TÉCNICO LISBOA

Website Analysis: Finding Fraud Patterns in Web Pages

Tiago António Campos

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Sandra Pereira Gama
Eng.º Eurico José Teodoro Doirado

Examination Committee

Chairperson: Prof. Rui Filipe Fernandes Prada
Supervisor: Prof. Sandra Pereira Gama
Member of the Committee: Prof. João Manuel Brisson Lopes

October 2020

Acknowledgments

If I would thank every person that helped me up to this point in my career, I would probably not have time to finish writing the rest of the document. Since, I have to follow strict guidelines to write the document, I will take the opportunity to use this page to be a bit more like myself and less robot-ish.

Before starting naming special people I want to acknowledge, I want to thank my family for all the support they gave me throughout all these years, even on those days where all I wanted to do was to throw the PC through the window, because there were some annoying bugs (honorable mention to Javascript) and other creatures (yes, I talking about some people I came across) that really got under my skin.

Now, I want to individually mention important people, with no particular order of importance. Teacher Sandra Gama, I am going to start with you. Thank you for all the help you gave me, thanks for always answering my questions and always providing essential feedback to everything I did and a special thank you to protect me against companies that were not that interested in working with us. Thank you Gonçalo Pereira (Phybbit's Team Leader and Senior Software Engineer), for always seeing the half full glass, even when things were not that great and thank you for always taking your time to explain the things to me, sometimes like I was five years old. Thank you Eurico Doirado (Phybbit's CTO) for trusting me to help your company, in the first time you had a student doing the thesis with you; thank you for being the guy that always went straight to the point (so straight to the point that you could scare Gonçalo) and a special thank you for making me think and understand what I did wrong by myself without giving the easy solution. Last but not least, a special thank to Jéssica Veiga. You turned out to be a better friend than I could wish for. We basically did the all Masters and projects together, dealt with some crazy people, that were nothing but stones in our path. You were also the person who convinced me to do the course of Information Visualization, which means, if it were not for you, I would be doing some boring work instead of helping a real company with a real problem. So for all that, a big thank you Jéssica!

I want to give a very special shout out to my main sponsor Mrs. & Mr. Serpa Publisher. They have created one of the best and more powerful English Grammar Checker available called Gui Serpa - 95th Edition. It can detect even the smallest mistake as well as being a really good companion and a reliable source of funny and meaningful moments.

I also want to thank all the participants of the usability evaluation, that were available to do the evaluation in a difficult remotely environment.

To each and every one of you – Thank you.

Abstract

In the past decade, online advertisement fraud has been a growing issue, increasingly harming companies' business. There are several tools in the market that aim at protecting against advertisement fraud and, for this, Phybbit, a Japanese company, created SpiderAF. One of its unique features is the inclusion of a website blacklist that includes fraudulent websites. A website is classified as fraudulent after its web pages attributes and content were analyzed against know fraud patterns. In the ever evolving world of the web and advertisement fraud, new patterns keep emerging and its rapid detection is key to prevent fraud from impacting advertisers budget. We propose the application of Information Visualization techniques in order to create a system where researchers can look at the data from the several web pages and find fraud patterns, providing a more efficient alternative to what is available today. The newly discovered fraud patterns can then be used directly to analyze new websites, contributing to the protection of the digital advertisement ecosystem. We developed the system following an iterative and incremental design, closely with Phybbit. To validate our solution we conducted an usability evaluation and a utility evaluation done by Phybbit's researchers. The system met the proposed objectives and showed promising results on both usability and utility evaluations.

Keywords

Information Visualization; Online advertisement fraud; Website analysis; Fraud Patterns.

Resumo

Na última década, a fraude em publicidade online tem sido um problema crescente, prejudicando cada vez mais os negócios das empresas. Existem várias ferramentas no mercado que visam a proteção contra fraudes publicitárias e, para isso, a Phybbit, empresa japonesa, criou a SpiderAF. Um dos seus recursos é a inclusão de uma blacklist de sites que inclui sites fraudulentos. Um site é classificado como fraudulento depois que os seus atributos e o conteúdo das suas páginas terem sido analisados em relação a padrões conhecidos de fraude. No mundo em constante evolução da web e da fraude publicitária, novos padrões continuam a surgir e a sua deteção rápida é a chave para evitar que a fraude afete o orçamento dos anunciantes. Propomos a aplicação de técnicas de Visualização de Informação com o objetivo de criar um sistema onde investigadores possam consultar os dados das várias páginas web e encontrar padrões de fraude, proporcionando uma alternativa mais eficiente ao que está disponível hoje. Os padrões de fraude recém-descobertos podem então ser usados diretamente para analisar novos sites, contribuindo para a proteção do ecossistema da publicidade digital. Desenvolvemos o sistema seguindo um design iterativo e incremental, em estreita colaboração com a Phybbit. Para validar a nossa solução, realizamos uma avaliação de usabilidade e uma avaliação de utilidade feita por investigadores da Phybbit. O sistema cumpriu os objetivos propostos e apresentou resultados promissores nas avaliações de usabilidade e utilidade.

Palavras Chave

Visualização de Informação; Fraude em anúncios online; Análise de websites; Padrões de fraude.

Contents

List of Figures

List of Tables

Acronyms

Ad	Advertisement
Ads	Advertisements
BL	Blacklisted
CSV	Comma Separated Values
DB	Database
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
InfoVis	Information Visualization
ISP	Internet Service Provider
NBL	Not Blacklisted
SUS	System Usability Scale
UI	User Interface
URL	Uniform Resource Locator

1

Introduction

Contents

In the year of 2019, Advertisement (Ad) fraud cost an unprecedented estimated total of 34 Billion dollars, 19 Billion dollars of which came from the Asia-Pacific region (APAC) [15]. This is even more impressive considering that some organizations, such as the World Federation of Advertisements, predicts that by the end of 2025 the Ad fraud could become the biggest organized crime, surpassing the drug trade [16].

As one can understand, this is a serious and growing problem that not only affects the organizations that are directly targeted by the criminals but also the industry that depends on the online Ad traffic to provide free content for the general Internet consumer.

There are several companies that are specialized in fighting these criminals through tools that can detect the Ad fraud and protect organizations and/or individuals from getting their business harmed by the perpetrators. Phybbit is one of those companies. Phybbit was founded in 2011 in Tokyo, Japan, and it is specialized in creating, developing and commercializing SpiderAF [1], a tool designed to detect Ad fraud. SpiderAF itself can be divided in distinct services: it analyzes the client's Advertisements (Ads) activity information that contains data such as the number of clicks, timestamps for those clicks, and much more, which is used to detect frauds; beyond that, SpiderAF's clients can contribute to a *Shared Blacklist*[2] that contains, among other things, a list of websites that may show signs of suspicious activity or were blacklisted. This *Shared Blacklist* can then be used by each client to improve their advertisement business. It is important to mention that the analyzed websites are the ones where the client's Ads were placed by the advertisement network.

In some of its processes, SpiderAF analyzes clients' Ads information looking for fraudulent websites. The classification is done by analyzing the collected information (clicks, website content and more). If signs of fraud are found, the website is blacklisted. However, all websites that do not present suspicious activity in the previous stage, also need to be analyzed for potential new types of fraud. A particular case is about detecting inappropriate content often defined by advertisers worried about their brand reputation. Another case is detecting websites that were generated automatically for the sole purpose of doing Ad fraud. The process of analyzing these millions of web pages includes automation and machine learning models for detecting previously known fraud patterns. It also includes an exploration done by a team of researchers to discover new trends and patterns. During this exploration researchers often focus on the website's attributes and its content.

Our work focuses on providing a new exploration approach to discover website fraud patterns previously missed by other approaches.

Information Visualization (InfoVis) systems help people carry out tasks more effectively by constructing a visual representation of the used dataset [17]. InfoVis takes advantage of a high bandwidth communication channel humans have: vision. Being able to transmit information through visual elements,

[1] https://spideraf.com/intl/en
[2] https://spideraf.com/intl/en/shared-blacklist

allows the discovery of new approaches for problems that cannot be fully comprehended and specified by a computer algorithm [17].

All things considered, employing InfoVis techniques in the context of website analysis offers the possibility of finding patterns in fraudulent web pages more effectively, using both visual components as well as data components. Moreover, it provides an efficient approach to increase both the fidelity and size of the *Shared Blacklist*, protecting more clients against advertisement fraud.

1.1 Objective

The main objective of this research is to **study the application of Information Visualization techniques in the website analysis context in order to help find fraud patterns in web pages**.

Having an interface that shows the hidden patterns in the data can be game changing. Considering that until now most of the data is left unchanged in files, no one knows for sure if the used techniques for classification are appropriate, so, to fulfill the objective of this research, an interface that applies the principles and techniques of InfoVis is necessary. Developing an interface that allows the researchers to visualize the websites data, will help them find fraud patterns and, ultimately, provide insights that enable the refinement of the techniques used in their business.

1.2 Document Structure

This document is organized as follows: in Chapter 2 we look at other researches that focused on InfoVis techniques that could be useful to accomplish the goal of our research.

In Chapter 3 we go through the development of our solution, from the first prototype until the end result, explaining each decision we made and the reasons behind it.

In Chapter 4 we first explain our evaluation process and then, the obtained results, alongside a discussion to explain the results.

Lastly, in Chapter 5 we draw the conclusions of the work done and what could be the next steps towards improving our solution.

2

Related Work

Contents

Web page data is often stored in Database (DB) tables. Most of the web pages are dynamic, which means their content is constantly changing (e.g. a news website updates its content every time something relevant happens). This implies that different visits to the same web page, in distinct moments in time, may produce unique data results. Since this context is rather unique and highly depends on how/what data is stored, the related work must focus on techniques that either handle the visualization of files/tables line-by-line or the visualization of web page related data. Taking these requirements into account, two types of visualizations were chosen: log file visualizations and clickstream visualizations.

2.1 Log File Visualizations

Certain anomalies in the system can only be found at a later point in time by manually analyzing log files [18], which size can grow up to millions of entries. One way to ease this problem is to create visualizations that, given a log file, show information that help the analysts to quickly find abnormal patterns in the behavior of the system. Even though most of the subsequent approaches focus on security, techniques such as exposing patterns and processing each entry of a file are desirable in the context of web page analysis.

2.1.1 Seesoft [1]

Seesoft [1] was one of the first tools developed with the intent of showing statistics and relationships of several source code files, at a line level of detail.

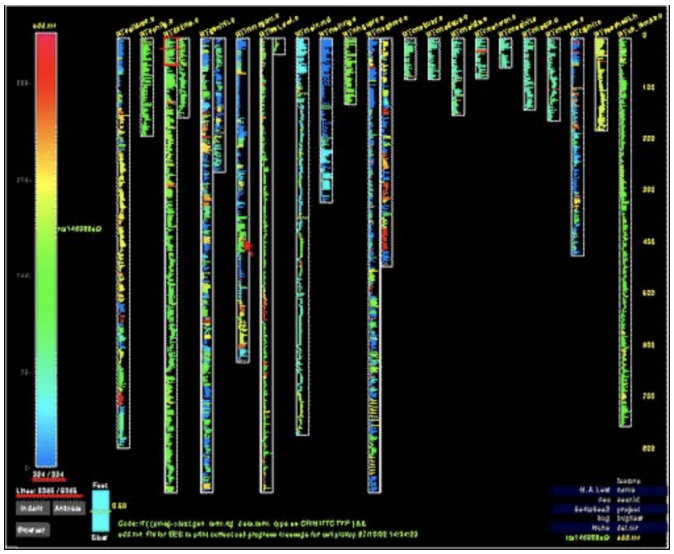


Figure 2.1: Interface of Seesoft [1].

Figure 2.1 shows a general view of Seesoft. Each column encodes a source code file and each line

is a colored row, where the color encodes the last time that line was modified (red to dark blue scale, where red encodes the most recent change and dark blue the oldest change). On the right, the numbers represent the total number of lines each file has (each column is limited to 900 lines of code and if a file has more than 900 lines, a second column is drawn under the same file name).

Seesoft offers interesting interactions for the time: moving the mouse over the color scale will highlight the rows of the source code files that were modified in the corresponding time; hovering a row in a column, will show the new code that originated the change and it is possible to select more than one file and see how the modification in one file affect the others. The ability to see the modification done in the source code files is extremely helpful for new programmers to understand the evolution of the code and which lines may need to be changed. Seesoft is limited to source code analysis but it set the ground for the future work not only in this area but also in other areas such as web server log file visualization.

2.1.2 LogSpider [2]

LogSpider [2] is a security-oriented visualization, which goal is to analyze a single log file with a line level of detail and so, it is based on the work done in Seesoft.

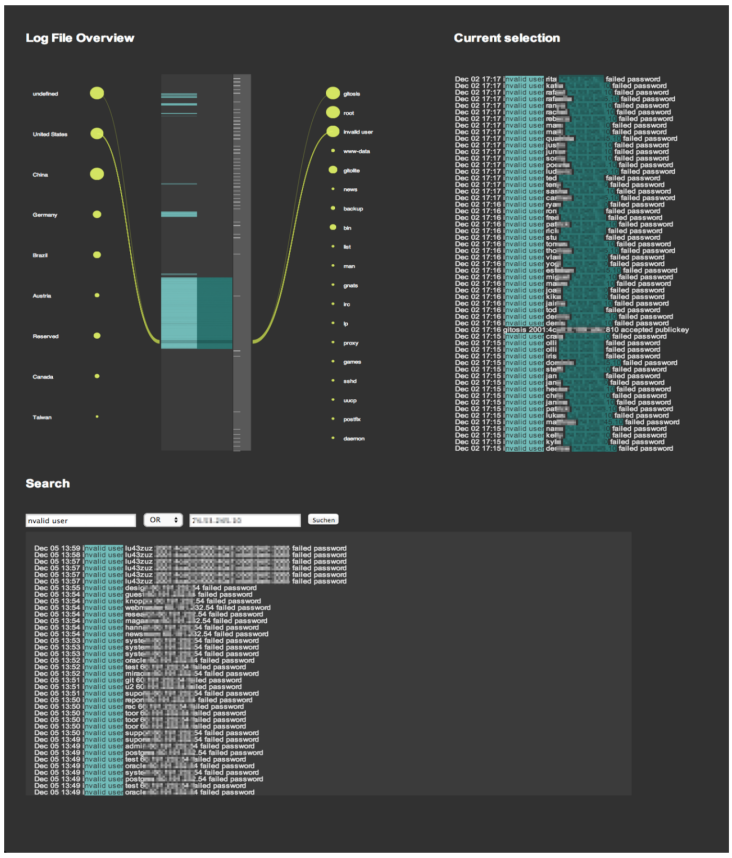


Figure 2.2: Full view of LogSpider [2].

