running smoothly, it was Kirill Levchenko. He was always available to help and resolve issues in a timely manner, so a big thanks to him.

Special thanks to Professor Charles Elkan for agreeing to be on my thesis committee.

ABSTRACT OF THE THESIS

Detecting malicious Webpages using content based classification

by

Sushma Nagesh Bannur

Master of Science in Computer Science

University of California, San Diego, 2011

Professor Lawrence Saul, Co-Chair

Professor Stefan Savage, Co-Chair

In this thesis, we propose a supervised learning approach to detect malicious Webpages. We use features from the URL, textual content, structural tags, page links and visual appearance of a Webpage. First, we demonstrate an offline classification model using batch learning algorithms and evaluate the benefit of including each of the feature types. Then we illustrate an online classification model using online learning algorithms on a larger dataset. For both these models, we use a live feed of labeled data collected from a large webmail provider. For a base rate of 66%, we achieve 98% accuracy using a combination of URL and Webpage content features for classification. We observe that incorporating Webpage content features in addition to the URL features, reduces the error rate by about 50%.

Chapter 1

Introduction

The ubiquity of the Internet and its applications have caused a steady increase in the number of Web based crimes in the past few years. The growing popularity of the World Wide Web, rise in e-commerce, absence of a strong security model for the Web and the incentive to make large profits are driving more cyber criminals towards the World Wide Web. Web crimes result in billions of dollars in losses to both individuals and businesses worldwide. Be it simple or sophisticated Web crimes, the scale and complexity of the Web make it highly vulnerable to security threats. The absence of uniform regulations, compliance and laws across different countries adds more fuel to the menace of Web crimes.

Scams that specifically target the World Wide Web come in a variety of forms, but they fall into a few well-known categories based on what is compromised or stolen.

In one category of attacks, users are spoofed by Websites that masquerade as legitimate ones and try to extract information like usernames and passwords, social security numbers, credit card numbers, etc. In a phishing attack, user information is stolen by Websites that look identical to ones that are known and trustworthy. In a retail fraud attack, counterfeit products without legal licences from the manufacturers are sold through legitimate looking Websites. The consumer either never receives the purchased product, or receives a counterfeit product that is much cheaper and of sub-standard quality compared to the original one. Online pharmacies selling counterfeit drugs are a common example. These Websites are

often backed by commercial advertisements with spam emails.

In another category of attacks, user's computers are infected with malware such as viruses, worms and trojan horses. To facilitate malware injection, adversaries exploit vulnerabilities in either the browser or the applications launched on the user's computer. In some cases, malware can be injected as a result of some explicit user actions. The user is spoofed into downloading malicious binaries by making them appear like legitimate software. Examples of this includes prompting the users to install some multimedia plugins in order to watch online video, or by displaying graphics on the Webpage that resembles regular dialog boxes of operating systems (like a Windows dialog box). Spyware and some forms of adware are examples of software that is installed surreptitiously as a result of some user action. Spyware can track the activities of users and can interfere with normal activity by causing browser redirections and installing programs that run when the operating system starts executing. Key loggers record the keystroke activity that might give them access to passwords, credit cards etc. Some malware can be so damaging that they can completely seize control of the computer to send spamming emails, initiating denial of service attacks and wiping out the hard disk contents. Along with the growing number of Web scams, the nature of these attacks is continuously evolving. What we have described just scratches the surface of the number of possible Web attacks.

A victim is exposed to these attacks by visiting a malicious Webpage. The URL to the malicious Webpage can be propagated by the adversaries in different ways such as emails, social networking, altering results of search engine, posting links on legitimate sites etc. Apart from these explicit means, there is a strong chance of landing on a malicious Webpage as a result of general surfing on the internet. Based on the analysis in [16], a large number of Webpages contain malicious code. This suggests a clear need for a Web filtering system that can inspect a Webpage and detect if it is malicious. Such a system can be incorporated by the browser or various other Web services to identify the malicious Webpage and protect the user from fraudulent activities.

One of the methods commonly used for this task is checking the member-

ship of the URL against blacklists. This involves comparing the properties of the URL such as domain name, IP address etc against a compiled list of identified malicious Websites. As one can imagine, this mechanism is not robust and fails for a number of scenarios as elaborated in [14]. The other technique [15, 16, 17] is to use a Virtual Machine (VM) environment to run browser instances and use instrumentation techniques to investigate the impact of visiting a Webpage. All state changes to the virtual machine are recorded by specifically checking if binaries are downloaded or if any unauthorized file system or system configuration changes occur as a result of visiting the Webpage. Detection using this method is limited to identifying malware injection and does not address any of the traditional spoofing scenarios such as phishing. Moreover, it is a heavyweight operation which might not scale for performing realtime classification. Another technique, which is the focus of this thesis, is to apply machine learning methods based on recognizing prevalent patterns distinguishing malicious and legitimate Webpages. [18, 19, 20] outline approaches designed specifically for phishing detection. [10, 9] demonstrate classification using the information from the URL of the Webpage. We believe that information extracted from the contents of a Webpage can be quite valuable and an improved classification can be achieved by incorporating the same.

With this motivation, we describe a supervised learning approach to identify a malicious Webpage. The suggested approach is not specific to a particular variety of threat and is designed as a generic method to detect any malicious Webpage. We include information from both the URL and the content of the Webpage for learning. We carefully design the feature selection and classification while paying attention to speed and scalability. We extract features from the URL, textual content, structural tags, page links and visual appearance of a Webpage. We analyze the benefit of incorporating features from different categories of information in a Webpage. We first propose a batch learning based offline classification method and perform a detailed analysis to evaluate the model. Finally, we illustrate a real time classification method for detecting malicious Webpages using the features evaluated in batch learning. Our approach identifies malicious Webpages with about 98% accuracy, with marginal number of false positives. We observe

that incorporating Webpage content information in addition to the URL features, reduces the error rate by about 50%.

The outline of this thesis is as follows: Chapter 2 describes the infrastructure used for data collection. Chapter 3 provides a detailed illustration of the features used to represent a Webpage. Chapter 4 elaborates on the offline classification strategy; we discuss the batch learning algorithms used and the evaluations performed. Finally, in Chapter 5 we present a realtime classification method based on the features configured from batch learning evaluations. We explain the online learning algorithms used and summarize our findings.

Chapter 2

Data

To train a classifier to detect malicious Webpages, we need a dataset of labeled examples of Webpages. In this section we describe the data collection infrastructure that is needed. The data for the malicious samples comes from a live feed from a large Webmail provider. The details of the associated feed sources can be found in [13]. The data for the benign samples consists of a random selection of URLs obtained from Twitter. These Twitter URLs have been already inspected and confirmed as legitimate. Considering the immense variety of URLs on Twitter, we expect the collection to be a fairly diverse and unbiased sample. The data from the feed comprises of the URL and the raw HTML contents of the Webpages. In case of a Webpage with URL redirection, the raw data corresponds to the URL of the final landing page. Thus we never get shortened URLs as input. In addition to the raw Webpage data, we also use a screenshot of the Webpages to be further analyzed using computer vision methods. We render the screenshots of Webpages using the screengrab utility of firefox. We build the necessary infrastructure to collect and process the data from these feeds in real time. The raw data is processed as soon as it arrives to gather the information required to represent the Webpage. The malicious Websites are generally short lived, as they are taken down soon after being detected as malicious. Thus, it is necessary to perform the feature extraction in real time when the data is fresh to guarantee that we have access to the original Webpage that was marked malicious.

Figure 2.1 describes the data collection and feature extraction infrastruc-

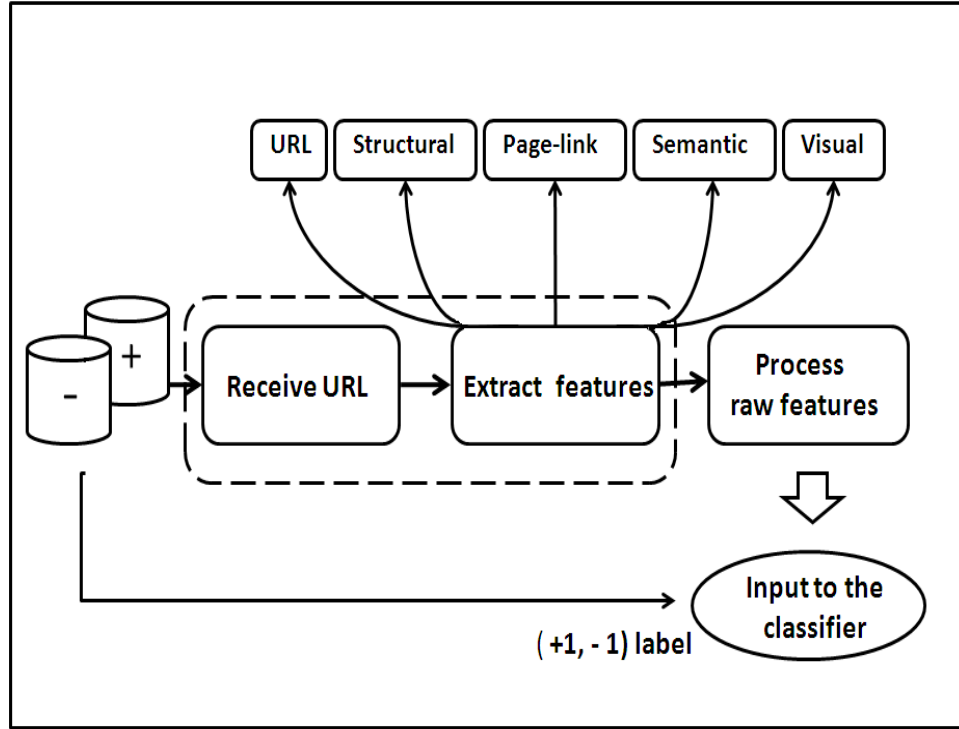


Figure 2.1: Data collection and feature extraction infrastructure.

ture. Chapter 3 outlines the URL, structural, page-link, semantic and visual features in detail.