In Figure 2.2, there is an overview of LogSpider. On the left, a timeline is used where new log entries are at the top and older entries are on the bottom. It is possible to select a portion of that timeline using a brushing mechanism. The selected log entries are shown on the right. The green lines show the connection between origin entity and destination entity of a given selected log entry. The size of the green circle encodes the number of times a given entity appears in the selected portion of the log file. In this visualization in particular, the green color has no particular meaning. It is also possible to insert search terms in the search box (it supports AND/OR queries and regular expressions) on the bottom left side of the visualization and the match results will be highlighted using this blue/green color on both the timeline and in the search results.

In addition to interactions (such as hovering and brushing), advanced searching mechanism and the ability to keep the context of the log files in all states of the visualization, LogSpider was developed closely with experts that provided feedback and gave insights of what a useful visualization of network logs should have. Despite all this, LogSpider has several shortcomings. It only supports one log file at a time, which means it is not possible to view multiple log files at the same time and so, more complex attacks cannot be found using this visualization. Although the experts asked the analysts to show the entries of the log files in the search result so they could look at them and see if something was wrong, there is still too much text in this visualization, and it should be possible to somehow present more charts that show some patterns in the flagged log entries. The timeline is limited to a two hours' timespan and the fact that new entries are shown on the top may be counter intuitive to some people.

2.1.3 Pixel Carpet [3]

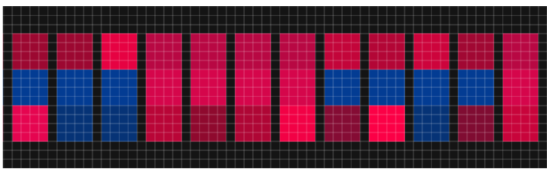


Figure 2.3: Columns and squares are used to encode log file's entries and fields, respectively [3].

Figure 2.4: Full representation of a test log file [3].

Pixel Carpet [3] is an alternative to LogSpider. It offers more granularity in the visualization of the log file and reduces the amount of text in it.

Pixel Carpet uses columns to represent data records (one column = one data record). Figure 2.3 shows an example of several columns. In one column, each colored square encodes (from top to bottom) source country/IP, user name and log message. It is used a colored scale, where red encodes

rarer entries and dark blue encodes more common entries. Figure 2.4 shows a global view of a complete log file (the older records are on the top left; newer records are in the bottom right and the numbers in white indicate the hour of day where the records happened). Pixel Carpet also allows interactions such as tooltips on hovering over columns or filtering options, using the remaining parameters.

The proposed visualization effectively allows rarer entries in the log to be spotted more easily by the analysts (those entries are bright red and stand out when compared with common entries encoded in a dark blue color). Pixel Carpet was developed closely with the team of security analysts that will use the tool. This with the addition of feedback sessions, ensures Pixel Carpet usefulness. A problem with Pixel Carpet is the fact that the color scale is not adequate for log files with a considerable amount of diversified entries (which is the case of, for example, Apache log files). When this happens, the solution is to apply filters and recalculate the algorithm used to build the color scale, which is time costly and not intuitive. Pixel Carpet does not scale well. The number of log entries shown in the visualization is limited to the screen space (about 10 000 entries), which means this solution will not work in large datacenters, where the log files often have millions of entries.

2.1.4 ELVIS [4]

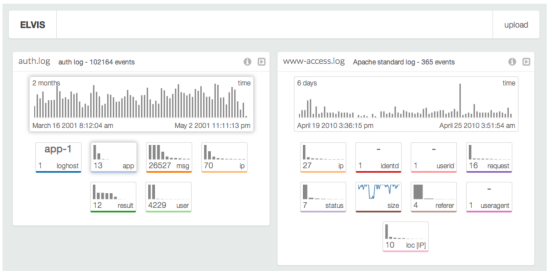


Figure 2.5: Dashboard view of ELVIS [4], with two log files loaded.

Figure 2.6: Automatically generated charts after user selected "IP" [4].

ELVIS [4] adds an important feature to the work done in LogSpider and Pixel Carpet: it allows the visualization of more than one log file at the same time.

ELVIS has a set of preloaded log file formats and, upon receiving the input, it tries to match the file with one of the available formats. If no match is made, the user itself can add a new format and load it into the tool. This feature makes ELVIS compatible with a wide variety of log files.

Figure 2.5 shows ELVIS interface when two log files are loaded into the system. Each file has its summary view. In this view, the first chart in the top displays the event distribution in the file. Then, there are small multiples views of each of the fields present in the file. The colors have no particular meaning; the size of the bars encodes the value of each type of field; the sparkline shows the variation of the packets size through time. When the user clicks on a small view, ELVIS generates charts that are relevant considering the relationship between the field in the selected view and other fields in the file (these relationships are defined in pre-loaded rules. For example, selecting a categorical field, always generate a donut chart). This ELVIS' feature can be seen in Figure 2.6.

ELVIS offers a set of interesting features to the user: summary view that becomes more specific upon selection of a field, with new charts being created automatically; possibility to load multiple log files, which allows the identification of more complex anomalies; ability to process any log file as long as the correct entry format is loaded into the system. However, there are some flaws in this tool: when more than one log file is loaded, the datasets cannot be combined for exploration; the user cannot interact with the charts automatically generated and ELVIS is not fully optimized since the computation time grows exponentially with the size of the dataset.

2.1.5 CORGI [5]

CORGI [5] is the next iteration of ELVIS. CORGI solves one of the main issues of its predecessor: the views of two or more log files can now be combined for exploration.

Figure 2.7: Dashboard view of CORGI [5].

Figure 2.7 shows the dashboard of CORGI. On the left, the timeview panel displays the distribution of events for each imported log file (in this example, three log files were loaded and each one has its

own color). On the middle, field summary view displays a summary chart of each field of the dataset selected in the timeview panel. On the right, the full-size chart view displays the automatically generated view for the fields the analyst selected in the field summary view. In these automatic generated charts, the bar charts have the field encoded in the X-axis, the Y-axis encodes the occurrence number of that field and the color encodes the log file where the data refers to.

CORGI offers filtering options, interactions in the charts and the possibility of combining several fields from the dataset to generate a suitable chart. The main issue with CORGI is scalability. CORGI depends on the web browser performance and, in cases where considerable amounts of log files are loaded, the system may lag and some visualizations may not scale accordingly (for example, if a system gets requests from hundreds of different IP addresses, a bar chart will not work to display all the unique IPs).

2.1.6 Alsaleh et al. [6]

Alsaleh et al. [6] developed a tool that is still security-oriented but this time, it does not support several log files' types. Instead it is built to work as an extension of PHPIDS.

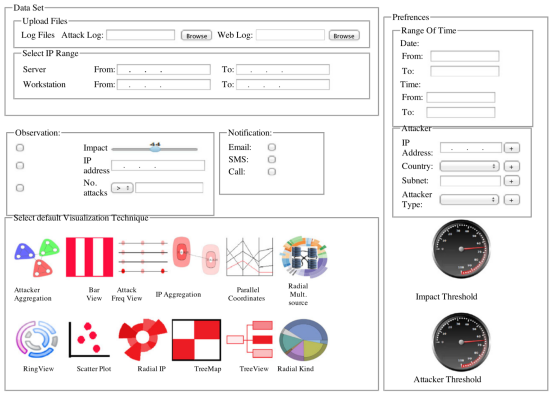


Figure 2.8: Dashboard view of Asaleh et al. [6] extension.

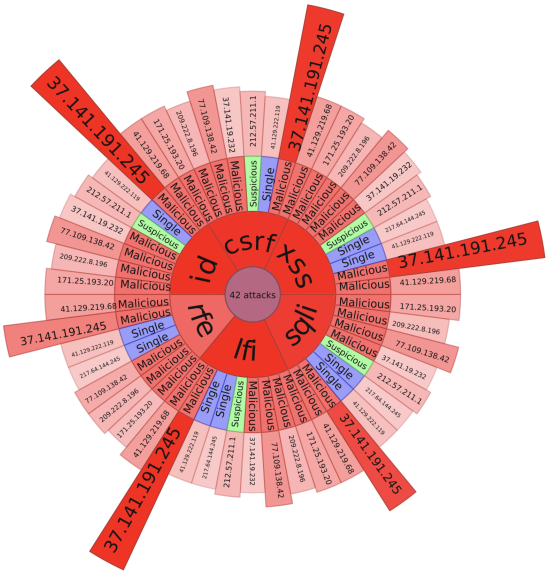


Figure 2.9: Radial view was the most complete in the tests made [6].

A dashboard view can be seen in Figure 2.8. Here, the user has the ability to upload log files and choose parameters for the visualization. Furthermore, the user can choose a visualization from the 10 options available. Alsaleh et al. concluded that the Radial View (Figure 2.9) was the most complete visualization. In the inner ring of the visualization, there are represented the different types of attacks; in the middle ring, the category of the IP (malicious, suspicious or single) is represented with different

colors and in the outer ring, it is represented the attacks' origin IP addresses, the red color scale encodes the total impact of the attack in each node and the size of the segment encodes the number of attacks that were launched by the corresponding IP address.

Alsaleh et al. solution offers a wide variety of visualizations to the user choose from. Each visualization has its own advantages and disadvantages, so it is up to the user to choose which suits best its needs. Each visualization offers some sort of interactivity such as filtering mechanisms, zoom or hovering. However, the user cannot choose more than one visualization at the same time. Radial view has some problems too: the red scale does not have enough steps to allow the user to correctly distinguish impact numbers and the circular aspect of this view implies some kind of cyclical feature, which is not the case of the data used to build this view. Another potential problem of Radial view is comparing the size of segments, but this may be neglected because in most cases security analysts care more about which was the IP that launched more attacks than about how many attacks it launched.

2.1.7 Zhang et al. [7]

Zhang et al. [7] considered that most of the existing tools to analyze log files lacked one important feature: the ability to represent relationships between events and/or between servers. Showing these relationships allows the analysts not only to find the origin of a given anomaly but to also track down what caused it.

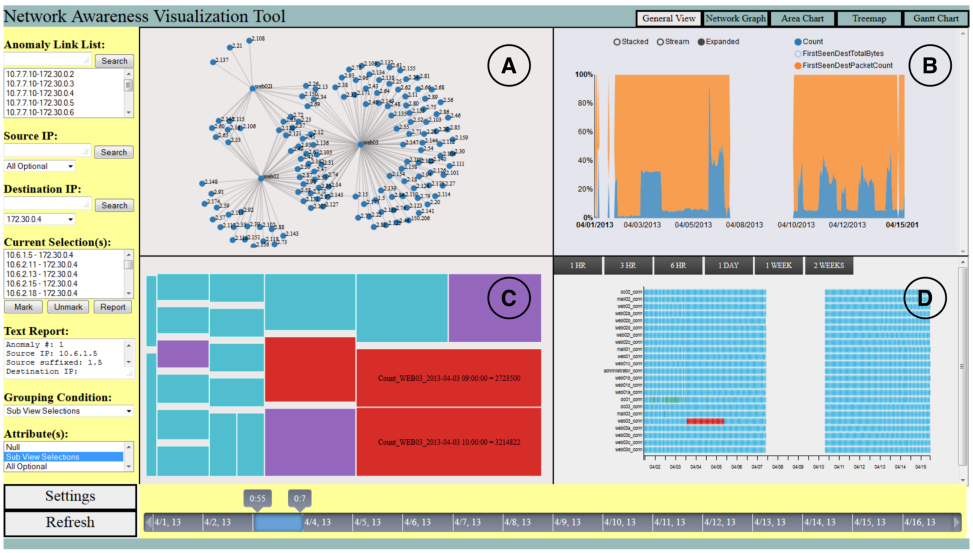


Figure 2.10: Dashboard of Network Awareness Visualization Tool [7].

Figure 2.10 shows the dashboard of the visualization. Figure 2.10-A shows a Network Graph used to represent the network flow between two machines (i.e. source IP and destination IP). In this graph, the central node is the machine the analysts want to investigate (i.e. source IP), each node around the

central point is the destination IP and the edge width encodes a chosen attribute (e.g. number of packets sent). In Figure 2.10-B, stacked area charts are used to represent attributes of a server machine. In those charts, each color encodes a different attribute; X-axis encodes the date of given record and the Y-axis encodes the value of each attribute. Figure 2.10-C shows a Treemap, used to represent the records count grouped by a time unit from a chosen server machine. In this case, both the color scale and the area encode the sum of the records (red being the higher sum and blue the lower sum). Figure 2.10-D shows a Gantt chart, used to represent the connection status of each server. Here, the color encodes the server status; the X-axis encodes the date of a given record and the Y-axis encodes the different connected servers.

The created dashboard allowed interactions such as filtering attributes from the records and brushing in the timeline at the bottom of the dashboard. All views are connected, which means a change in one of them will propagate to all others. There is also the possibility of visualizing each chart individually. However, the solution is not scalable. If one looked closely at the charts, they would notice that only three server machines are present. In cases where the anomaly happens in more than one server, charts like Treemap and Network graph can grow uncontrollably and lose expressiveness, which means, finding the cause of the anomaly may be a difficult task to be done exclusively with these charts.