The ratio of malicious to benign URLs in the collected data is approximately 1:2. The live feed supplies approximately 6000 URLs per day. For the offline classification experiments we use a smaller set of data, corresponding to the data collected over a period of about 10 days. For realtime classification we integrate the classification engine with the data collection and feature extraction system, performing classification over about one month of data.

Chapter 3

Features

Features extracted from a Webpage define a measurable representation of the Webpage that can be used by the classification algorithm to distinguish between malicious and benign samples. We extract features from both the URL and the content of the Webpage. In this section we provide a detailed outline of our feature extraction design.

The contents of a Webpage provide numerous types of information, some of which are directly visible to the end user and some of which are hidden. Visible information includes the URL, textual information, non-textual information in the form of images, audio and video, page links to other sources and the overall visual appearance of the Webpage. The hidden information consists of the HTML code of the Webpage, the embedded scripts and the stylesheets used for formatting. We extract features from both the hidden and visible content of the Webpage. Additionally, we gather a number of indirect features using the information available on the Webpage.

As explained in Chapter 2, it is necessary to extract features from a Webpage as soon as it is received. Thus, our feature extraction step is tightly coupled with data collection. We observe that extracting features when the data is fresh improves classification results.

The gathered information is processed to represent each Webpage by a vector of numerical values containing boolean, integer and real valued components. When converting the raw data to numerical values corresponding to the features,

we ensure that the increase or decrease in the values corresponds to a meaningful correlation. The features are derived from a diverse variety of attributes from the Webpage and hence each feature can assume a range of numerical values that is completely different from another. It is important that we scale all the features to ensure that features with large numerical values do not dominate the classification output. We scale the features such that each feature is within the range $[0, 1]$. We scale the features using the min-max scaling method. For a feature vector $\mathbf{x}$, the normalized value of each feature component $\mathbf{x}_i$ is:

$$\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i - \mathbf{x}_{i_{min}}}{\mathbf{x}_{i_{max}} - \mathbf{x}_{i_{min}}} \quad (3.1)$$

where $\mathbf{x}_{i_{min}}$ and $\mathbf{x}_{i_{max}}$ are the minimum and maximum value of the feature component $\mathbf{x}_i$ over all examples.

We categorize the information gathered for a Webpage as URL, structural, page link, semantic and visual features. Features corresponding to each of these categories are explained in detail in the following sections. Gathering information from the various attributes of a Webpage results in a very high dimensional feature representation with a feature size of about a half a million for the data collected over 10 days. However, most of these features are sparse. Figure 3.1 outlines the feature size for each of the feature categories. Figure 3.2 presents the growth in the cumulative number of features over a period of about one month. With such a large number of features used, it is important to confirm that a significant number of features are participating in the classification decision. Section 4.3 explains the algorithm used for this task.

While gathering information from a Webpage, it is sometimes required to strike a balance between the features extracted and the time it takes to extract those features. For example, when extracting the features from the page links of a Webpage, we limit the feature extraction to a few links to ensure it completes within acceptable time limits. Basically, we make a trade-off between speed and classification performance.

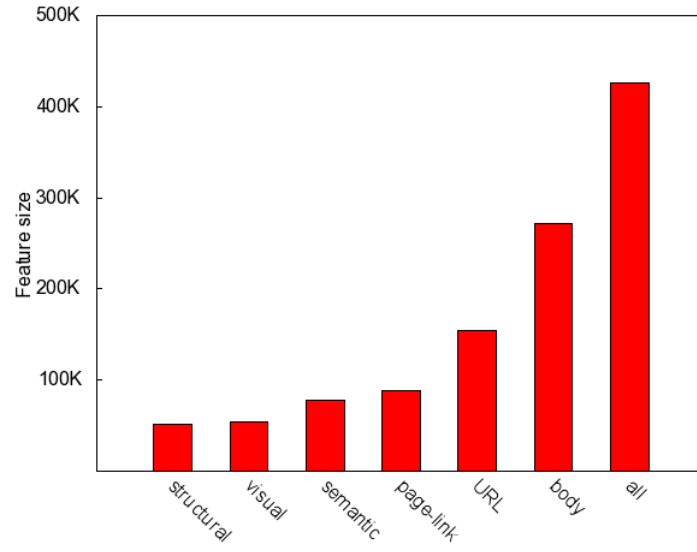


Figure 3.1: Feature size for the various feature categories for data collected over 10 days.

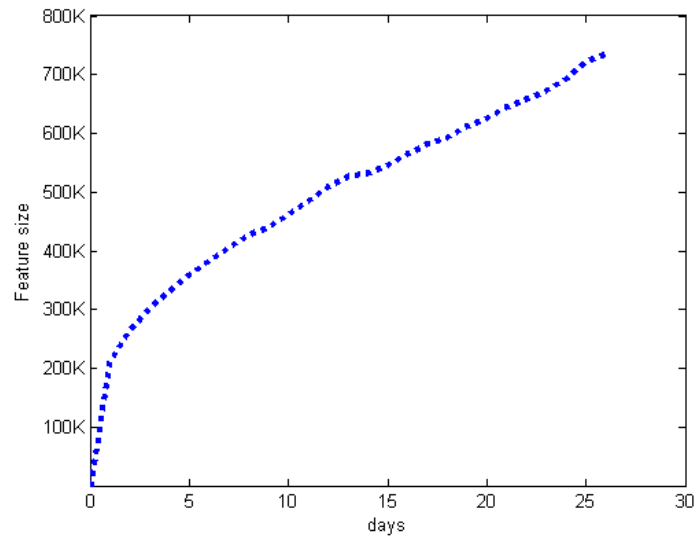


Figure 3.2: Cumulative number of features observed over a period of about one month.

3.1 URL

[9, 10] illustrate detection of malicious webpages using just URL information. URL features are categorized into lexical and host based features. We incorporate the described features and additionally propose some new URL features. We first give a brief overview of the features included from [9, 10] and then propose some new URL features.

Lexical features correspond to the tokens present in the URL. These tokens are found using /, ?, ., = and - as delimiters and distinguished based on their occurrence in the specific part of the URL. Length of the URL and hostname are also included as separate features.

Hostname features describe the properties of the entity hosting the Webpage. The whois information provides details on when the domain name was registered and updated, as well as the associated registrar and registrant name. These attributes constitute the whois features. Blacklist membership is included as a feature, which is determined by looking up the hostname against known malicious hosts. Features based on the location of the host constitute the geographical features. These include information about the associated city, state, language, connection speed etc. Other than these, DNS related properties are also incorporated as features.

In addition to the features described above, we design other URL features. We include the information in the http header as the header features. Header carries a number of useful details about the Webpage such as content-type, content-language, server, mime-version, cookie, the http status code etc. For example, header fields *server* and *x-powered-by* indicate the software used by the server to handle the request. It is likely that a Webpage injecting malware might be running outdated versions of software with known vulnerabilities. Another example is the information in the http status code: it is unlikely to associate an error page as being malicious. We include most of the http header details as features except for information that is related to the time of crawling.

Based on the observation that it is uncommon for a legitimate Webpage to contain an explicit port id after the domain name in the URL, we include the

existence of the port id as a feature. Similarly, it is rare for a legitimate Webpage to have an explicit ip address as the domain name; therefore we also include this as a feature.

A Webpage with https protocol has a digital certificate. The certificate authority (CA) issuing the certificate validates that the Webpage is legitimate and trusted. However, it is important to check the credibility of the issuer. We use the details on the CA issuer and the start/end dates of the certificate as features. For any feature which is a date, its numeric value in the feature vector corresponds to the measure of elapsed or remaining time.

3.2 Structural

The HTML code contains the entire description of the Webpage. Only part of this information is visible to the user on the Webpage; the rest remains hidden. We use the hidden information which describes the structure of the Webpage as structural features.

To get a high level summary of the structure, we gather information from the HTML tags that appear in the HTML code. We model this by counting the number of times each HTML tag appears. This simple HTML tag count can give an idea if the Webpage has content which requires attention, such as if the page contains any forms, embedded objects, scripts, etc. For example, a Webpage created for a phishing attack contains a form where the victim is expected to enter his credentials. Another scenario could be heavy use of scripting languages such as javascripts. It is common to use javascripts to frame malicious attacks. For example, javascript code can be used to find out the vulnerabilities present on the victims computer which can be further used to design appropriate exploits.

Additionally, the HTML tag feature describes the general layout of the page - if the page contains a title, sub headings, tables etc. It also gives information on specific display formatting - if the HTML contains stylesheets, specific fonts, frames etc. Layout of the Webpage describes how much attention is paid to make the Webpage look good. Considering that malicious pages are likely to be designed

with careful attention so as to attract users, the layout information is useful.