2.1.8 LongLine [8]

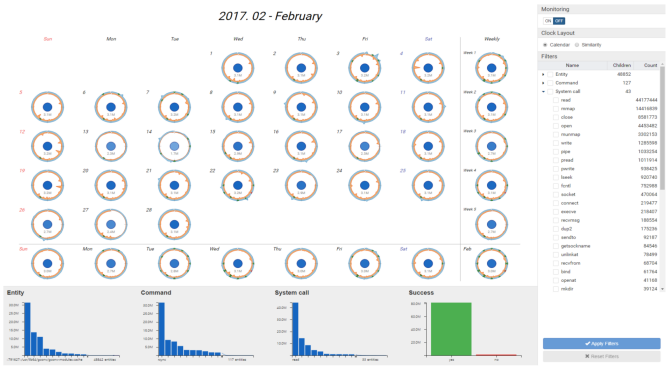


Figure 2.11: Calendar view of LongLine [8].

LongLine [8] focuses on a different kind of system log files: audit logs. Audit logs differ from network logs (used in all previous works) because they register more primitive operating system events: system calls. Audit log are also large-scale logs. Since system calls happen all the time, each log will certainly have millions of entries. LongLine was developed to tackle this problem.

Figure 2.11 shows a dashboard of LongLine, with the calendar view selected. For each day of the month, it has a circular chart named BiClock. In each BiClock, the color encodes the 12 hours timespan (orange for daytime and blue for nighttime), the Y-axis encodes the number of entries for that specific

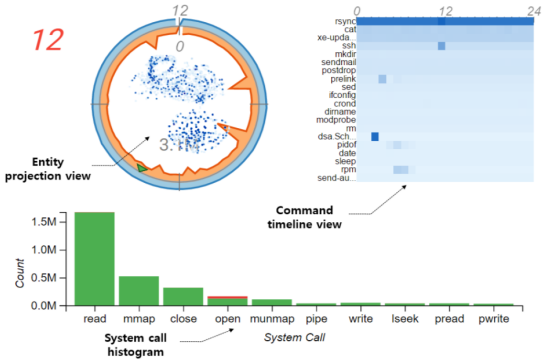


Figure 2.12: View after user zooms in a given BiClock [8].

time, the triangle represents an important event and its color encodes the status of that event. If the user zooms in a specific day, it will show a view close to Figure 2.12. A heatmap appears, showing the top-20 most called commands, where the color encodes the number of times each one was called. It is also possible to compare two days. The BiClocks are transformed into area charts, with the same properties as the BiClocks. On the bottom of the dashboard, there are histograms showing global statistics for each event type.

LongLine offers detailed views of the audit logs with interactions, filtering mechanisms and with well-chosen colors. On the other hand, LongLine does not allow to distinguish which user made the system calls and the calendar view is limited to one month, making it impossible to compare days from different months.

2.2 Clickstream Visualizations

Clickstreams are sequences of timestamped events generated by the user [14]. Understanding the user behavior in a company's website is essential for improving the business model and generate revenue [19]. To analyze the generated data and to understand the several relationships that can be established it is important to have visualizations capable of providing such insights. MOOC (Massive Open Online Courses) are a particular case where analyzing the clickstream can be useful. MOOC have attracted the attention of millions of users in the last few years [20] and so, it raised the necessity of having tools to help the people involved improving the available courses. Even though MOOC analysis does not have the same context as the one found in this research, techniques such as visualizing paths taken by a user and the establishment of relationships between the content of a website and its visits can be useful in the target context.

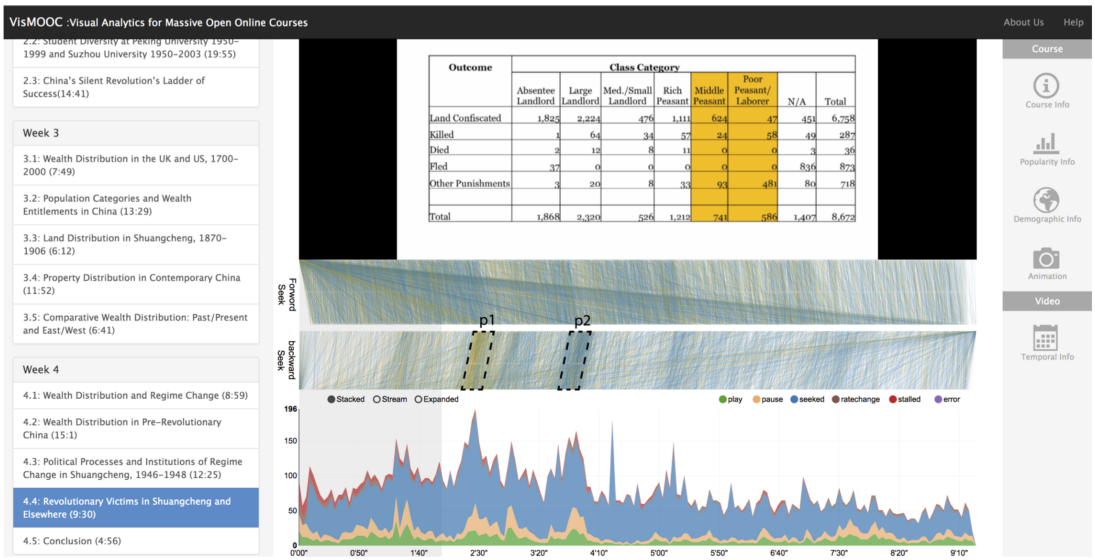


Figure 2.13: Dashboard of VisMOOC [9].

2.2.1 VisMOOC [9]

VisMOOC [9] is a tool developed with the intent of providing insights about the students' clicking routines in the course videos. Teachers can then use the insights to improve the course.

Figure 2.13 shows a view of the dashboard the teachers have access to. In the middle, there is the video, the seek graph (parallel coordinates) and the event graph (stacked area chart); on the left there is a list view with the several videos of the course and on the right, there are several options the teacher can choose to visualize. In the seek graph, the horizontal lines encode the starting and ending position of seeks events; a line is drawn connecting the start and end points of a seek; the blue color encodes seek events in the first view of the video and the orange color encodes seek events in the next views. In the stacked area chart, the X-axis encodes the length of the video; the Y-axis encodes the number of events and the color encodes the type of events.

VisMOOC offers a set of visualizations that show the statistics of the course. Each view is interactive. These features and, the fact that the video is always present in the dashboard, allows the teacher to do a close analysis of the course and see if changes are needed to improve it. On the negative, the seek graph has a pretty obvious scalability problem: if numerous seek events happen in a certain area, the set of lines will produce a darker area and hide other seek events close to it. One can also argue that using a stacked area chart to show the number of so many types of events may not be the best option.

2.2.2 PeakVizor [10]

PeakVizor [10] shares some views with VisMOOC, but instead of having them hidden behind menus and selections, it tries to display all the available visualizations in the same dashboard view.

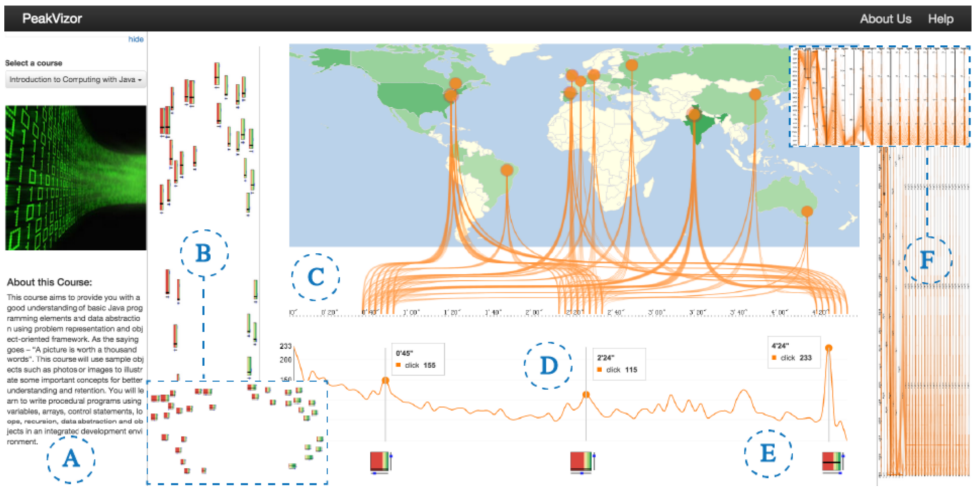


Figure 2.14: Full view of PeakVizor [10].

Figure 2.14 shows a general view of PeakVizor. The first elements are the Glyphs Figure 2.14-B. These Glyphs represent peaks of video activity (stop, pause, repeat and others). They are divided in six bars, one for each final student grade interval, with colors encoding those same grade intervals. Figure 2.14-C shows the timeline of the video of the course. Connected to this timeline, there are lines, each one encoding a student click and the origin of the line is the corresponding student country (in the choropleth, the color encodes the number of students for each country). Below the video timeline, there is a line chart. Here, the X-axis encodes the time of the video and the Y-axis encodes the number of clicks (when a peak happens, a glyph appears underneath it). On the right side Figure 2.14-F, a parallel coordinates show correlations between several attributes: country, grades, loyalty, delay, dropout time, video activeness, review activeness and forum activeness, with each line encoding a student.

PeakVizor succeeds in showing correlations, detailed statistics for a given course and still provides interactions that help the teacher understand how the course is going. However, it has a huge problem that can be spotted immediately: all views become unusable when the number of students grow. Even with semantic zoom (PeakVizor offers it), it would be extremely hard to, for example, understand the correlation view (parallel coordinates). Another issue with PeakVizor is the fact that teachers cannot compare their courses with someone else's courses.

2.2.3 DropoutSeer [11]

DropoutSeer's [11] main goal is to visualize one of the most critical component of MOOC: students' dropout rates. Both VisMOOC and PeakVizor show general statistics of the course but they failed in providing more detailed information that could help teachers understand when and why a student drops out.

Figure 2.15: Overview of DropoutSeer [11].

Figure 2.15 shows the DropoutSeer system. Figure 2.15-A shows the prediction model results, using clusters, where each dot encodes a student. Based on the cluster results, the timeline view Figure 2.15-B is constructed as follows: each subgroup of students is encoded in the vertical axis, whereas the horizontal axis encodes the temporal information i.e. student actions throughout the course. To show the average clickstream and assignments, a glyph is used. In this glyph, the outer radius encodes the average number of videos watched by students of that subgroup, the inner radius encodes the standard deviation of the same subgroup, the length of the arc encodes the total number of actions and the colors encodes the type of action done by the students. On the far right of Figure 2.15, there are statistics from the course itself and a bar chart with the results of the prediction model. Figure 2.16 shows the flow view. In the bar chart in Figure 2.16-a, each bar encodes a student subgroup and its length encodes the number of students that made a post in the forum. On the right, the vertical timeline encodes the timeline of the course, the grouped bar chart represents the total number of posts made in that day, the color scale encodes the type of post made (question, discussion or other) and each line encodes the flow of a student from the subgroup until when it makes a post in the forum.

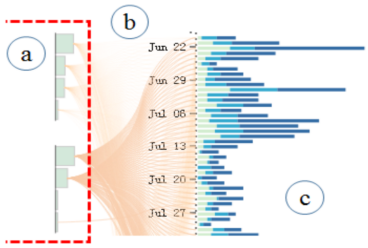


Figure 2.16: Flow view of DropoutSeer [11].

DropoutSeer offers a unique view of dropout rates and its causes. It has some level of granularity, interactions and filtering mechanisms that enhance the user experience. But it cannot solve the problem

other MOOC visualizations have: scalability. The glyphs do not scale well, and the flow view becomes unreadable with large amounts of lines. DropoutSeer also has other problems: the timeline view only supports weeks, so if there are courses that have daily assignments, the visualization will not work, and it is not possible to establish relationships between different courses and students' dropout rates.

2.2.4 Liu et al. [12]

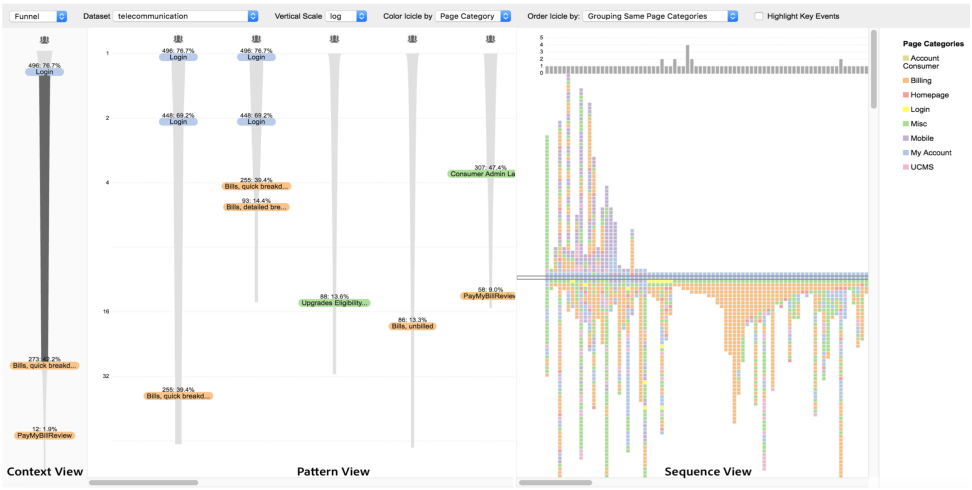


Figure 2.17: General view of the Liu et al.'s interface [12].

Liu et al. [12] focus on a wider variety of clickstream data. If previous researches' main goal was to help teachers improving their online courses, Liu et al. aims to give analysts a tool that helps them finding the most common paths taken by the users when they visit a website.

Figure 2.17 shows the interface design of this tool. The interface is divided in three section: the first is the context section, which purpose is to highlight the current selection; then, it has a pattern view, where the extracted patterns are represented in a funnel manner; lastly, the sequence view provides detailed information for the select sequence. Taking a closer look to the sequence view (Figure 2.18), each square encodes an event from a sequence laid out from top to bottom, the color encodes the category of the event and the grey squares encodes a metric chosen by the user (e.g. number of visitors in that path). Hovering over a square displays event information. In the pattern view, the vertical scale is used to encode summary statistics (e.g. average number of clicks that takes a user to reach a given page).

This tool allows the analyst to view the most common sequences of events in a clean way, with high level of detail and offers a pipeline consisting of pattern mining, pattern pruning and coordinated visualizations exploration. However, it suffers from a problem that is common among this type of tools: it lacks scalability. For large datasets and high number of clicks, the interface will require a considerable

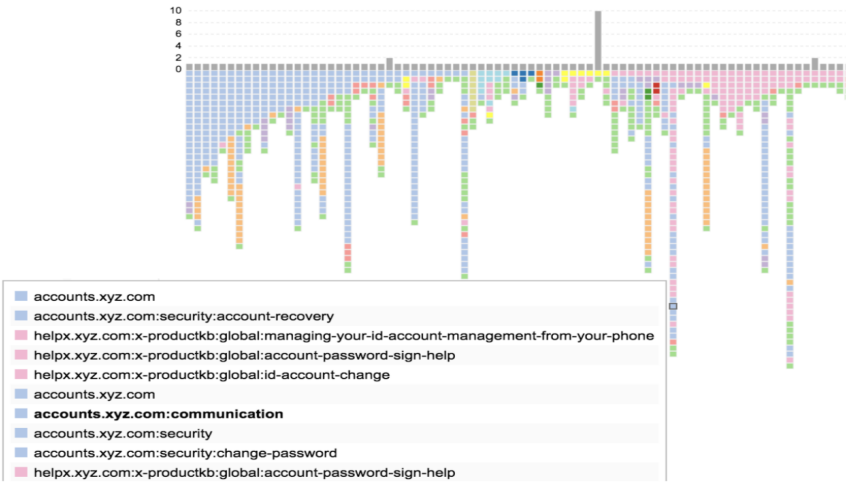


Figure 2.18: Sequence view with a square hovered on [12].

amount of navigation with the horizontal scroll bar and the computation time of the three stages' pipeline will increase exponentially.

2.2.5 MatrixWeave [13]

MatrixWeave's [13] purpose is to solve a common problem to all previous clickstream visualizations: the inability of comparing data from different courses, days of the week, months or years.

Figure 2.19: Dashboard view of MatrixWeave [13].

The datasets used in MatrixWeave were from a large company's website. Only the first 6 steps of the 1000 most common event sequences on two different days were considered.

Figure 2.19 shows the dashboard of MatrixWeave for the aforementioned datasets. The numbers indicate the step number. Beneath the numbers, there is a stack of rectangles. Each one, encodes a node step, the size encodes the volume of traffic passing in that node and the color encodes the difference in traffic of a node in the two datasets. Inside each matrix, there are colored squares that encode links between nodes, the inner square size represents the average traffic volume in the link and the color of the square encodes the difference in traffic volume. Hovering over a square, highlights the line of the matrix and it is possible to show the path when more than one object is selected.

The possibility of comparing two large datasets from different clickstreams is MatrixWeave's biggest advantage. It offers interactions, filtering and sorting options to facilitate the navigation between matrices. However, the learning curve of MatrixWeave is steep and it requires large datasets, otherwise the matrices will have several blank spaces making it lose expressiveness.

2.2.6 Wang et al. [14]

Wang et al. [14] developed a tool that visualizes the clickstream data in a completely different way. If previous researches use several charts to display the information, Wang et al. only uses clusters to show similar user behavior. This is done with a unsupervised clustering algorithm developed by the authors.

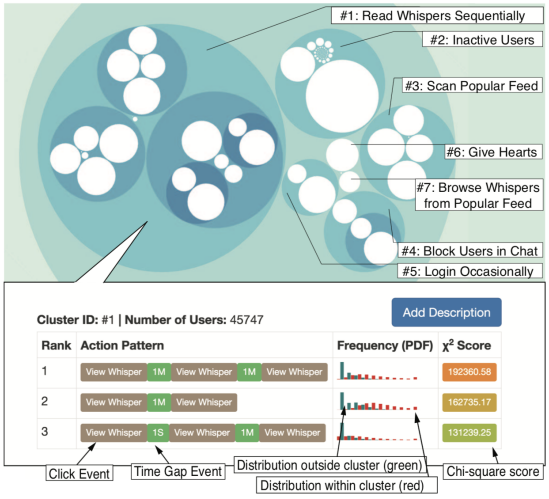


Figure 2.20: *Whisper*'s user behavioral clusters [14].

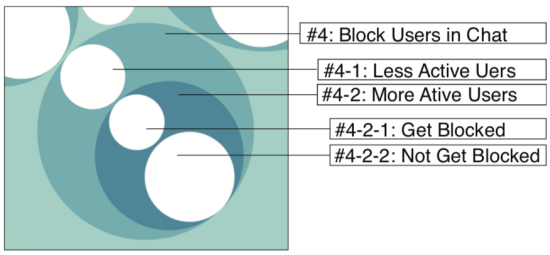


Figure 2.21: Example of cluster #4 of *Whisper* [14].

There were two datasets used to build this tool: clickstreams from 100K users in an Android application called *Whisper* (more than 125 million total events) and clickstreams from 16K users of *Renren*, the Chinese social network (more than 7 million total events).

Figure 2.20 shows *Whisper*'s behavioral clusters. Each parent circle encodes the user behavior and

each child circle encodes some user sub-behavior. The action pattern is the result of the feature pruning algorithm used. The frequency (PDF) divergent bar chart encodes how frequently each action appears inside a cluster, the red bar encodes the distribution within the cluster and the green bars encodes the distribution outside the cluster, which means the more divergent the two distributions are, the better is the unsupervised cluster algorithm used. Figure 2.21 shows an example of a behavioral cluster and its sub-clusters.

Wang et al.'s algorithm has higher precision and recall than other cluster algorithms for the same datasets. For its purpose, the tool gives helpful insights about how users behave in online services. In an Information Visualization point of view, this tool have some shortcomings: becomes unreadable when there are a large amount of child clusters inside a parent cluster; there are few interactions and filtering options and only one visualization is available (packed circles), which limits the data exploration.

2.3 Discussion

	Multiple Views	Connected Views	Scalable	Exploration	B&W Test
Seesoft [1]	✗	✗	✗	✓	✗
LogSpider [2]	✗	✗	✓	✗	✓
Pixel Carpet [3]	✗	✗	✗	✓	✓
ELVIS [4]	✓	✗	✗	✗	✓
CORGI [5]	✓	✓	✗	✓	✓
Alsaleh et al. [6]	✓	✗	✗	✓	✓
Zhang et al. [7]	✓	✓	✗	✓	✓
LongLine [8]	✓	✓	✗	✓	✗
VisMOOC [9]	✓	✓	✗	✓	✗
PeakVizor [10]	✓	✓	✗	✓	✓
DropoutSeer [11]	✓	✓	✗	✓	✓
Liu et al. [12]	✓	✓	✗	✓	✗
MatrixWeave [13]	✗	✗	✓	✓	✓
Wang et al. [14]	✗	✗	✗	✗	✓

Table 2.1: Comparative evaluation of the related work.

Ever since Seesoft [1] was developed in 1992, a vast amount of work has been done with respect to the analysis of files line-by-line. Throughout the years, visualizations have become more complex and now they have a tendency to offer more options to the user with more sophisticated interactions. Scalability is a common concern in this type of visualization. Seesoft (1992) supported a total of 50 000 lines, while LongLine (2017) [8] needs to support over 100 millions of lines (with some limitations to be discussed later) since log files have grown in size with the increasingly more complex information systems. It is also relevant to evaluate the used colors. However, since the color perception can be subjective [21], a more scientific procedure to evaluate the colors and other elements of the visualization is needed. Black and White Test is a simple procedure where all the images from the different approaches are converted into black and white. If the patterns, categories and other elements of the visualization are still distinguishable, then the test was passed. Otherwise, it is considered that the visualization failed the test.

Table 2.1 shows a comparative evaluation of all related work. Most of the approaches offer multiple views for the user to explore, but not all of them are connected. Both ELVIS [4] and Asaleh et al.'s [6] tool have more than one visualization in the dashboard but even if they show some relationships the user's interactions are limited to the current selected visualization and do not propagate to the others. In terms of scalability, only LogSpider [2] and MatrixWeave [13] are capable of handling large datasets with some shortcomings: LogSpider is scalable because uses one timeline view to encode all log entries and displays the results of the search queries in plain text; MatrixWeave can encode thousands of paths taken by the users in a website but those paths need to be limited (e.g. 6 steps, as used in the evaluation of the tool), otherwise the screen is not wide enough to show the several matrices and the user has to use scroll bars to see all of them. LongLine although it supports files with million of entries, it can only show data from one month at the time and so, it is not scalable if the user wants to view more than one month at the same time. With regards to exploration, LogSpider does not offer interactions, only filtering options; ELVIS does not allow the user to interact with the automatically generated visualizations; Wang et al.'s [14] approach allows the user to select the cluster to view and after that no more interactions are available. Seesoft, LongLine, VisMOOC [9] and Liu et al. [12] all failed the black and white test, which means some of the original expressiveness was lost when the images were stripped from the original colors.

Overall, CORGI [5], Zhang et al.'s [7] approach, PeakVizor [10] and DropoutSeer [11] are the most complete tools, but none of them can scale when the dataset has millions of entries and still encode high granularity of information. On the other hand, LogSpider and MatrixWeave scale effectively but they offer limited visualization options and only work in specific scenarios.

3

System Development

Contents

As stated in Chapter 1, the main goal of this research is to **study the application of Information Visualization techniques in the website analysis context, in order to help find fraud patterns in web pages**. In Chapter 2 we saw some approaches used to solve sub-problems related to this research. The next step is to develop a system that takes advantage of InfoVis techniques in order to address the problem: finding fraud patterns in web pages.

In this chapter we will describe all the prototypes that led to the final version as well as the reasons why they were developed the way they were and why they needed improvement (informal evaluation made alongside Phybbit).

3.1 System Requirements

Before one starts developing any system, it is important to define its functional requirements in order to produce a solution that achieves a desirable result.

In this research, the requirements were defined closely with Phybbit. In our meetings, we debated what the system should support immediately (high priority/main features) and some features that would be interesting to have but were low priority. These low priority features are described in detail in Chapter 5. The functional requirements we defined as the main functions of the system are detailed following this paragraph.

1. Visualize the data distribution in a given set of web pages: The system should provide a view to examine how a given indicator (for example, links) behaves inside a set of web pages.

2. Discover fraud patterns (if they exist): The system should allow the discovery of fraud patterns by analyzing different indicators and establish a set of rules that may indicate the existence of suspicious activity inside a web page or set of web pages (for example, low content size and low image count).

3. Verify if a web page has fraudulent activity: Subsequently, the system should support a process to verify if the web pages that belong to a discovered fraud pattern have, indeed, fraudulent activity.

4. Save fraud patterns and fraudulent web pages: The solution should grant the user the possibility to save fraud patterns so they can be used again in case new web pages need to be analyzed, and to save fraudulent web pages, so they can be inserted in a database that contains all web pages found as fraudulent.

3.1.1 Tasks and Questions

Defining the tasks and respective questions, is a crucial step in crafting the visualization [17]. The tasks and questions must be based on the system requirements but more focused on the visualization and

usage of the system itself. As before, the following tasks and questions were defined after consultation with Phybbit and were approved by them.

Task 1: Discover suspicious activity that may indicate that a web page is fraudulent.

Example Question 1: How many Not Blacklisted (NBL) web pages, have no internal links, no external links, no images and content size less or equal than 1000 bytes?

Example Question 2: How many NBL web pages have more than 1000 Word Press (WP) references?

Task 2: Explore web pages in different suspicious patterns and classify them as a suspect of fraudulent activity.

Example Question 1: Given all NBL web pages with no internal links, no external links, no images and content size less or equal than 1000 bytes, how many of them actually appear empty when previewing them?

Example Question 2: How many web pages appear in the subset of web pages with no internal links, no external links, no images and content size less or equal than 1000 bytes and in the subset of web pages with no images and content size less or equal than 1000 bytes?

Task 3: Store/Load patterns of fraudulent activity and/or web pages classified as suspected of being fraudulent.

Example Question 1: What is the percentage of web pages marked as Blacklisted (BL) that have no internal links, no external links, no images and content size less or equal than 1000 bytes?

Example Question 2: How many web pages (union of BL with NBL) have no images and content size less or equal to 1000 bytes?

The relationships between requirements and tasks are characterized as: Task 1 covers the first and second requirements of this research; Task 2 is based on requirement three and allows not only the verification of suspicious activity in a web page but also checking if a given web page appears in more than one fraud pattern; lastly, Task 3 directly covers the forth and last requirement of this research.

We specified the functional requirements of this research and defined tasks and questions from a InfoVis perspective. In the remaining sections of this chapter, we will explain every prototype produced in a chronologically fashion.

3.2 First Prototype

In the previous stage of this research, a concept of a system was proposed as the solution for the problem we were trying to solve (Figure 3.1). When the previous stage ended, we had a meeting with

Figure 3.1: Proposed Solution of this research dissertation project.

Phybbit where we discussed what should be improved in the concept before starting to implement it. Two main conclusions of the mentioned meeting were drawn: there was a lot of information in the starting page, which could lead to the user being confused and not know where to look at, and some information about web pages was not the most relevant to finding fraud patterns.