To capture detailed structural properties, we include the HTML attribute associated with the HTML tags as features. Specifically, we count the number of times an HTML attribute and an HTML tag-attribute pair appear in the HTML code. Some HTML tag attributes are assigned values; for selected meaningful value types, we incorporate the HTML tag attribute and value pair as a feature. For example, the value of *type* attribute for the *input* tag within a *form* conveys useful information. An input type of *radio* or *checkbox* looks less suspicious than an input type *text*. The *language* attribute for a script specifies the scripting language used; it is likely that some scripting languages are more commonly used for framing malicious attacks than others. For example, it is common for scripts designed for malicious attacks to contain obfuscated code and certain scripting languages are easier to obfuscate. Thus, these languages might be a common choice for attacks that use code obfuscation.

3.3 Page link

Links on the page have a strong influence in determining if a Webpage is malicious. We incorporate a number of features summarizing the properties of the links on the Webpage.

We include features defining the number of internal links, external links, and total number of links. A page link is an internal link if its domain name matches that of the base URL; otherwise it is an external link. A Webpage containing only internal links appears less suspicious than the one containing external links. The number of unique hostnames on the page is included as a feature, which explains the number of different domains the external links on the page are directed to. For external links, we also count the number of links associated with a particular URL protocol.

Objects such as images, audio, video, plugins, binaries etc are often embedded within a Webpage with a link to the source of the object. To capture information on the objects embedded within a Webpage, we incorporate features

that define the file types associated with the page links that appear on the Webpage. Specifically, based on the associated file type we categorize the page link as image, video, audio, data, text, binary, Web, game, system file, compressed file, code or display related file. We include features specifying the number of times page links associated with a specific file category appear on the Webpage. Moreover, the extension of the file gives further details. We count the number of times a particular file extension appears on the Webpage. We expect that certain object types are more likely to be suspicious. For example, it is known that a lot of malicious attacks are framed using image, audio and video files.

We explore if there exists a pattern associated with the size of these files. Thus, we include features corresponding to the minimum, maximum and median file size for every file extension type that appears on the Webpage. Computing the file size by downloading the file can be very expensive for large file sizes, hence we retrieve this information from the *Content-Length* attribute in the http header of the file. Extracting this information from the header for every link can be expensive. In practice, the file size is more useful for image, audio and video files, and hence we compute the file size only for these types.

All the URL features described in Section 3.1 can be applied to the links on the page. However, extracting all of these features for every link can turn out to be computationally expensive. Moreover, it is possible that including all the features may blow up the feature size to a very large value resulting in a deteriorated classification performance. Hence, we carefully select the URL features applicable to the page links ensuring it is computationally efficient.

We summarize the URL lengths of all the links on the page by computing the mean and variance of the URL lengths. We count the number of links with hostnames specified as explicit IP address. We count the number of links with a particular URL protocol. Since extracting the header attributes for every link can be expensive, we limit the header properties only for links that are either image or video extensions. We summarize the header features by counting the number of times a specific header attribute was identified.

The network properties of the URL being powerful features, we incorporate

the WHOIS and geographical features for the page links. These properties being unique for a given hostname, we skip the explicit computation for links with the same hostname. In order to keep the computation efficient, we limit the number of hostnames to 20. For the domain name registration dates, we include the minimum, maximum and the median value of these features over all the links. We perform blacklist lookups for the page links, limiting the number of hostnames inspected as mentioned earlier. We skip the lexical features and the DNS features for the page links, owing to the sparsity and high dimensionality of these features.

3.4 Semantic

The semantic features correspond to the textual information visible on the Webpage. The text on the Webpage identifies the associated context, indicating if the Webpage is about shopping, banking, news, sports etc. We investigate if certain textual information is more likely to be associated as malicious than benign. For example, a shopping Website selling clothes sounds less suspicious than a Website selling male enhancement drugs. Also, certain words in the text such as “password” and “credit-card” can convey important information.

We use the Hpricot parser to distill the textual information. We preprocess the text to remove stop words and filter text such as numbers and most single characters. Extracted text is modeled as a bag-of-words, simply counting the number of times a word appears in the text. We also use the length of the text as a feature.

We also explore the term frequency-inverse document frequency (tf-idf) model, often used in text-mining. In a tf-idf model, a weight is assigned to each word in the document representing the importance of the word in the document relative to the entire corpus. Term frequency defines the frequency of the word within the document and inverse document frequency is given by the inverse of the number of documents containing that word. The weight assigned to a word in a document is the product of the term-frequency and the inverse document frequency. The idea is to come with a model such that a word that occurs rarely

in the corpus characterizes the document well compared to the common words which appear in most documents in the corpus.

[19] uses textual information for phishing detection. However, they generate a lexical signature using only the 5 words with largest tf-idf weights. These words are further used to define a Google search engine query. This is different from our approach where we use all available textual information to derive the textual features.

For Webscale learning, bag-of-words representation over large vocabularies result in very large number of textual features. We suspect that too many features can hurt performance as well as scalability. To address these issues, we explore an alternative topic modeling representation for the semantic features. Specifically, we model the textual information on the Webpage as a distribution over topics. Intuitively, these topics capture higher-order features of each Webpage’s content and context. Section 4.2 elaborates on the application of topic modeling.

3.5 Visual

Visual features represent the information related to the appearance of the Webpage. We use visual features to encode attributes as simple as the colors on the page and as complex as the other previously recognized images.

To extract various visual features we process the image obtained from the screenshot of the Webpage. We limit our analysis to the holistic screenshot and do not consider the images on the page as candidates for individual visual feature extraction. It is interesting to investigate the visual features of individual images within the Webpage. However, retrieving each image and computing individual visual features can be very expensive and might not be practical over a large scale. In our experiments, we include features defining the color histogram, the holistic image summary and the recognizable local image patterns within the Webpage.

Color histogram of an image gives the distribution of the various colors in the image. It is a vector of counts of the number of pixels corresponding to each unique color. We compute the color histogram over RGB colorspace. We

preprocess the image by converting it to a standard size so that all samples have a uniform size. As the number of color shades in RGB colorspace can be as large as $256 \times 256 \times 256$, we limit the color space to a precomputed fixed color palette. We use a 3-3-2 bit color palette corresponding to 256 unique colors. We use color histogram features to analyze if there exists any similarity between the pattern of colors used for each class of Webpages.

We capture the holistic visual appearance of the Webpage using a low dimensional feature vector. The idea of using the holistic representation is to obtain the high-level summary of the visual appearance. We do not want to focus on recognizing individual objects here but rather broadly try to understand the visual category to which the Webpage belongs. For example, we could have different categories corresponding to Webpages containing only text and no images, Webpages containing many small images and the ones containing single large image.

For this task, we use *gist* [11], which estimates the holistic visual structure using a few perceptual dimensions which describe the spatial properties of the image. Spatial properties measure each image’s *degrees of naturalness, openness, roughness, expansion* and *ruggedness*. If we consider just the gist features, we expect the similar looking Webpages to be projected close to each other in the multi-dimensional gist feature space. We preprocess the input to gist by converting the screenshots to grayscale and resizing the image to 100×100 size for computational efficiency.

We use *SIFT* (Scale-Invariant-Feature-Transform) [12] to identify local visual objects in each Webpage and also to obtain a global summary. SIFT computes the feature descriptors at particular points of interest in an image such that the resulting representation is invariant to scale, orientation, affine distortion and noise. This makes SIFT ideal for matching images based on the similarity of their local visual content.

The SIFT algorithm detects the keypoints representing the image. These keypoints correspond to the high contrast regions of the image such as local object edges. Gradient directions and magnitudes are computed in a local region around each keypoint and these orientations define the keypoint. Each keypoint fea-

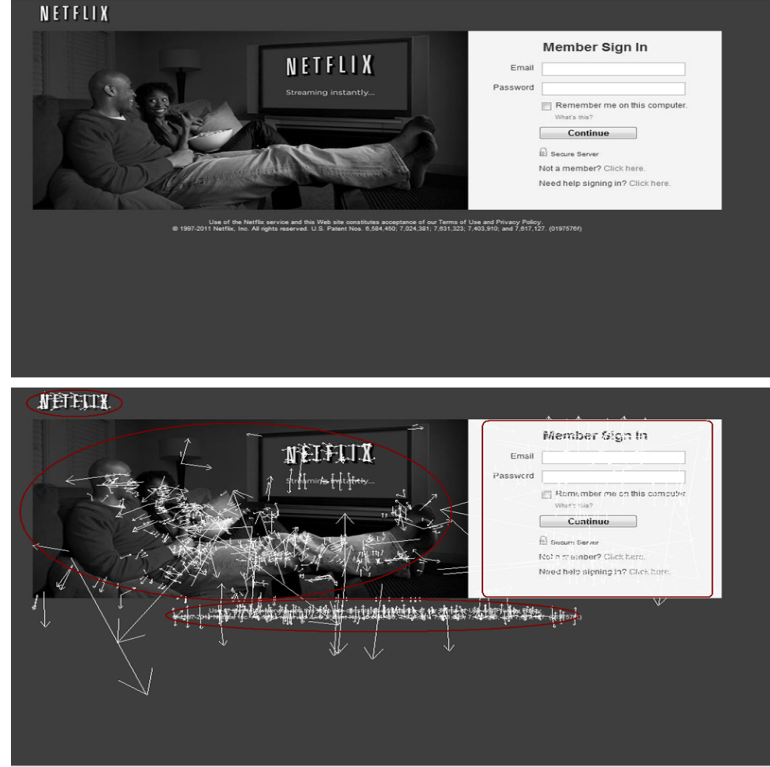


Figure 3.3: Keypoints identified by SIFT, arrows indicating the locations, scales and orientations of the keypoints.

ture descriptor has 132 dimensions with 128 dimensions for the orientations, and the remaining for coordinate, scale and rotation. Figure 3.3 illustrates keypoints detected for a sample Webpage.