With the aforementioned conclusions, we understood that we needed to do a deeper data analysis in order to choose which were the attributes that were the most relevant in our context. The deep data analysis is fully described and explained in the following section.

3.2.1 Data (Re)Analysis

Before starting the data analysis itself, it is important to specify some concepts to better understand what will described in this section:

1. Web Host: Internet service that allows an entity to make its website accessible via World Wide Web [22].

2. Web Page: Collection of information provided by the website and accessible by a user through a web browser [23].

3. Blacklisted Host/Web Page: Phybbit considers blacklisted a host or web page if it shows evidence of fraudulent activity.

4. Not Blacklisted Host/Web Page: Phybbit considers not blacklisted a host or web page which contents were not yet analyzed.

5. Fraudulent Activity: Phybbit's fraudulent activity definition is directly related to the fact that a web page is qualified to include Ads or not. This means that if a web page has something wrong (for example, no content or illicit content) or if it is against guidelines imposed by Phybbit's clients (for example, Phybbit's clients do not want their Ads in web pages with adult content), that web page is classified, internally, as fraudulent.

Based on the concepts described, we realized that there are two levels of granularity in the available data: data related to the hosts (i.e. indicators/attributes such as country, Internet Service Provider (ISP) and more) and data related to the web pages (i.e. indicators/attributes such as number of links, content size and more). In Phybbit, the data is collected through scheduled crawls and it is stored in DB tables. The first step of our data analysis was to create new helper tables that contained the data divided by granularity and by classification (either BL or NBL). So, there were a total of four groups of tables:

1. **Tables of BL hosts**
2. **Tables of NBL hosts**
3. **Tables of BL web pages**
4. **Tables of NBL web pages**

Upon completion of the first step, we ended up with the summary results shown on Table 3.1. It is important to mention that sometimes a crawl to a web page is not successful. The failure of a crawl generates a stream of *NULL* values for the data the crawler tried to collect. In the context of finding fraud patterns, it does not make sense to consider those web pages with *NULL* values for every entry since we have no information for that web page and so, we cannot use it as part of a pattern. This means that in Table 3.1 the total amount of web pages only includes web pages that have at least one successful crawl (i.e. there was some information collected by the crawler). Another important aspect to mention is that Phybbit first classifies a web page as BL and then classifies the host for the fraudulent web page as BL, which means that a host can have more than one web page classified as BL, explaining why there are a total of 5258 BL blacklisted hosts and 73838 BL web pages.

	BL	NBL
Total Hosts	5258	144108
Total Web Pages	73838	13633129

Table 3.1: Total of hosts and web pages per classification.

At this point, we had the data properly divided. The next step was to choose which attributes were the most important in the context of finding fraud patterns. To evaluate the importance of each host, or web page attribute, the following criteria was used:

1. **Accuracy of the attribute:** How reliable is the value collected for that attribute?

2. Variability inside a set [1]**:** How many different values that attribute has in the group of BL/NBL hosts and/or web pages?

3. Variability between groups [1]**:** Does the attribute have a similar distribution in both BL and NBL set of hosts and/or web pages?

4. Theoretical usefulness to find fraud patterns: Considering the characteristics of a given attribute, how useful it is to define suspicious behavior?

For the first working prototype, we started from the concept developed in the previous stage of this research, then we looked at the attributes chosen for that concept and at the attributes available in the newly organized helper tables and we evaluated the importance of each one.

3.2.2 Working Prototype

The goal of this prototype was to improve the concept of the previous stage of this research by reducing the number of attributes shown and to separate the granularity of the data, i.e. make a distinction between attributes for hosts and attributes for web pages. By Analyzing the Figure 3.1, we can observe a total of eight different attributes selected. The importance of each one was evaluated using the criteria defined in Section 3.2.1:

1. Geographical Distribution: Obtained by analyzing the routing done when accessing a given host. Since the routing may change depending on the router configuration, this attribute can be inaccurate, and so, it should not be used alone to classify as BL or not a given host. However, it may be interesting to use geographical information to complement other indicators.

2. Domain Name System (DNS) Registrant: This attribute only specifies who registered the DNS and offers no other information. The idea behind using this attribute was that if a host has one BL website then it could have more fraudulent web pages. But, since a host is considered BL if it has at least one fraudulent website, it is not necessary to look through other websites from the same host. This attribute was not used anymore.

3. Internet Protocol: Since Hypertext Transfer Protocol (HTTP) protocol does not have a security layer, it could be the preference of BL web pages. However, the data analysis shows that NBL web pages show the same HTTP/Hypertext Transfer Protocol Secure (HTTPS) ratio as BL web pages set. This means that the Internet Protocol cannot be used alone to classify a web page. However, it may be useful in conjunction with other indicators.

4. Response Status Code: This attribute is especially useful when the status code indicates client error (4xx) or server error (5xx), because it means the web page we were trying to visit is either down

[1] Cannot be fully described to protect the privacy of Phybbit's data.

or does not exist. If that is the case we do not want to include Ads in it, which means the web page should be classified as BL.

5. Internet Protocol - Ports: The vast majority of web pages (both BL and NBL) use either port 80 (for HTTP) or port 443 (for HTTPS). Data analysis show us that the percentage of web pages that use other ports is below 0.01%, which means this attribute cannot be used to classify web pages as fraudulent and it does not show enough variability to be used in combination with other attributes. This attribute was not used.

6. Last Content Update: This attribute indicated when the last content update occurred on a web page. Data analysis showed that is was very accurate. However, it presented two major drawbacks: only 40% of web pages had this value different from *NULL*; considering the content of a web page as being old greatly depends on the content itself: if the web page is from an online newspaper and is not updated in more than one day, the content may be considered old; if the web page belongs to an online encyclopedia, its content may not be updated in more than one year and still be considered up to date. These drawbacks are problematic enough to lead to the exclusion of this attribute.

7. Links relationships: In the concept, there were three included cases: where the number of internal links is the same as the number of external links; where the number of internal links is greater than external links or where the number of internal links is less than the number of external links. After the data analysis, we realized that these case are not relevant since they do not hint at anything suspicious or anything legitimate. However, there are other three cases with greater importance: number of internal links is zero and number of external links is greater or equal to one; number of external links is zero; and number of internal links is greater or equal to one and both types of links are zero. The reasoning is that if a web page lacks one or more types of links it may work as a dead end (if it has no external links, the user cannot go to an outside web page), as a web page that only redirects to external web pages (if it has no internal links) or as a highly suspicious static page if as no links whatsoever.

8. Existence of ads.txt [24]: Ads.txt is a standard used to by a company to list other companies that are authorized to sell their products and/or services [24]. The presence or absence of this text file by itself does not indicate if a host has fraudulent activity or not. Only by analyzing the content of the file is it possible to know if something is suspicious or not. However, this analysis is beyond the scope of this research, and so, this indicator was not used anymore in this research.

To conclude: from the original system concept, only four of the eight indicators were still being considered relevant after the data analysis and were included in the first working prototype. The data analysis also showed some indicators that have the potential to be useful to find fraud patterns:

1. Content Size (in bytes): The content size of web pages gives us a lot of information about a web page, especially when we consider small content size: if a web page has small content size, it may

indicate that the web page only has text on it which may be, for example, the default page of a server application such as Apache HTTP Server[2](we do not want to place Ads in this type of web pages). On the other end of the spectrum, a sizable web page may indicate the presence of a lot of videos and/or images, which may indicate the web page has, for example, adult content.

2. Header Content Type: The content type defined in a HTTP header usually indicates the type of content that the web page loads. This is useful because we do not want to place Ads on web pages that only load a media file (such as an image, video and others), so, it seems a reasonable criteria to classify a web page as BL.

3. References: As mentioned in Section 3.2.1, one of the criteria used by Phybbit to consider that a web page has fraudulent activity is to not place Ads in web pages that go against clients' guidelines. Phybbit's clients do not want their Ads in two types of websites: blogs and adult websites. Phybbit takes this into consideration and the crawler they developed counts the number of pornographic references and the number of Word Press[3]references in each visited web page. This means that if a web page has a great number of at least one these two types of references, it should be classified as BL.

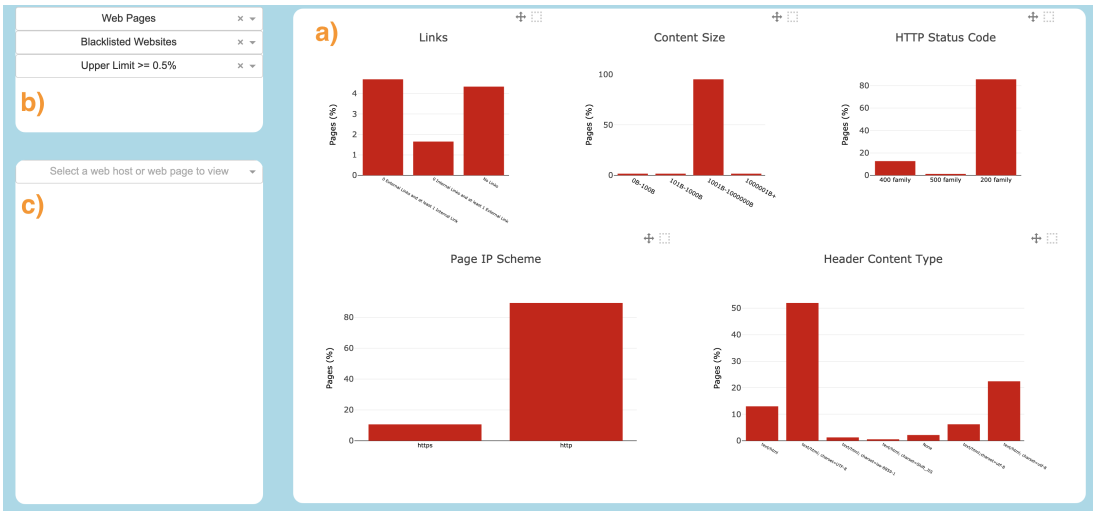


Figure 3.2: Landing page of the first prototype.

At this point, we had all of the data selected and it was time to start developing the prototype itself. The first prototype was developed with Python 3[4] and *Plotly Dash*[5]. It followed a simple structure, where we used Python to fetch and format the data from the local DB were all the tables were stored and *Plotly Dash* to draw the charts and the User Interface (UI). Figure 3.2 shows the landing page of the first working prototype. The general idea of how to use this system was to select one or more attributes we

[2] https://httpd.apache.org/
[3] https://wordpress.com/
[4] https://www.python.org/download/releases/3.0/
[5] https://dash.plotly.com/

found suspicious, then, on the left side of the UI, a list with all the web pages or hosts with the selected attributes (depending on the set selected using the dropdown menu) would appear.

The system started by default on the BL web pages set. Figure 3.2-a) shows all attributes related to the web pages themselves. Bar charts were chosen to represent the data, because most of the data was categorical and we wanted to compare the values of the several keys [17]. The x-axis of those bar charts encoded the possible values of each attribute, the y-axis encoded the percentage of web pages that have the given value, and the color encoded the set of web pages we were viewing (red for BL set and blue for NBL set). Figure 3.2-b) is the menu of the system (still incomplete at this stage). There were three options in this menu: an option to choose between web page attributes and host attributes (Figure 3.3); an option to choose the set to visualize; lastly, an option to analyze the values of the attributes based on the percentage inside the set of web pages or set of hosts. Since we were working with values that were greatly dispersed, there are certain values from a given attribute that appear in an incredibly small amount of hosts or web pages. A possible solution was to use logarithmic axis scale to display all values in the same chart, however, even with a logarithmic scale, some bars were impossible to visualize. So the adopted solution was to split the values into two groups: values that appeared in more than 0.5% of the web pages or hosts; and values that appeared in less than 0.5% of the web pages or hosts. Figure 3.2-c) was the place were the Uniform Resource Locator (URL) of the web pages or hosts would appear after the user select one or more attributes in the bar charts.

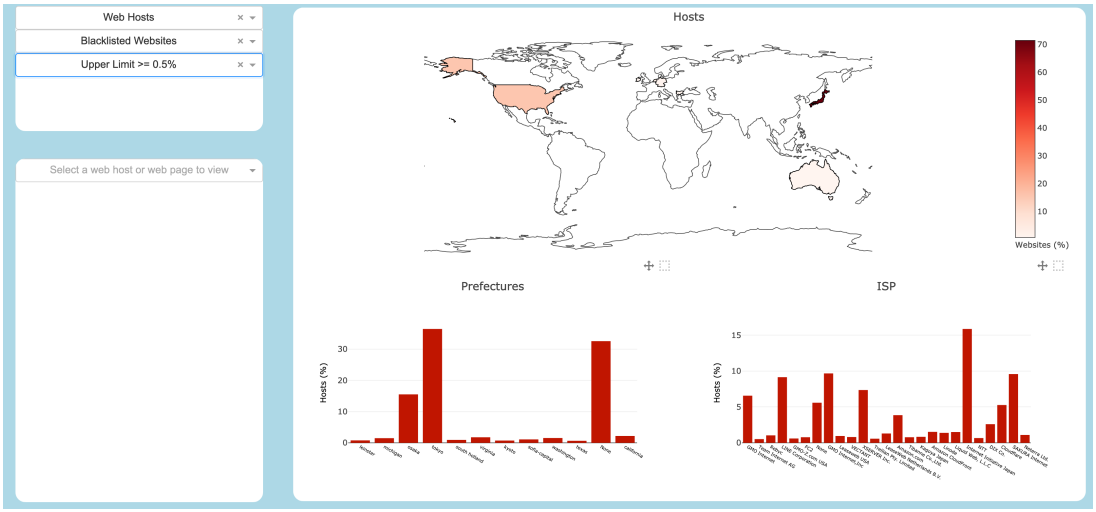


Figure 3.3: View of the attributes related to the hosts.

Figure 3.3 shows the charts chosen for the host attributes: country of origin, prefecture of the host and ISP. Even though Choropleths have problems regarding spatial aggregation and a continuous color scale could make it difficult to read the exact value for a given attribute [17], we chose it to encode the country of the host, since we wanted to analyze the data in a visual manner, to evaluate its potential as main criteria to find fraud patterns. The color scale encoded the percentage of hosts from a given

country. The prefecture values and ISP values were represented using bar charts. In each bar chart, the x-axis encoded the possible value of each attribute, the y-axis encoded the percentage of hosts that have the given value and the color encoded the same as in the web pages view.

Another feature we included in this prototype was the possibility to compare the set of BL and NBL hosts or web pages. Figure 3.4 shows how it was implemented. The idea was simple: transform the bar charts in grouped bar charts with the same properties as before but this time with a bar for each set. This feature allowed the user to compare directly both sets and understand if there was a difference great enough in some attribute that could indicate suspicious activity.

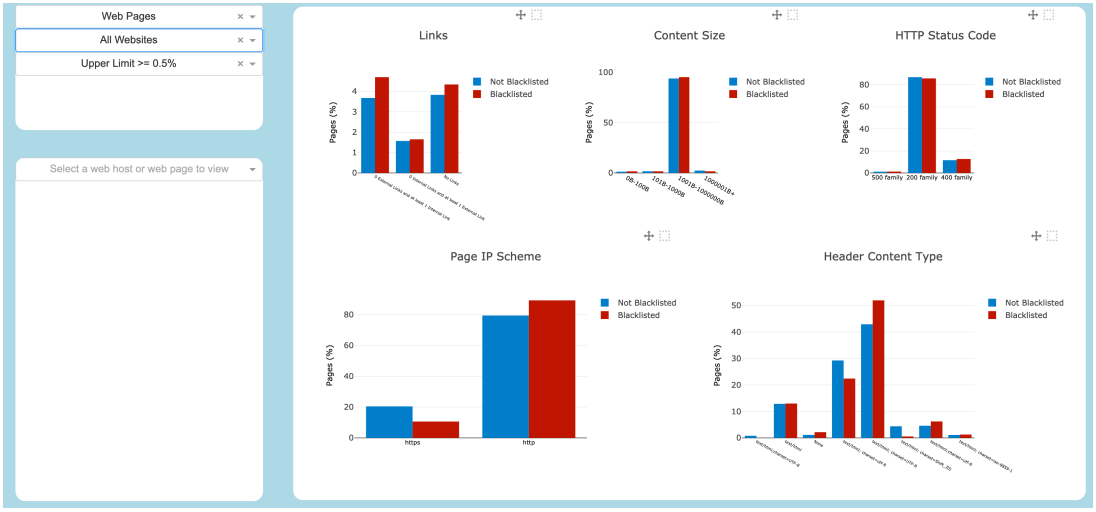


Figure 3.4: Web page attributes comparison with both BL and NBL sets

Although we had in mind the feature that included the URLs appearing in the left menu, it was not implemented right away because, before continuing, we wanted to validate our work with Phybbit. The references to Word Press and to adult content were also not included, since at the time this prototype was developed, we had not yet figured out an efficient way to display its data. The feedback provided by the team at Phybbit is detailed in the following section.

3.2.2.A Informal Evaluation

The informal evaluation was done with two people of Phybbit: Eurico Doirado (CTO) and Gonçalo Pereira (Team Leader and Senior Software Engineer), both with a vast experience in detecting advertisement fraud. The evaluation consisted in a remote meeting where we would describe the current state of the system, explain what was the process to find fraud pattern and what would be our next steps. After the description, Eurico and Gonçalo would give their feedback about the state of the system. The feedback was more centered in the process of finding fraud patterns than about the visual aspect of the system,

since they thought the visuals would not be relevant without a clearly defined process underneath it.

After implementing the first prototype's main charts, we had a meeting with Phybbit to receive some feedback about the adopted solutions. The meeting was extremely important to understand what was missing and how the prototype could be improved. The main conclusions of the meeting with Phybbit were:

1. There is still too much information in the landing page: Even reducing from nine to five different attributes in the landing page, the feeling that the user does not know which is the main focus of the system was still present.

2. The comparison between sets is an interesting feature, but it is not high priority: It was suggest to include this feature as future work and to focus more on finding fraud patterns.

3. There was no clear path on how the user would find fraud patterns: The feature to select values in charts and the corresponding hosts or web pages appear in the left menu bar (Figure 3.2-c)) was not an obvious interaction and it did not show what the pattern chosen to investigate was.

4. Data separation from hosts and web pages is useful, but attributes from hosts are not relevant: As we discussed previously, the geographical information of a host can be extremely inaccurate due to the method used to extract it. Therefore, Phybbit advised us to abandon the idea of having host attributes and focus solely on web pages attributes to find fraud patterns.

5. Verification stage was not present: In Section 3.1, one of the specified functional requirements was that we needed to inspect the web page in order to check if it contained fraudulent activity or not. This feature was not implemented at this stage.

With the provided feedback we concluded that there was not a clear process that would lead the user to find fraud patterns. Phybbit noticed this flaw and suggested us to manually find a fraud pattern that works and try to build a system that follows the steps we took to find it but instead of doing a manual analysis, transform that process using InfoVis oriented techniques.

The fraud pattern found, the steps that we took to find it and the prototype that originated from this process are specified in Section 3.3.

3.3 Second Prototype

The first prototype showed that we did not have a clear idea of how the system would work as a whole. The next developed prototype should focus on two important aspects: more emphasis on the attributes that support the process of finding fraud patterns and it should show a clear path from the start until the point where the user finds a fraud pattern. To achieve this, Phybbit advised us to find a fraud pattern that worked and then adapt the process to a system that used InfoVis techniques.

3.3.1 Discovering Fraud Patterns

The process to find fraud pattern requires, firstly, to theoretically find attributes whose values would indicate something suspicious is happening in the web page. After combining information from our data analysis for the previous prototype (Section 3.2.1) and Phybbit's feedback for the first prototype (Section 3.2.2.A), we chose to focus on these attributes:

1. **Links;**
2. **Content Size;**
3. **Header Content-Type;**
4. **Images:** This attribute was added because if we were focusing on web pages attributes, it made sense to consider images as a potential candidate.
5. **References.**

The reasoning behind finding fraud pattern is straightforward: find combinations of web pages attributes that lead us to a subset of web pages, in which the majority of web pages have evidence of fraudulent activity (keep in mind that crawl implementation bugs and/or updates that happen in a web page creates discrepancies between the data we have in the DB and what we observe when we open a web page).

Given that we had a huge amount of data finding all suspicious combinations using DB manipulation would be impractical. The solution was to choose extreme cases of some combinations of attributes that revealed suspicious activity inside a web page, using some knowledge we acquired while performing the data analysis. After reasoning about possible combinations that would work, we came up with the following working fraud pattern:

- **Zero Internal Links - Zero External Links - Zero Images - Content Size less of equal to 1000 bytes**

The chosen pattern finds web pages that are either empty, default pages from server applications such as Apache Server[6] or are not active/reachable. The values for links was chosen based on the assumption that an empty web page would not have connections to either an internal web pages or to external ones. The value for the content size and images were chosen after analyzing the corresponding table in the DB and upon verifying (opening random web pages manually) that most web pages with low content and low images count were suspicious. One can argue that the content size by itself would work as a standalone fraud pattern. However, we realized that sometimes the value for the content size was not the most accurate and that we had a higher success rate of finding actual fraudulent web pages with the full pattern.

[6] https://httpd.apache.org/

In the process of finding this working fraud pattern, we realized that by combining the number of images with the content size of a web page, we would obtain more accurate results than by using the header content-type as a filter by itself. The consequence of this finding was that we chose to drop the header content-type from our analysis, which means from that point on, we would work only with **number of links**, **number of images**, **number of references** (word press and pornographic) and **content size**. **These were the four (and only) attributes that made into the final version of the system developed in this research.**

The process that led us to find a working fraud pattern is described as follows:

1. Select attributes that could show suspicious activity in a web page.

2. From the chosen attributes, investigate their values in the NBL set: Manually done by going into the corresponding DB tables and search for values from extreme cases (for example, web pages with no images or with more than 500 images). There was also the possibility to look for values common in the BL set and search for web pages in the NBL set that had similar values. We have done that and we found two smaller fraud patterns:

(a) Word Press Reference greater than or equal to 19 (60^{th} percentile of BL set): This pattern revealed some potential but it was very hard to find what was wrong with the web page, so we decided not to focus on it at this stage of the prototype;

(b) Porn references greater than 20 (60^{th} percentile of BL set): This pattern was extremely accurate, but it was found after the main one, so it was an interesting finding, without adding anything new to the process of finding fraud patterns.

3. Combine attributes: Find attributes that are closely related and that together make a stronger and more accurate fraud pattern.

4. Verify if the web pages found actually showed evidence of suspicious activity.

The aforementioned process is what that user should be able to do when using the system. This is the flow Phybbit said our first prototype lacked. Since we had the attributes that actually worked (proven by the working patterns we found) and we had a clear process on how to find fraud patterns, it was time to develop a prototype that implemented this process, leveraging InfoVis techniques.

3.3.2 Working Prototype

One of the main issues of the first prototype was the fact the landing page of the system had too much information and it was not clear where the user should focus their attention to or even where to start. To fix this issue, the landing page was completely redesigned as shown in Figure 3.5. It is important

to mention that for this prototype we dropped the use of *Plotly Dash*[7] (it offered limited customization, required to build a custom UI) and started using HTML5[8] and Javascript[9]

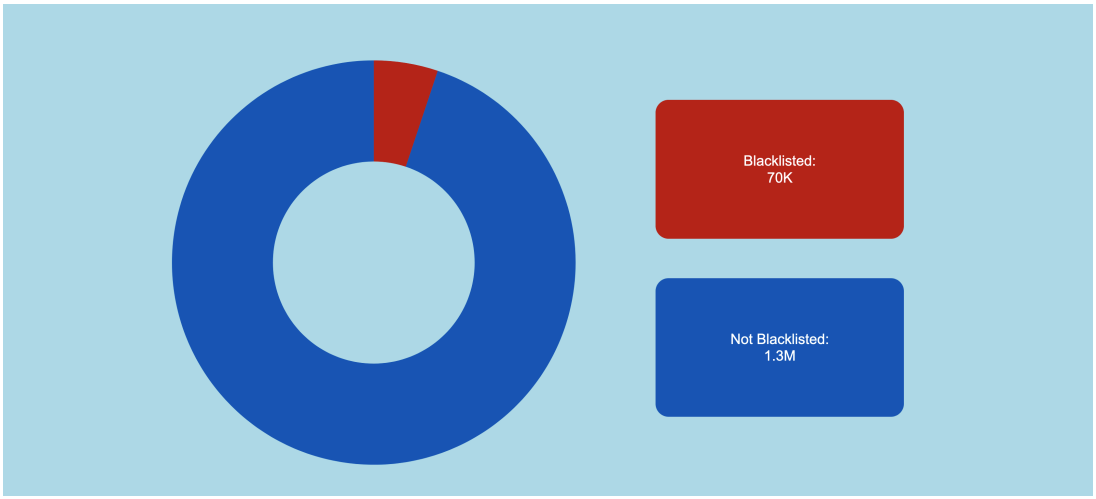


Figure 3.5: New landing page for the second prototype, showing the summary of both sets.

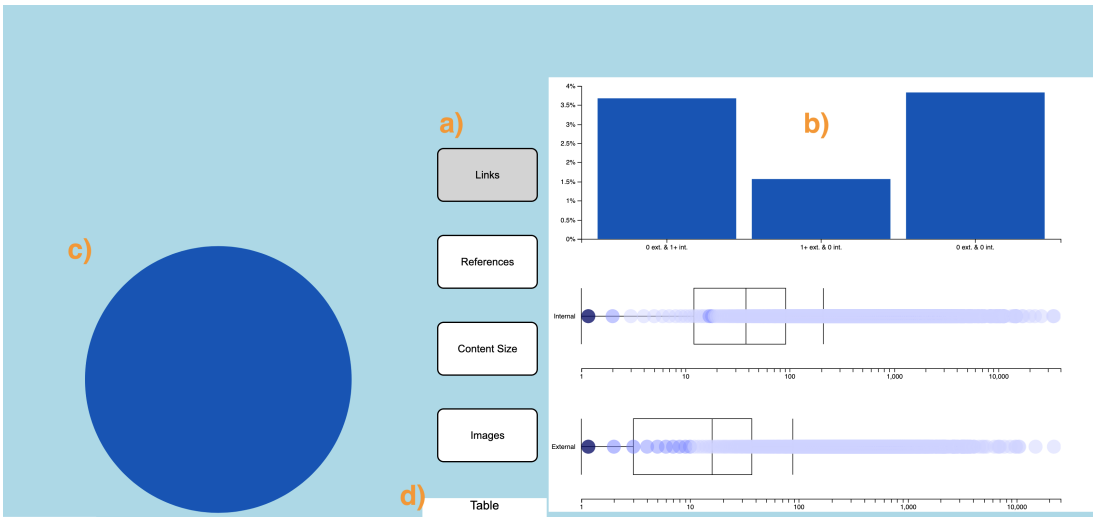


Figure 3.6: System view after user selects NBL web pages in the landing page and then chooses links.

In the landing page, a donut chart is used to encode the number of web pages in each set. The length of the arc encodes the number of web pages itself and the color encodes the type of set (red for the BL set and blue for the NBL set). Two colored rectangles were also included to show the number web pages. The user could select the set by clicking in either the donut chart or in one of the rectangles of the desired set they wanted to investigate.