We use SIFT to extract visual features in two ways. Firstly, we incorporate features corresponding to the statistics of the SIFT keypoint descriptors. We include the number of keypoints, and the mean and variance of the SIFT keypoint orientations as features. This corresponds to a 257 dimensional feature vector. We refer to this representation of SIFT features as *sift-stats*.

Secondly, after using SIFT to identify the local visual objects in the Webpage, we match the local images found in the Webpage against a set of known images corresponding to popular logos. The repository of known images consists of images with logos of popular banks, e-commerce brands, social-networking sites etc. We use the SIFT matching algorithm for recognizing logos in Webpages.

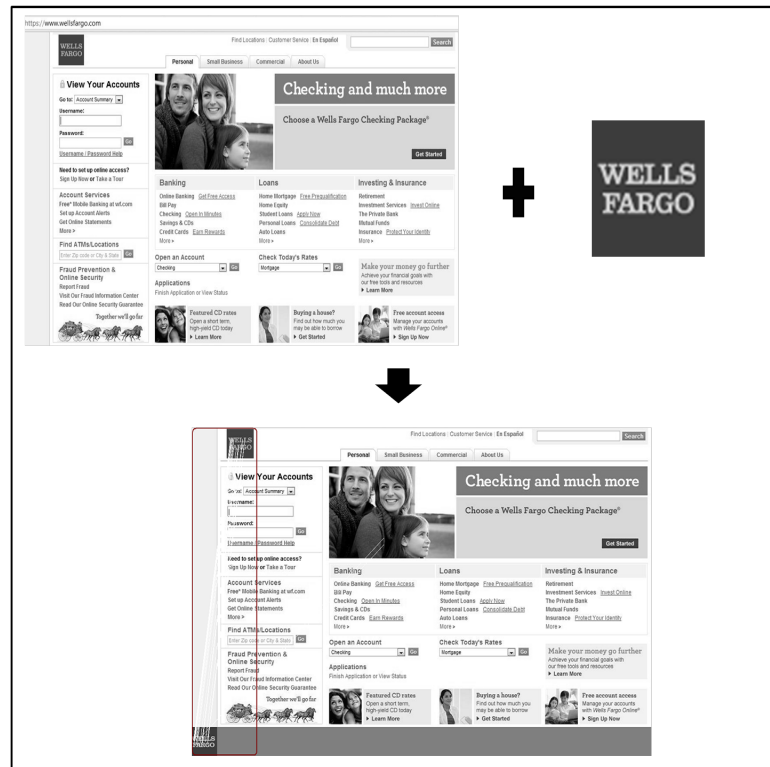


Figure 3.4: Output of the SIFT matching algorithm: white lines from the logo on top-left to the one on bottom-left indicate the matching locations identified by SIFT.

The matching algorithm identifies the local objects in a Webpage by searching the images in the repository to find the nearest neighbors to the Webpage keypoint descriptors. Distance between keypoints is computed using Euclidean distance measure. The images in the repository are assigned a score based on their distance to the local object in a Webpage. Images with scores below a threshold are discarded and the one with the highest score is the best match. Figure 3.4 presents the output of the sift matching algorithm for a sample Webpage.

We include features indicating if a particular logo was identified in the Webpage and the associated matching score. It is possible that an exact match is not found and there are a number of candidate matches with about the same matching score. Hence, we also incorporate information for the logos which are candidate matches but do not have the highest matching score. We refer to this set of SIFT features as *sift-matching*.

Chapter 4

Offline classification

Classifying a Webpage as malicious or benign is an instance of the binary classification problem. In offline classification, also known as batch learning, the classification model is built in one step using a finite set of training data and the trained model is deployed for making future predictions.

Offline learning has the advantage of building a precise model for a given training data because it can learn by considering all the examples in the training set. However, offline learning algorithms are not practical for large scale learning when the data is much larger than the available physical memory. We apply offline learning as the first step towards designing and evaluating the optimal feature configuration for the task of malicious Webpage detection.

Section 4.1 gives a brief overview of the offline classifiers explored. We investigate the benefit from dimensionality reduction when very high dimensionality is a concern; this is outlined in Section 4.2. Section 4.3 discusses the feature validation technique applied to scrutinize the achieved classification. Finally, Section 4.4 gives a detailed description of the experiments performed and summarizes the experimental results.

4.1 Classification algorithms

The training data includes a data matrix $\mathbf{X}$ consisting of n feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ and a label matrix $\mathbf{Y}$ consisting of n labels $y_i \in \{-1, +1\}$ such that y_i is

the label of example $\mathbf{x}_i$. A malicious Webpage is associated with a positive label and the benign Webpage with a negative label. The problem at hand being of large scale learning, we select classification algorithms that are fast and scale well for large number of samples with very high dimensionality. We limit our experiments to linear classifiers.

Linear classification involves learning the parameter $\mathbf{w}$ of a linear function $f(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x} + b)$ that specifies the output label for example $\mathbf{x}$. We use logistic-regression (LR) and linear support vector machines (LSVM) for our evaluation.

4.1.1 Logistic regression

LR is a probabilistic model for binary classification where the probability of a positive label for the input sample $\mathbf{x}$ is determined by the value of the logistic function:

$$P(y|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}} \quad (4.1)$$

The binary output label is computed by thresholding the output of the logistic function. The model is trained to learn the parameter $\mathbf{w}$ that maximizes the conditional log-likelihood. A positive component in $\mathbf{w}$ indicates a positive correlation between the associated feature variable and increases the probability of the positive label. Conversely, a negative correlation is characterized by a negative component in the weight vector.

Regularization is incorporated to build a robust classifier that avoids over-fitting the training data. The idea is to avoid very large components in $\mathbf{w}$ to ensure that the prediction is not dominated by a single feature variable regardless of the others. This is achieved by including the minimization of the magnitude of the $\mathbf{w}$ in the objective function. The regularization strength μ determines the balance between maximization of conditional log-likelihood and penalty for large weight components. The model is trained by maximizing the l_2 -norm regularized objective function:

$$LCL(\mathbf{w}) = \sum_{i=1}^n \ln P(y_i|\mathbf{x}_i) - \mu \|\mathbf{w}\|_2^2 \quad (4.2)$$

We use the logistic regression implementation for large scale learning from [3] in our work.

4.1.2 Linear support vector machines

SVMs are known to provide a robust solution for the problem of binary classification on high dimensional data. SVM finds a hyperplane that partitions the positive and negative samples in the vector space such that the margin of correct classification is maximized. To achieve a maximum margin classifier, the algorithm tries to minimize the hinge loss function:

$$l(f(\mathbf{x}), y) = \max\{0, 1 - y(\mathbf{w} \cdot \mathbf{x} + b)\} \quad (4.3)$$

Incorporating l_2 -norm regularization and considering the average hinge loss over all examples, the objective to be minimized is:

$$L(\mathbf{w}) = C \sum_{i=1}^n \max\{0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i)\} + \|\mathbf{w}\|_2^2 \quad (4.4)$$

where C is the SVM cost function. Value of C determines the trade-off between the regularization and minimizing the loss function. The objective function being convex, the solution is globally optimal.

The described SVM model is called the unconstrained primal formulation. There exists an equivalent corresponding dual formulation where the model is trained to learn the support vector $\boldsymbol{\alpha}$ that maximizes the objective function:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (4.5)$$

subject to the constraints $0 \leq \alpha_i \leq C$.

There exist quadratic programming solvers to solve the formulated constrained optimization problem efficiently. The parameter $\mathbf{w}$ is computed from the support vectors $\boldsymbol{\alpha}$ as:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \quad (4.6)$$

The parameter $\mathbf{w}$ is a linear combination of the input samples where α determines the contribution of any input sample to the classifier. The output label for an input feature $\mathbf{x}$ is obtained by thresholding the dot product $\mathbf{w} \cdot \mathbf{x}$.

We use the Linear SVM implementation for large scale learning from [3] for our work.

4.2 Dimensionality reduction

Dimensionality reduction can be used to extract low dimensional features from the original features that are very high dimensional. Such feature extraction transforms the original data in d dimensions to k dimensions such that $k \ll d$. We explore non-negative matrix factorization (NMF) for dimensionality reduction.

4.2.1 Non-negative matrix factorization

NMF [1, 2] is a dimensionality reduction algorithm that learns a part based representation of the data. It factorizes the data matrix $\mathbf{X}$, into non-negative matrices $\mathbf{V}$ and $\mathbf{H}$ with dimensions $d \times k$ and $k \times n$ respectively, such that $\mathbf{X} \approx \mathbf{V}\mathbf{H}$. $\mathbf{V}$ represents the basis vectors of the data matrix and $\mathbf{H}$ provides the distribution over the basis vectors, the low dimensional representation of $\mathbf{X}$.

Divergence between $\mathbf{X}$ and $\mathbf{V}\mathbf{H}$ can be measured in different ways. We use the Kullback-Leibler divergence given by:

$$D(\mathbf{X} \parallel \mathbf{V}\mathbf{H}) = \sum_{ij} [\mathbf{X}_{ij} \ln(\mathbf{X}_{ij}/(\mathbf{V}\mathbf{H})_{ij}) - \mathbf{X}_{ij} + (\mathbf{V}\mathbf{H})_{ij}] \quad (4.7)$$

The divergence is minimized using multiplicative update rules. In the training phase, $\mathbf{V}$ and $\mathbf{H}$ are computed w.r.t the training data $\mathbf{X}$ using:

$$\mathbf{H}_{an} \leftarrow \mathbf{H}_{an} \frac{\sum_i \mathbf{V}_{ia} \mathbf{X}_{in} / (\mathbf{V}\mathbf{H})_{in}}{\sum_j \mathbf{V}_{ja}} \quad (4.8)$$

$$\mathbf{V}_{ia} \leftarrow \mathbf{V}_{ia} \frac{\sum_n \mathbf{H}_{an} \mathbf{X}_{in} / (\mathbf{V}\mathbf{H})_{in}}{\sum_p \mathbf{H}_{ap}} \quad (4.9)$$

The low dimensional features $\mathbf{H}$ serve as the input to the classification algorithm. For the test samples $\mathbf{X}'$, for which the output label is to be predicted, the low

dimensional representation $\mathbf{H}'$ is computed using the basis vectors $\mathbf{V}$ identified for the training data:

$$\mathbf{H}'_{am} \leftarrow \mathbf{H}'_{am} \frac{\sum_i V_{ia} \mathbf{X}'_{im} / (\mathbf{V} \mathbf{H}')_{im}}{\sum_j V_{ja}} \quad (4.10)$$

Minimization of divergence as described above does not involve any learning rate to be specified. This saves significant computation time as selecting an optimal learning rate can be expensive for large data.

We apply NMF to represent the HTML textual content of a Webpage as a distribution over topics. The NMF part based representation corresponds to the semantic topics in the Webpage text. The non-negativity constraints enforce the textual content to be modeled as additive combinations of topics, which correlates to the fact that the textual content of each Webpage belongs to a subset of topics. NMF reduces the high dimensional *bag-of-words* representation of the HTML textual content to low dimensional *topic coefficients*. The topic coefficients correspond to the NMF extracted features. As NMF discovers the semantic features from context, it is capable of handling synonymy and polysemy.

4.3 Feature validation

With the large number of features used for the task of Webpage classification, it is important to understand the contribution of the features. As the positive samples and negative samples are collected from different feeds, we need to validate that the classification decision is not biased by a specific characteristic of the current feed which cannot be generalized to always hold. We analyze the classification to confirm that (i) the prediction is not dominated by a handful of features overfitting the training data, (ii) a reasonable number of features are participating in classification, and (iii) the involved features are generic.

We use logistic regression with l_1 -regularization for this task. The objective function is:

$$L(\mathbf{w}) = \sum_{i=1}^n \ln P(y_i | \mathbf{x}_i) - \mu \|\mathbf{w}\|_1 \quad (4.11)$$

The regularization strength μ controls the number of non-zero components in weight $\mathbf{w}$; with the number of non-zero components decreasing with increasing μ . The set of solutions for a range of regularization values is computed. The range of regularization values explored defines the regularization path. The idea is to keep increasing μ till the classification accuracy degrades significantly. We can then validate that the number of non-zero weight components for a reasonable classification accuracy is not due to a handful of features.

If it is observed that only a handful of features obtain high classification accuracy, then the corresponding features are investigated. If the features are found to be specific to the training data, then those features are not included for the classification experiments.

We use the l_1 -norm regularized logistic regression implementation from [4] for the task of feature validation.

4.4 Evaluation

In this section we outline the experiments performed for offline classification and provide a detailed analysis of the experimental results.

4.4.1 Experimental design

We perform the offline classification experiments on data collected over a period of 10 days. This corresponds to about 60,000 Webpages with 1:2 split between positive and negative samples. We use 70% of the data for training and 30% of the data for evaluating the prediction of the trained classifier. We repeat the classification over 10 random splits of data and report the average value of the classification results.

We perform a preliminary analysis for semantic features to select between tf-idf, bag-of-words or topic modeling representation for the textual content. Similarly, for the visual features we analyze the gist, sift-matching, sift-stats and color-histogram features individually.

First, we present a classification model combining features from each of

the URL, structural, page link, semantic and visual categories. We then evaluate the classification using features from each category independently to get an estimate of the contribution of a particular category of features. We demonstrate the classification using Linear SVM and LR.

Each of the URL, structural, page link, semantic and visual features provide different information about the Webpage. It is interesting to understand how the classification using features in one category differs from the others. We explore this by finding the overlap between classification output of classifier trained on each of these features individually. A lower overlap between different classification outputs indicates that the classifier learns something new from features in each category.

We carefully select the parameters for the different algorithms. Parameters for LR and LSVM are selected by performing 10 fold cross validation over the training data.

When applying topic modeling over the textual content of the Webpages, it is important to select the right number of topics k to represent the entire data. We repeat NMF for different number of topics in the range 100 to 1500 and select k based on the divergence cost. We select k which minimizes the divergence such that increasing the number of topics further would not provide a large decrease in the divergence cost.

We validate the contribution of the features towards the classification output using LR with l_1 -regularization. We analyze the features associated with positive and negative weights and measure the relevant feature size given by the number of features with non-zero weights.

We keep the ratio of positive to negative samples 1:2 for the above described experiments. However, we inspect the performance of the classifier for varying splits of positive and negative samples. We want to confirm that the classification performance does not deteriorate significantly for skewed distributions of positive and negative samples.

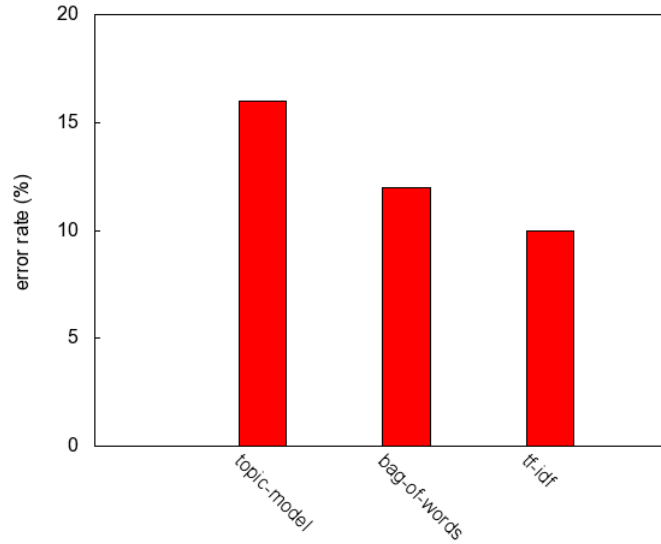


Figure 4.1: Classification with tf-idf, bag-of-words and topic-coefficients representation of semantic features.

4.4.2 Results

In this section we present the results for the experiments outlined above.

For NMF based topic-modeling for semantic features, we find that the model identifies about 500 topics to represent the dataset. Table 4.1 lists the top weighted words for some of the topics identified by NMF.

We perform a preliminary analysis on the semantic features by comparing the classification results using the tf-idf, bag-of-words and topic coefficients representation for the textual content. Figure 4.1 details the classification results using SVM for the different semantic feature representations. We observe that topic-coefficient approximation performs poorly compared to the full textual content representation and tf-idf achieves the best results. Hence, we select tf-idf as the optimal representation for semantic features. From here on, semantic features refers to the the tf-idf representation of the textual content.

We perform a similar analysis to determine the useful visual features among the gist, sift-matching, sift-matching *exact*, sift-stats and color-histogram features. Table 4.2 details the feature size for the different visual features. For sift-matching

Table 4.1: Top weighted words for some of the topics identified by NMF.

hormone	signed	shipping	lesbian	itunes	mortgage	error	craps	slim
sildenafil	sign	shopping	slut	song	logins	timed	blackjack	cambogia
soma	form	code	horny	avenue	debt	temporarily	slots	polynicotinate
medication	case	discount	busty	hardcore	wealth	unavailable	roulette	gymnema
cure	password	cart	blowjob	streaming	bank	server	poker	extract
active	lock	refund	tits	listening	mortgages	problem	casino	garcinia
sexual	caps	special	pussy	play	applicants	moments	promotions	exercising
impotence	sensitive	checkout	cock	music	homeowner	busy	games	berry
levitra	username	prices	asian	played	freedom	proxy	video	antioxidant
erection	log	clearance	amateur	recommend	banks	respond	seven	dieting
erectile	forgot	affordable	anal	load	equity	again	stars	powerslim
dysfunction	remember	handbags	dildo	log	loan	try	promotions	pounds
cialis	logged	shoes	ebony	hq	payments	network	conditions	running
viagra	id	jewellery	bondage	jazz	interest	page	terms	jog
tramadol	unchecked	shirts	cams	hiphop	financing	few	game	resveratrol

Table 4.2: Feature size for the different visual features.