[7] https://dash.plotly.com/
[8] https://www.w3.org/TR/2017/REC-html52-20171214/
[9] https://en.wikibooks.org/wiki/JavaScript/Print_version

The main page of the system is shown in Figure 3.6. The general idea of this page was to offer the possibility to, firstly, choose which attribute to visualize (Figure 3.6-a)); secondly, the user could view the information of a given attribute with the help of the charts chosen to encode the information; and then the user could apply filters by selecting an interval in one or more charts. At this point, two things would happen in the system: on the left, the chart would adapt itself to show the originated subset of pages after applying the filters and all the web pages would be displayed in the *Table* element, with the possibility of the previewing them using a HTML element called *iframe* [25].

In terms of charts, Figure 3.6-b) shows the chosen charts for the links attribute. The first is a bar chart: the x-axis encoded the three link relationships considered relevant (0 external links and at least 1 internal link; 0 internal links and at least 1 external links and no links whatsoever); the y-axis encoded the percentage of web pages with the given relationship; and the color encoded the active set. The second chart is a box plot for each type of link, with a heatmap made of circles: the x-axis encoded the number of web pages with a given value of links (a logarithmic scale is used due to huge amount of distinct values for each type of link); and each circle encoded a given value of links and the color scale encoded the number of web pages that had each different value. The idea was to use the heatmap to show how many web pages had a given value, without using a different chart. The charts used to encode the references information was the same box plot with heatmap as described for links (one for each type of references). For both content size and images it was used a histogram, where the x axis encoded each interval of values and the y-axis encoded the percentage of web page that had values in each different interval.

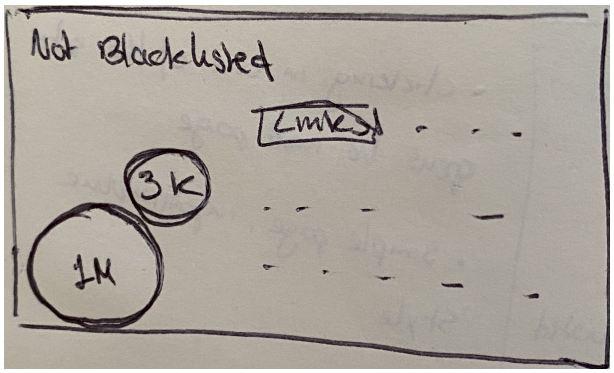


Figure 3.7: Hand sketch of how the chart in Figure 3.6-c) would work.

Figure 3.6-c) shows a sketch of a custom chart that would be updated after the user applied one filter using the charts on the right. The goal was to create proportional bubbles where the size encoded the number of web pages that had the filters chosen by the user (schematics in Figure 3.7). However, this chart was not fully implemented in this prototype because, at the time, we still did not find a proper

solution to encode the information we wanted.

An important feature in this prototype was having the possibility to view all web pages that match a filter created by the user (this feature had been planned since the first prototype, but it was only implemented in this prototype). Figure 3.8 shows how this feature was implemented. On the left, a table shows all the URLs of the web pages in the obtained subset after applying some filter and, on the right, the preview of the selected web page is shown, allowing the user to verify if there was something suspicious happening in it or not.

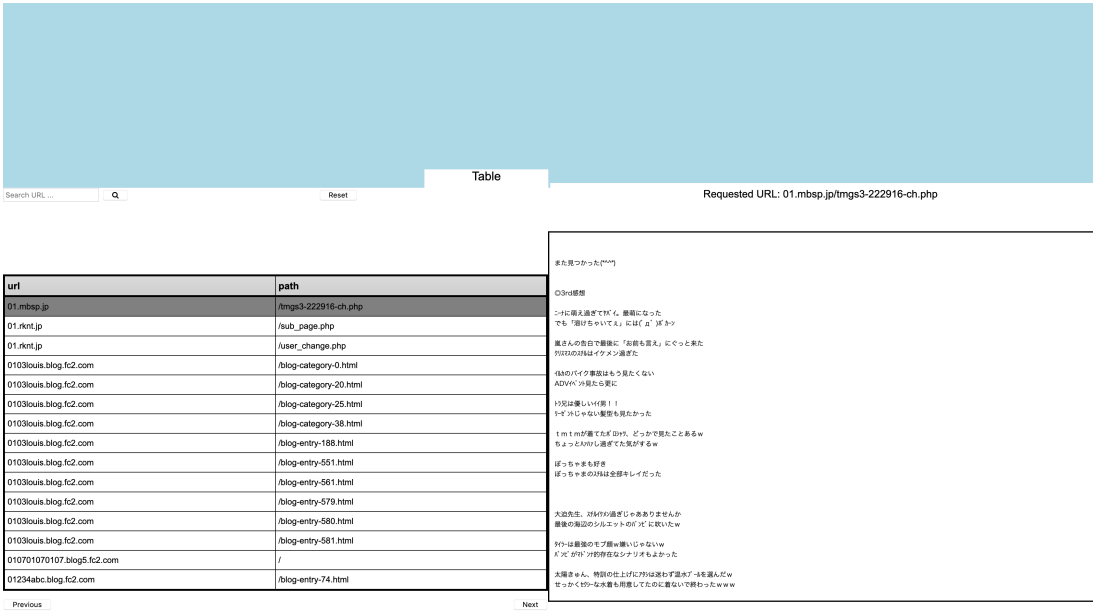


Figure 3.8: *Table View* with the URLs of the web pages and the preview of one of them.

This prototype was still incomplete when it was brought to the meeting with Phybbit: the user could only apply one filter at the time; the left chart was not yet chosen; there was no way to change the active set of web pages without reloading the entire page and the UI was still at an embryonic state. In the next section, we present Phybbit's feedback on the second prototype.

3.3.2.A Informal Evaluation

The meeting with Phybbit was, once again, particularly useful and productive. The main comment made was that, for the first time, they could observe a path from where the user should start and how it would reach the goal of this research: finding fraud patterns. The complete feedback is detailed as follows:

1. **The chosen attributes are more useful than the ones in the first prototype:** It was clear that these attributes would be more helpful. There was also, no extra information that could confuse the user.

2. UI has too much wasted space: The space could be better used to, for example, increase the size of the charts and offer a better view of the encoded information.

3. There was no indication of what pattern the user was analyzing;

4. Comments about the chosen charts: Phybbit suggested we look for improvements in two specific charts.

(a) box plots: They liked the box plot but thought the heatmap added an extra layer of entropy, and so, they advised us to think about an alternative solution.

(b) Links Bar chart (Figure 3.6-b)): The chosen relationships are useful, but since there was no possibility to visualize other relationships, it limits the user freedom when exploring the information.

Bonus feedback: Use HTML and Javascipt exclusively: Phybbit also suggested that an overall improvement in our workflow was to download the DB tables we needed to implement the charts and drop the support of fetching data in a local DB, because for every new chart, it was required to change the queries that fetched the data and we were losing time with it. With HTML, Javascript and the tables in Comma Separated Values (CSV) files, it was easier for the data to be custom processed for each type of chart, but it had the trade-off of making the overall system slower.

Phybbit's feedback showed that there were significant improvements from the first to the second prototype, however it also hinted that it was needing a clean up in the UI itself and that we should try to find solutions to some of the issues we had with the chosen charts.

3.4 Final Version

In Section 3.2 and Section 3.3 we described the prototypes that helped us understand the available data and to identify a possible path the user could take to find fraud patterns. We sill need to explain how we implemented the final version of this research and the reasoning behind every decision made. The final version description will be split in three parts: first, we will explain the chosen UI; then, we will focus our attention on the chosen charts and others that were rejected along the way; and lastly, we will describe all system's functionality.

3.4.1 User Interface Implementation

One of the criticism made to our second prototype was that there was too much wasted space in the UI that could be used to better integrate all elements (Section 3.3.2.A), so, it made sense to improve it the best we could because it would improve greatly the overall user experience.

Based on previous feedback, we decided that the UI needed to support two important features: the

user should be able to easily change attributes and analyze the corresponding charts; and the UI needed a region to place the results of the filters made by the user.

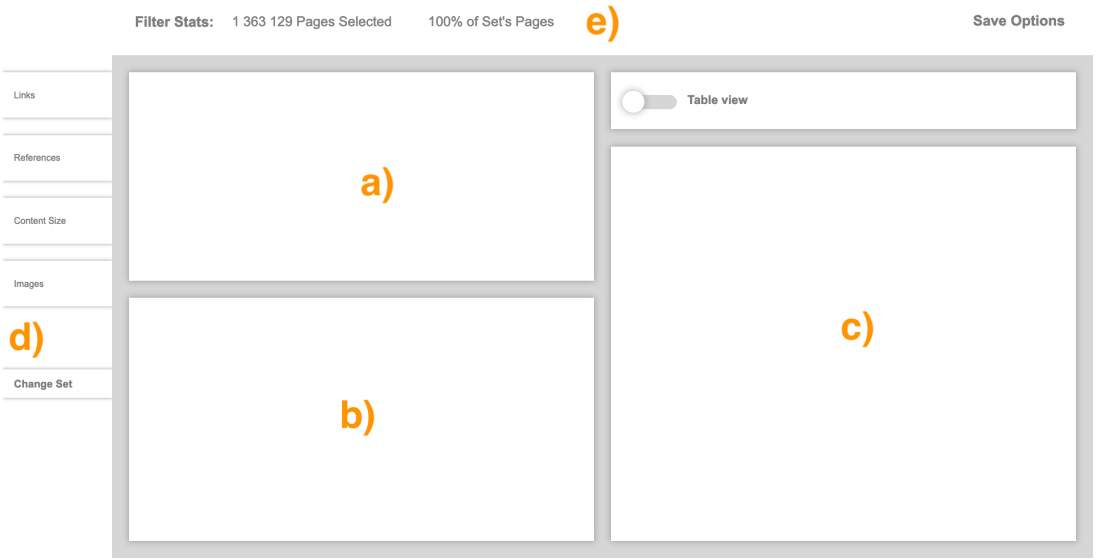
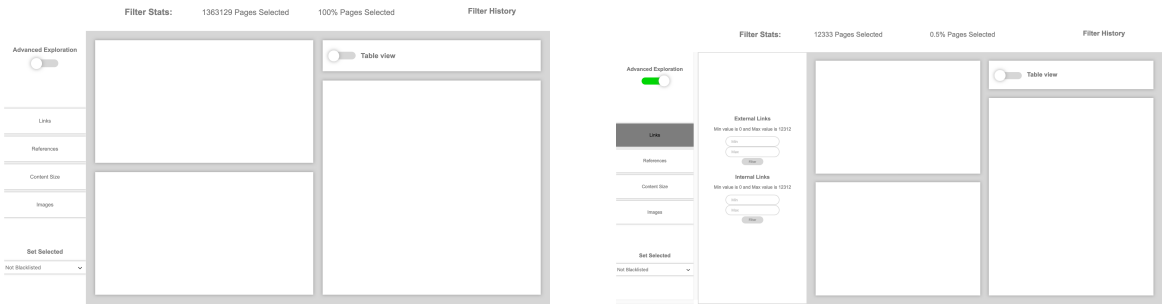


Figure 3.9: Main page of our UI final version.

Figure 3.9 shows the main UI of our research. Figure 3.9-a) and Figure 3.9-b) are the regions reserved for the charts we developed (more details on Section 3.4.2). Figure 3.9-c) shows where the results of the applied filters will be placed. If the user wants to inspect the web pages that are inside a subset, they can switch to *Table View* and they will see a table with all web pages and a space to preview a web page (done by selecting one web page in the table). On the left of the UI (Figure 3.9-d), we have the main menu where the user can select which attribute they want to investigate and the charts for the selected attribute will appear in the corresponding UI regions; or change the active set by clicking on *Change Set* button. Finally, on the top of the UI (Figure 3.9-e), we have a toolbar that starts by showing the logotype of Phybbit's main fraud detection tool (upon clicking, it redirects the user to the company's website); then we have what we've called *Filter Stats*, whose purpose is to show how many pages the user is investigating (it updates once the user starts analyzing a given pattern); and lastly, we have the save menu where the user can save the progress of their investigation or load previously found patterns.

The UI was not always like we explained. The first version is shown in Figure 3.10, which includes a feature we later dropped: *Advanced Exploration*. This feature would allow the user to specify an interval of values (Figure 3.10(b)) and the charts would be drawn just with the values that belonged to the specified interval. However, this feature was quickly dropped because it did not make sense in the context of finding fraud patterns to first choose an interval and only then view the information for that attribute. It makes much more sense the other way around: first check what is happening in a set of web pages (BL or NBL) for a given attribute and only then, choose and investigate an interval of interest.



(a) First version of the final UI, with *Advanced Exploration* toggled off.

(b) When *Advanced Exploration* was on, it appeared a new menu.

Figure 3.10: First iteration of the final UI.

It remains to explain the landing page of our system. Figure 3.11 shows its final look. It remained very similar to the landing page we developed for the second prototype (Figure 3.5). The main difference is that we included a third option: the user can choose to investigate all web pages regardless of their classification, i.e. the set of web pages that contains the union of the BL set with the NBL. The idea is to, in the future, add a functionality that allows the user to choose a fraud pattern, apply it to all web pages and verify how many of them were already blacklisted and how many were not.