Feature	Size
Gist	960
Sift-matching <i>exact</i>	339
Sift-matching	53077
Sift-stats	257
Color-histogram	256

features we consider two representations, sift-matching and sift-matching *exact*. Sift-matching *exact* includes information only for the logos with the highest matching score whereas sift-matching incorporates the logos with highest matching score as well as the logos corresponding to the candidate matches. Other than the sift-matching features, all the other visual features are dense and hence can be computationally expensive for classification. Additionally, the overhead involved for extracting the visual features is much higher than the other feature categories. This makes it important to analyze the benefit of including these features.

We expect the visual features to contribute more when combined with the URL and other content features. However, we consider that if a particular feature category is useful, it will be reflected partially when used independently. Figure 4.2 illustrates the classification results using SVM for the different visual features used individually. We observe that there is no gain from the gist and the color-histogram features. Both the sift-matching and sift-stats features are useful, with sift-matching performing better than sift-matching *exact*. We discard the gist and sift-matching *exact* features. We consider the sift-matching, sift-stats and color-histogram features as visual features for further evaluations. We include color-histogram features assuming it might be useful in combination with the sift features.

To find the ideal feature combination, we perform classification using SVM for different combinations of the features from the various categories. We begin with the URL features and demonstrate the gain by adding the features in the other categories. Table 4.3 summarizes the classification results. In Table 4.3, feature

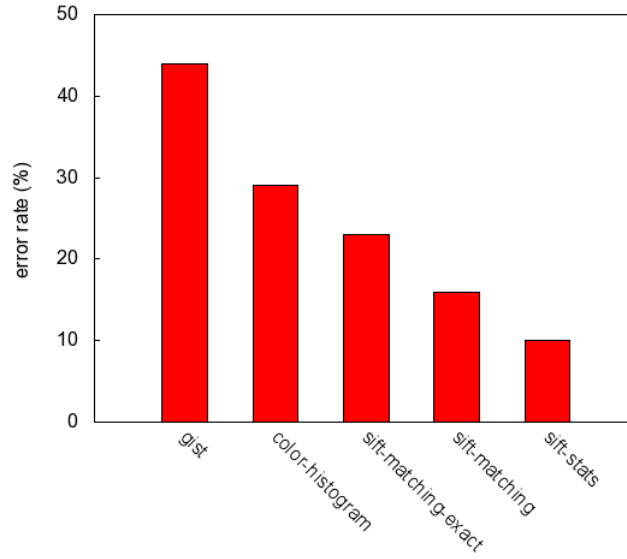


Figure 4.2: Classification with different visual features.

category URL* corresponds to the URL features from [9, 10] and feature category URL combines URL* with the additional URL features proposed in Section 3.1. We find that classification using the URL and the content features achieves classification accuracy upto 98.1%, reducing the error rate by more than 50% compared to using just the URL features. We notice that the ratio of false positives to false negatives remains about the same for the different feature combinations. We observe that incorporating sift-matching and the color-histogram features does not result in noticable gain in classification accuracy. Thus, here onwards we use only sift-stats features to represent the visual features. Except for the gist and sift-matching visual features, we note that it is beneficial to combine features from all the other categories. The combination of URL, structural, page-link, semantic tf-idf and sift-stats features results in highest classification accuracy. We consider this combination of features as the optimal feature configuration. Figure 4.3 presents the ROC curve for classification with this feature configuration.

We measure the gain from features in each of the URL, structural, page-link, semantic and visual categories by performing classification using the features in each category individually. For visual features, we limit the evaluation to only sift-stats features. It is interesting to study the variation in classification results using

Table 4.3: Classification results using SVM for combinations of features from various categories.

Features	Error rate(%)	FP	FN
URL*	4.3	378	405
URL	4.2	372	376
URL + structural	2.6	223	254
URL + structural + page-link	2.5	175	274
URL + structural + page-link + semantic	2.4	187	253
URL + structural + page-link + semantic + sift-matching	2.4	177	260
URL + structural + page-link + semantic + sift-stats	1.9	141	204
URL + structural + page-link + semantic + sift-matching + sift-stats	2.0	149	220
URL + structural + page-link + semantic + sift-stats + color-histogram	1.9	141	205

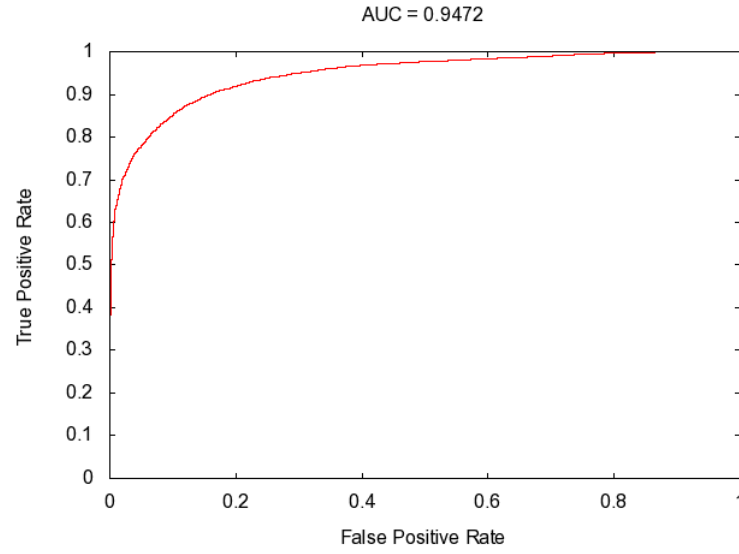


Figure 4.3: ROC curve for classification using SVM for the optimal feature configuration.

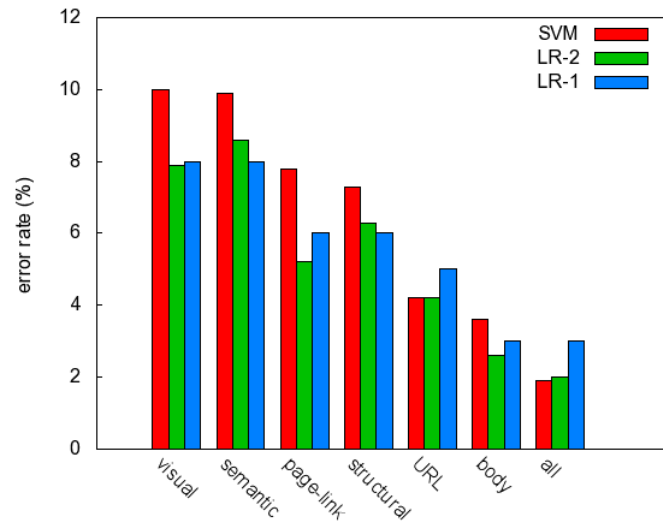


Figure 4.4: Classification results using SVM and LR for different feature categories.

Table 4.4: Actual and relevant feature size for different feature categories.

Feature Category	Actual	Relevant	Positive	Negative
URL	153,918	7,608	5,713	1,895
Structural	51,973	4,742	2,568	2,174
Page-link	88,251	9,657	6,556	3,101
Semantic	77,689	8,170	3,634	2,083
Visual	257	257	106	151
Body	272,387	11,368	6,949	4,419
All	426,305	15,470	9,875	5,595

different classification algorithms. Figure 4.4 illustrates the classification results for the various feature categories using LR and SVM. *LR-1* corresponds to LR with l_1 -norm regularization and *LR-2* corresponds to LR with l_2 -norm regularization. We observe that LR and SVM provide similar classification results, with LR-2 performing better.

Table 4.4 details the number of features for the different feature categories, outlining the actual feature size and the relevant features participating in classification. We notice that the classification is not dominated by a handful of features. Also, the number of features with positive and negative weights seems to be almost balanced. To compute the relevant features, we run LR with l_1 -norm regularization, increasing the value of regularization strength in small steps until the point that the classification accuracy begins to deteriorate. The classification results LR-1 in Figure 4.4 present the classification accuracy associated with the feature validation experiment.

Table 4.5 displays the classification error overlap between the output of classifier trained on the features in different categories individually. The reported percentage value is computed w.r.t the feature category with higher number of misclassifications. A low value of overlap for the different feature category pairs illustrates that features in each category provide new information and the classifier is likely to predict with higher accuracy by combining features from the different

Table 4.5: Misclassification overlap (%) between different feature categories.

	Structural	Page-link	Semantic	Visual
URL	14.1	21.9	14.5	5.0
Structural	-	22.3	19.6	8.9
Page-link	-	-	20.2	9.7
Semantic	-	-	-	8.9

Table 4.6: Classification results using the optimal feature configuration for different datasets with varying ratios of positive to negative samples.

Ratio	Precision(%)	Recall(%)
1:2	97.6	96.6
1:5	95.4	92.9
1:10	94.5	86.7
1:15	94.9	85.0
1:20	94.2	81.6

categories.

We evaluate the classification for the optimal feature configuration on different datasets with varying ratios of positive to negative samples ranging from 2 to 20. We perform classification using linear SVM. We use 70% of the data for training and 30% for testing. The ratio of the positive and negative samples is same for the training and test data. For unbalanced data, accuracy is not an ideal measure of classification success and hence we use precision and recall to evaluate the classification. Table 4.6 presents the precision and recall measures for classification on different datasets.

Chapter 5

Realtime classification

In realtime classification, also known as online learning, the data sample consists of an infinite sequence of input samples and the classifier learns incrementally from each input sample.

Online learning algorithms have the advantage of being simple, fast, scalable and robust. The incremental learning combined with a simple update rule makes the algorithm fast. Since the algorithm looks at only one example at a time, unlike batch learning, it is not required to load the entire dataset in memory and hence scales very well for large-scale learning. As the classifier learns continually, it can add new features and can generally adapt to changes in data, resulting in a robust classifier.

Section 5.1 gives an overview of the linear classifiers used. Section 5.2 describes the experiments performed and summarizes the experimental results.

5.1 Classification algorithms

For online learning, the classifier makes a prediction for $\mathbf{x}_t$, the input sample at time t , and then receives feedback based on the true label y_t , and updates the model incrementally using the feedback for the single input sample. The linear classifiers explored are described below.

5.1.1 Perceptron

Perceptron [5] is one of the simplest and oldest classification algorithm. It distinguishes between the positive and negative samples by learning a hyperplane that partitions the input vector space. The classifier learns the parameter $\mathbf{w}$ that defines the partitioning hyperplane. The output label for an input sample $\mathbf{x}$ is defined by $\text{sign}(\mathbf{w} \cdot \mathbf{x})$. Whenever the classifier makes a mistake, it updates the parameter $\mathbf{w}$ with the update rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t \quad (5.1)$$

However, the simple update rule does not capture the severity of misclassification. Thus, the classifier fails to make precise updates based on the amount of misclassification.

5.1.2 Passive-aggressive

Passive-aggressive(PA) algorithm [6] learns the weight parameters $\mathbf{w}$ to find a hyperplane which separates the positive and negative input samples with a margin of atleast 1. The margin of prediction is defined as $y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$, for instance $\{\mathbf{x}_t, \mathbf{y}_t\}$. The algorithm tries to minimize the hinge loss defined as:

$$l(\mathbf{w}) = \max\{0, 1 - y(\mathbf{w} \cdot \mathbf{x})\} \quad (5.2)$$

A loss is incurred only when the classification margin $y(\mathbf{w} \cdot \mathbf{x}) < 1$. The algorithm solves the following constrained optimization problem:

$$\mathbf{w}_{t+1} = \arg \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}_t - \mathbf{w}\|^2 \text{ s.t } y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 1 \quad (5.3)$$

The algorithm remains passive and does not update the weights when the classification margin achieved is atleast 1. However, the weights are updated aggressively when the margin is less than 1. Closed form solution for the optimization problem gives the following update rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \beta_t y_t \mathbf{x}_t \text{ where } \beta_t = \frac{l(\mathbf{w})}{\|\mathbf{x}_t\|^2} \quad (5.4)$$

Thus, passive-aggressive algorithm makes precise updates based on the severity of misclassification.

5.1.3 Confidence weighted algorithm

Confidence Weighted(CW) algorithm [8, 7] learns the confidence measure for each feature and updates the parameters in accordance to the measured confidence. The idea is that the confidence for the weight associated with a feature is related to frequency of occurrence of the feature such that the rare features correspond to a low confidence measure. The mistake in classification is attributed more to the less confident features and the weights for these features are updated more aggressively than the high confidence features.

A Gaussian distribution $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ is used to model the confidence in the parameter such that $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ indicate the parameter confidence associated with feature i . A rare feature with low confidence measure is characterized by high deviation from the mean value for the feature parameter.

The algorithm updates $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ by selecting the distribution with minimum KL Divergence to the current distribution such that the probability of correct prediction in round t is atleast the value of the defined confidence hyperparameter $\eta \in [0, 1]$. The constrained optimization problem is:

$$\begin{aligned} (\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) &\leftarrow \arg \min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \parallel \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)) \\ &\text{s.t } \Pr[y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 0] \geq \eta \end{aligned} \quad (5.5)$$

Solution to the constrained optimization problem results in following update rule:

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &\leftarrow \boldsymbol{\mu}_t + \alpha_t y_t \boldsymbol{\Sigma}_t \mathbf{x}_t \\ \boldsymbol{\Sigma}_{t+1}^{-1} &\leftarrow \boldsymbol{\Sigma}_t^{-1} + \alpha_t \phi u_t^{-\frac{1}{2}} \text{diag}^2 \mathbf{x}_t \end{aligned} \quad (5.6)$$

where α_t , u_t and ϕ are detailed in [7].

5.2 Evaluation

In this section we present the experiments performed for online learning and discuss the results.

5.2.1 Experimental design

For realtime classification, as we receive a data sample from the live feed, we extract and process the raw features and perform classification immediately. Classification is tightly coupled with data collection and feature extraction, all three steps being performed in realtime. We evaluate the realtime classification over a period of about a month, corresponding to a data of about 156,000 webpages. The ratio of the positive and negative samples is 1:2, same as that used for offline classification. As classification is performed in realtime, the classifier receives the data samples in chronological order.

We perform classification using the URL, structural, page-link, semantic and sift-stats features. We decide the features to be used for online learning based on the offline classification evaluation results. We analyze the classification results obtained using the URL, body and combination of all the features. We compare the classification achieved using Perceptron, PA and CW algorithms for these features.

Perceptron and PA algorithms do not require any parameter setting. For CW we choose the value of η to be 0.95.

We evaluate the feature size for each of the feature categories, measuring the actual feature size and the relevant feature size.

5.2.2 Results

In this section we outline the results for the experiments described above.

Figure 5.1, Figure 5.2 and Figure 5.3 present the results for classification with URL, body, and combination of URL and body features using different classification algorithms. We note that the CW algorithm achieves the lowest error rate, performing significantly well compared to PA and perceptron. These results are consistent with the definition of these classification algorithms. We observe that the classifier has a very steep learning curve in the beginning which tends to stabilize after training for about 10 days. This observation holds true for URL, body, and combination of URL and body features.

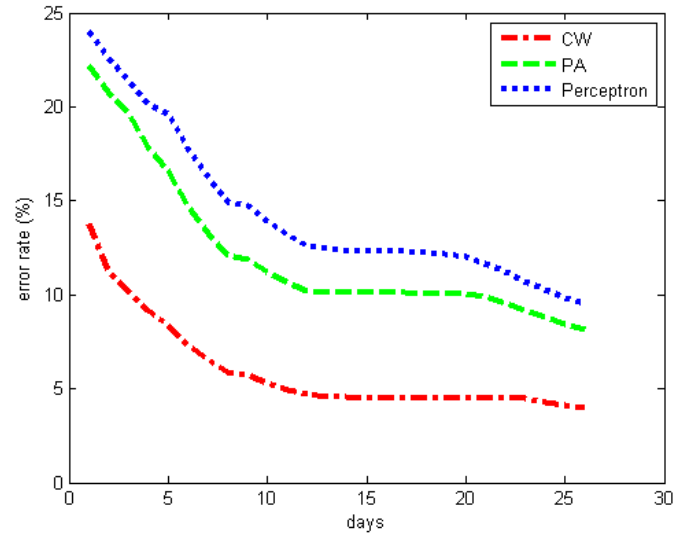


Figure 5.1: Classification on URL features using Peceptron, PA and CW.

Table 5.1: Actual and relevant feature size for different feature categories.

Feature Category	Actual	Relevant	Positive	Negative
URL	302,331	112,108	43,295	68,813
Body	432,702	208260	90,151	118,109
All	735,033	254,127	111,091	143,036

Figure 5.4 compares the classification achieved with URL, body, and combination of URL and body features. These results correspond to classification using CW algorithm. We observe that the realtime classification results are consistent with the findings for offline classification experiments detailed in Chapter 4. Classification with URL features attains about 96% accuracy and combining all the features provides about 98% accuracy. Figure 5.5 outlines the false positives and false negatives corresponding to these results. We find that including the features from the body of the Webpage reduce both the false positives and false negatives equally, reducing the error rate by more than 50%.

Table 5.1 elaborates the feature size for the different feature categories,

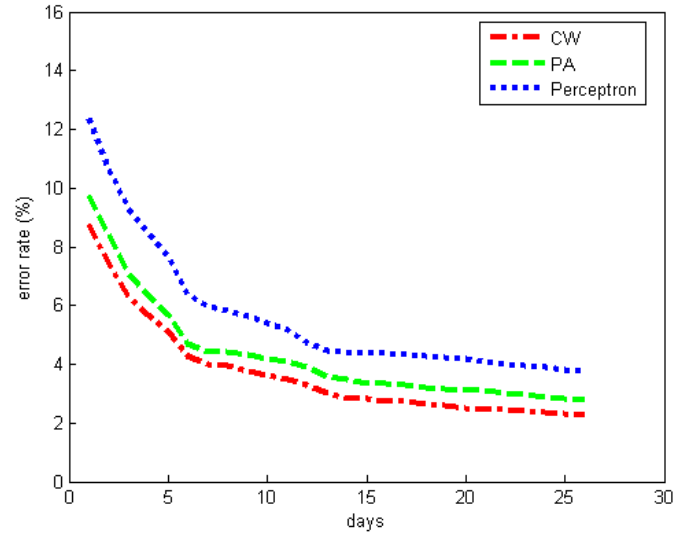


Figure 5.2: Classification on features from the body of the Webpage using Perceptron, PA and CW.

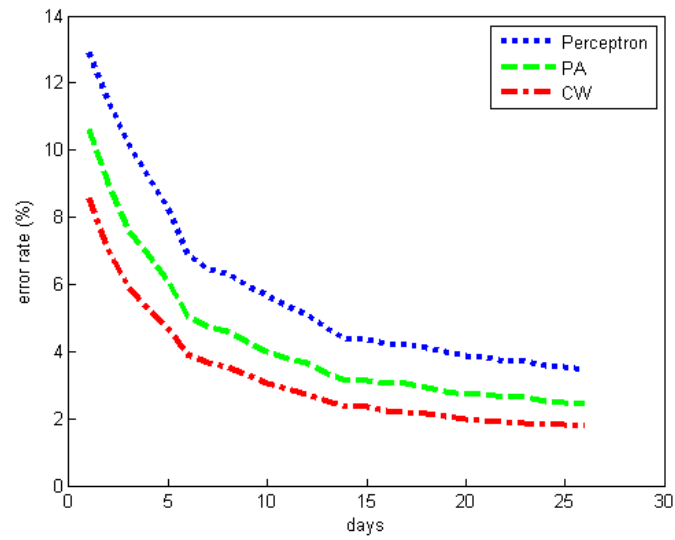


Figure 5.3: Classification on combination of URL and body features using Perceptron, PA and CW.

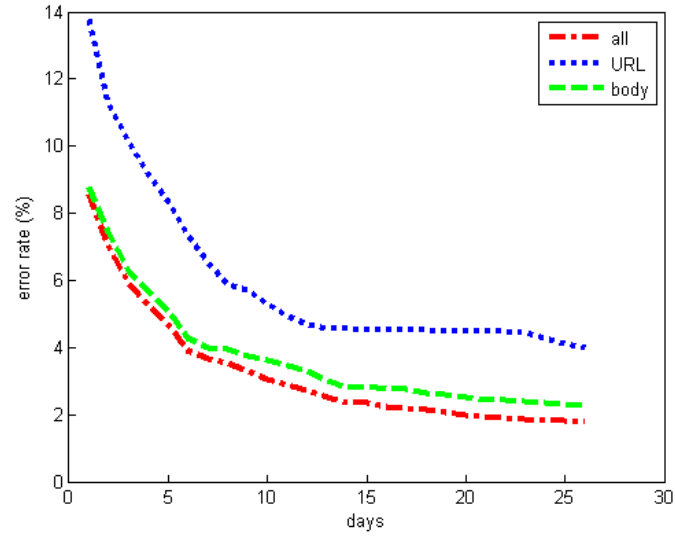


Figure 5.4: Comparison of URL, body, and combination of URL and body features using CW.

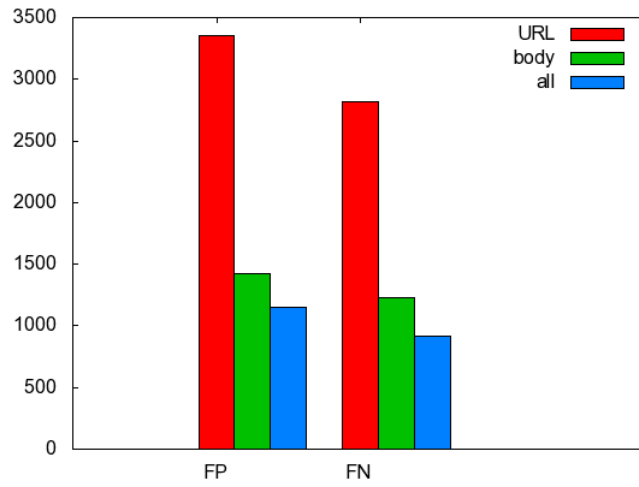


Figure 5.5: Cumulative false positives (FP) and false negatives (FN) corresponding to classification results in Figure 5.4.

presenting the actual and relevant feature size. On comparing these numbers with the feature size measurements made for data over 10 days in Table 4.4, we note that the feature sizes have almost doubled. The feature size is growing as new Webpages are visited and hence an online learning setup provides the classifier a chance to learn continually as new features get introduced.

Chapter 6

Discussion

In this thesis we present the design and evaluation of a supervised learning approach to detect the malicious Webpages. One of the key contributions of our work is the design of the features that can be used for classifying a Webpage. We explore features from the URL, structural, page-link, semantic and visual content of the Webpage. We demonstrate the gain from incorporating features from these different feature categories by performing classification in batch mode. Our experiments validate that the classifier learns something new from the features in each category. Based on the classification results, we confirm that the combination of URL, structural, page-link, semantic and sift-stats features provides optimal classification. We find that including the color-histogram, sift-matching and gist features does not improve classification. Combined with the URL features, among the different categories of features from the body of the Webpage, we observe that including the structural features results in a significant drop in the error rate, followed by the sift-stats visual features. What makes the structural features more useful is that they are lightweight, as the extraction of these features is fast and the feature size is relatively small. Hence, the structural features scale very well for large data. On the other hand, computation of the visual features is relatively expensive. Our method of using just the sift-stats features provides a way of including the visual features without a huge computational cost.

Our offline classification experiments on the data from the live URL feed illustrate that the proposed classification method can identify malicious Webpages

with an accuracy of about 98%. Our evaluations confirm that including the features from the content of the Webpage in addition to the URL features reduces the error rate by more than 50%. We also find that the classification result is not sensitive to the classification algorithm used, as both LR and SVM achieve similar results. Analysis on datasets with varying ratios of positive and negative samples indicates that the classification results do not deteriorate significantly as the data gets highly unbalanced.

Our realtime classification experiments demonstrate that with the latest online learning algorithms we can achieve accuracy as high as batch learning algorithms. The findings in realtime classification experiments reiterate the observations made for offline classification.

Our future goal is to deploy the real time classification method on much larger data. In our current experiments the feature size exceeds the number of examples. It is interesting to evaluate the classification when the data size is larger than the number of features. We believe that studying the behavior of different features on large scale data might provide useful insights. We would also like to evaluate the suggested approach further, exploring new features and using non-linear classification algorithms.

Bibliography

- [1] D.D. Lee, H.S. Seung. *Algorithms for Non-negative Matrix Factorization*, Advances in Neural Information Processing Systems 14, pages 556-562. MIT Press: Cambridge, MA, 2001.
- [2] D.D. Lee and H.S. Seung (1999). *Learning the parts of objects with nonnegative matrix factorization*. Nature 401: 788-791.
- [3] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. *LIBLINEAR: A library for large linear classification*, Journal of Machine Learning Research, 1871-1874, 2008.
- [4] K. Koh, S.J. Kim, and S. Boyd *l1-logreg: A large-scale solver for l1-regularized logistic regression problems*.
- [5] F. Rosenblatt (1958). *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. Psychological Review, 65, 386-408.
- [6] K. Crammer, O. Dekel, S. Shalev-Shwartz and Y. Singer (2006). *Online Passive-Aggressive Algorithms*. Journal of Machine Learning Research, 7, 551-585.
- [7] K. Crammer, M. Dredze, and F. Pereira (2009). *Exact Convex Confidence-Weighted Learning*. Advances in Neural Information Processing Systems 21 (pp. 345-352).
- [8] M. Dredze, K. Crammer and F. Pereira (2008). *Confidence-Weighted Linear Classification*. Proceedings of the International Conference on Machine Learning (ICML) (pp. 264-271). Helsinki, Finland: Omnipress.
- [9] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker (2009) *Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs*. Proceedings of the ACM SIGKDD Conference, pages 1245-1253, Paris, France.
- [10] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker. (2009) *Identifying Suspicious URLs: An Application of Large-Scale Online Learning*. Proceedings of the

International Conference on Machine Learning (ICML), pages 681-688, Montreal, Quebec.

- [11] A. Oliva and A. Torralba (2001) *Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope*. International Journal of Computer Vision 42(3), 145-175.
- [12] D. Lowe (2004) *Distinctive image features from scale-invariant keypoints*. International Journal of Computer Vision, 60, 2, pp.91–110.
- [13] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Felegyhazi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G.M. Voelker, S. Savage (2011) *Click Trajectories: End-to-End Analysis of the Spam Value Chain*.
- [14] S. Sinha, M. Bailey, and F. Jahanian (2008) *Shades of Grey: On the effectiveness of reputation-based blacklists* Proceedings of the International Conference on Malicious and Unwanted Software (Malware) (pp. 57-64). Alexandria, VA.
- [15] Y.M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen and S. King (2006). *Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities*. Network and Distributed System Security (NDSS). San Diego, CA.
- [16] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu (2007). *The Ghost in the Browser Analysis of Web-based Malware*. Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnets (Hot-Bots). Cambridge, MA.
- [17] A. Moshchuk, T. Bragin, S.D. Gribble, and H.M. Levy (2006). *A Crawler-Based Study of Spyware on the Web*. Network and Distributed System Security (NDSS). San Diego, CA.
- [18] I. Fette, N. Sadeh and A. Tomasic. *Learning to Detect Phishing Emails*.
- [19] Y. Zhang, J. Hong and L. Cranor. *CANTINA: A Content-Based Approach to Detecting Phishing Web Sites*
- [20] A. Bergholz, G. Paa, F. Reichartz, S. Strobel and J.H. Chang. *Improved Phishing Detection using Model-Based Features*