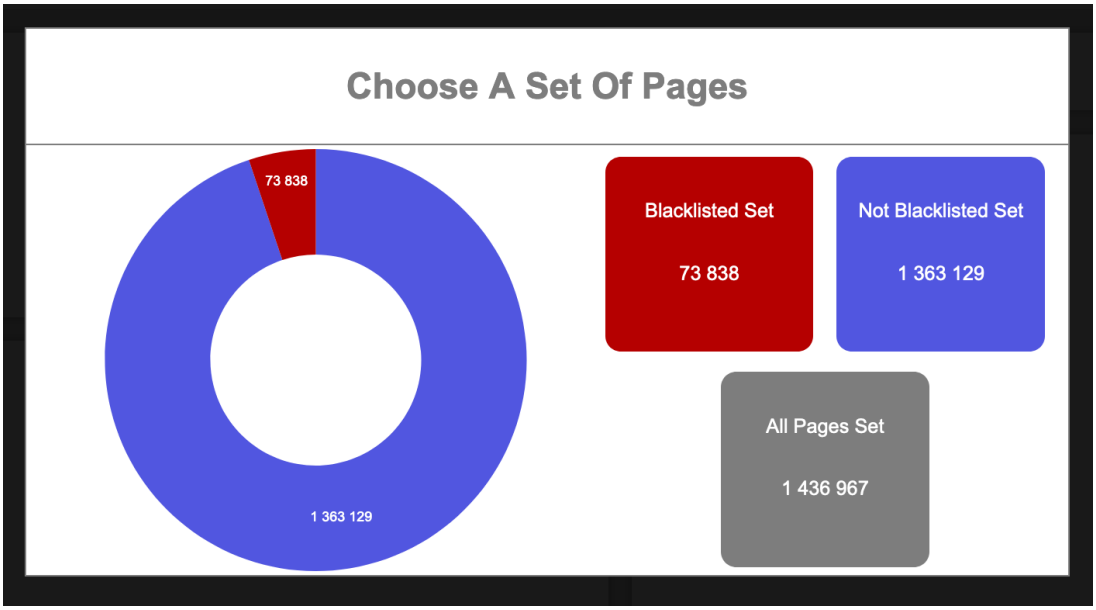


Figure 3.11: Landing page final version.

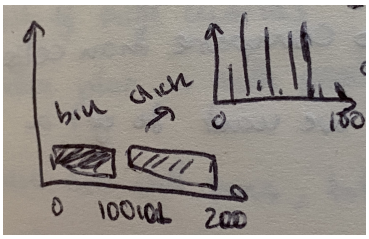
With the main UI and landing pages fully explained, the next step is to go through the process of selecting the charts we decided to include in our final version of this research and the reasons behind it.

3.4.2 The charts

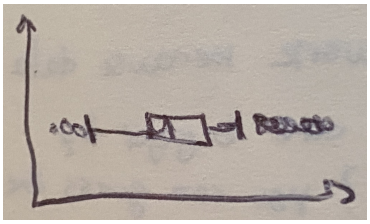
Since the second prototype, we have had this clear idea to use charts to filter some values of interest and the combination, i.e. intersection of several filters would originate our desired pattern. To achieve this, we divided the second prototype in charts that displayed information and in a custom chart that would gather all the filters and the intersections between them (Section 3.3.2).

In terms of charts used to display information, we wanted two kinds of charts: one that showed an overview of the distribution of values inside a give set (BL, NBL or both); and one chart that would focus on an interval of interest, i.e. an interval of values that our analysis said web pages with them would have some suspicious activity. The different granularity, would provide the user distinct ways to discover suspicious activity, which would connect directly to our first InfoVis task (Section 3.1.1).

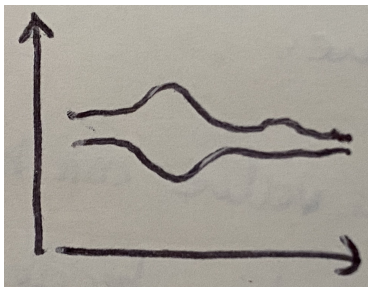
3.4.2.A Overview charts



(a) Heatmap and bar chart concept.



(b) box plot and outliers concept.



(c) violin plot concept.

Figure 3.12: Overview charts concepts.

The first option considered was to replace the box plot and heatmap from the second prototype (Figure 3.6-b), with a single heatmap made of bins (intervals) of values, where the color scale would encode the amount of web pages with the values in each bin. Then, to take a closer and more detailed look, the user could click on a bin and a bar chart would appear showing each individual value inside the bin (Figure 3.12(a)). This approach had a few problems:

1. Size of bins: If each bin covers a wide interval, the bar chart would be unreadable since it would have a lot of bars to fit in a small space. If we use small intervals for the bins, the bins could not fit in the area reserved for chart.

2. Color scale may mislead the user: Consider that we had a bin covering the interval from 0 to 100 internal links. If we take a closer look at our data, there are almost three times more web pages with 0 links than with any other value in that interval. So, the 0 would have a higher weight in that interval. Now, consider an interval of internal links from 101 to 200. In this interval the contribution is about the same for all values and, for this example, consider that the total number of web pages is the same as in the previous interval, which means the bin would have the same color. This could lead the user to think the distribution in both intervals was the same, but in reality, in the first interval the majority of web pages had the same value.

3. Complexity: This chart would require a few layers of interactions to allow the user to do the filter the data they wanted, which would hurt the user experience and would probably required a second chart to help show all the data, while keeping a record of the state of the visualization.

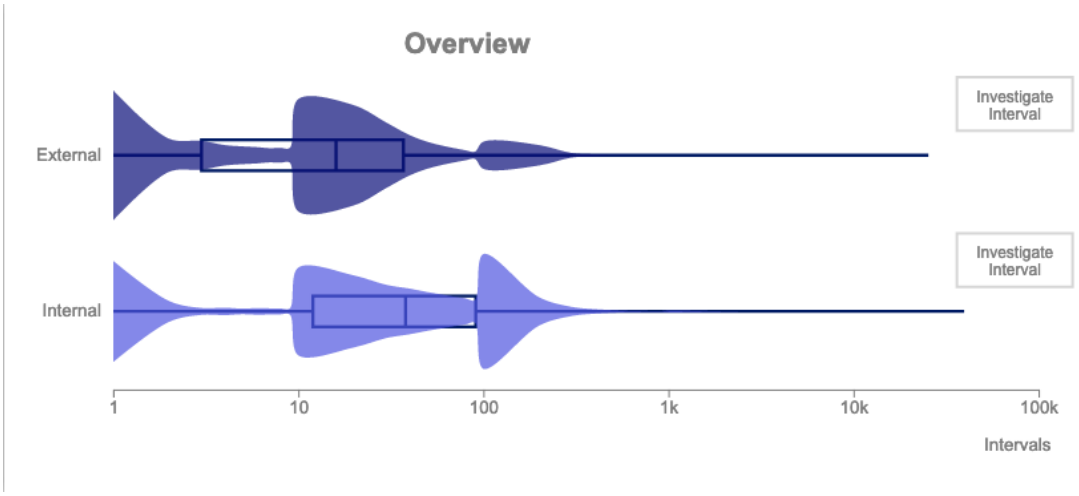


Figure 3.13: Violin plot and box plot used together to encode more information.

Figure 3.12(b) shows our second considered option. This time we had a box plot with the outliers drawn. However, the presence of circles encoding each value of the outliers would cause a problem called ”Rectangle Effect”, which means the circles, by being too close to each other, would lose their original shape and would form a bigger rectangular shape (this effect is visible in Figure 3.6-b). Also, the box plot by itself was not the best to filter since it does not show how many web pages have a given value.

Our final concept was the use of a violin plot (Figure 3.12(c)). violin plot would solve the problem of the box plot: violin plots can encode the the total number of web pages with a given value in the y-axis.

However it had a huge issue: violin plots did not work well with data that is as greatly disperse as ours (it would had an almost linear shape, with a small bump in the most common value), so, this means the violin plot could not be used without heavy tweaks to work with our data.

The solution we came up with was to blend the best of both box plots and violin plots: having the straightforward visual information of a box plot with the possibility of encoding the total number of web pages and the ease of filtering of a violin plot. Figure 3.13 shows the final solution for the overview chart.

The violin plot in our final solution is made by using a histogram and then applying a smoothing function to replace the straight edges of the bars of the histogram with a curved line. The box plot is placed underneath it to show the user what are the most common values of an attribute. Figure 3.13 shows the charts for the links attribute. Since there are two types of links, each has its own chart. In them, the x-axis encodes the different interval values for the attribute using a logarithmic scale; the y-axis encodes the number of web pages that are inside each interval; the color encodes the active set (in this case the active set was NBL) and the shade of the color is used just to distinguish each type of attribute. The chart used for the references attribute is the same as for the links. In the case of images and content size attributes the reasoning is the same, but since there is only one type of images and content size, only one violin plot is used for each. The interactions that are possible with these charts are explained in Section 3.4.3.

We described the chart used to show an overview of the set the user is investigating. In the following section we describe the chart used to show an interval of interest.

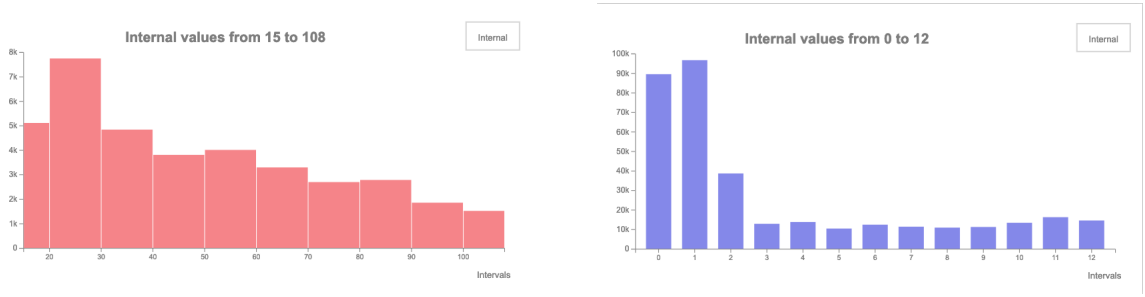
3.4.2.B Chart for intervals of interest

We wanted to provide the possibility for the user to investigate an interval in a more straightforward fashion than to include multiple layers of interaction in a single overview chart. The chart we would choose had to be as simple as possible but capable of showing a large amount of greatly dispersed data.

Before analyzing our solution it is important to mention that at the start, there are some default intervals of interest for the user to analyze, and they depend on the set of web pages that is active:

1. Default interval of interest for BL web pages set: For the BL we decided that would be interesting for the user to view the values that are most common in blacklisted web pages (they are good candidates to be used as filters for a fraud pattern), and so the default interval of interest would be: $Q1 <= Values <= Q3$, where $Q1$ and $Q3$ are, respectively, the first and third quartiles of the distribution of the chosen attribute.

2. Default interval of interest for NBL and all (union of BL with NBL) web pages set: For these sets, we want to investigate suspicious values, so we have to use cases of suspicious values (that



(a) Histogram for Internal Links in BL set

(b) Bar chart for Internal Links in NBL set.

Figure 3.14: Chosen chart to encode intervals of interest.

can also be called outliers). In the case of links, images and content size, web pages with low values for those attributes are suspicious, so the default interval of interest would be: $min <= Values < Q1$, where min is the minimum value for an attribute. In the case of references, we considered suspicious web pages that have a great amount of any type of references, so the default interval of interest would be: $Q3 + 1.25 * IQR < Values <= max$, where IQR is the interquartile range and max is the max value for an attribute.

Figure 3.14 shows our adopted solution. The chart for the default interval is initially either a histogram or bar chart, depending one the amount of values in that interval. Figure 3.14(a) is an example of a histogram: the x-axis encodes the values for the selected attribute in the form of intervals; the y-axis encodes the total number of web pages inside a given interval and the color encodes the active set of web pages. Figure 3.14(b) is an example of a bar chart: the x-axis encodes the values for the selected attribute; the y-axis encode the total number of pages for each value and the color encodes the active set of web pages. As before, the functionality of this chart is described in detail on Section 3.4.3.

The last chart in our research is the chart that combines all the filters and shows the pattern the user is investigating. We called this chart *Selected Filters Plot* and it will be described next.

3.4.2.C Selected Filters Plot

Once the user finds a value they think it could reveal something suspicious in a web page, the user needs to view the subset of web pages that have the selected value. In the second prototype, we had this in mind, but we still did not have a way to encode that information. So, for the final version, we went back to the drawing board and thought about how could we treat the filters made by the user and how we would display them using InfoVis techniques.

After brainstorming possible approaches, we came up with a solution that not only solved our problem

but also had some extra functionality:

1. Create a subset of web pages for every selected filter: For every time the user filters one of the charts of our final solution, we create a subset of web pages that have the selected value and store that subset in a custom made data structure.

2. Intersect subsets: Once the user filters more than once, we not only create the corresponding subset, but we also compute the intersection of the new subset with all subsets in our data structure. But with one caveat: it only computes the intersection with subsets of different attributes. Figure 3.15 shows an example of this computation. When the user reapplies filter *F1*, the subset will not intersect with the first branch, because the first branch already have a intersection with a filter of the same type (first *F1*).

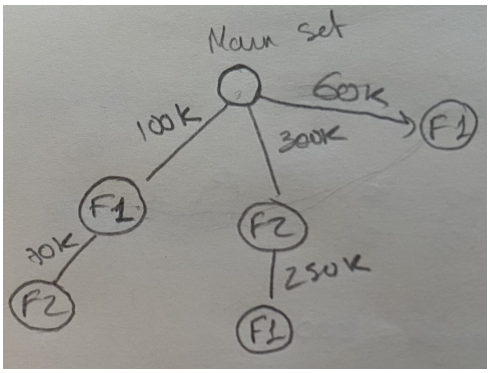


Figure 3.15: We start in the active set and for each filter, we intersect the corresponding subset with the ones already there. (Filters order: F1-F2-F1).

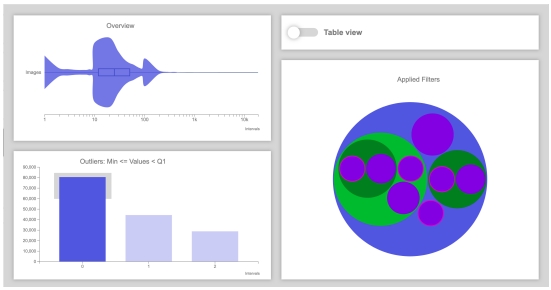
This solution allows the user not only to visualize the pattern with the applied filters, but also other smaller combinations of the same filters. The main advantage is that by doing multiple combinations, if the user wants to investigate all the smaller patterns, they could find some smaller version of pattern that is as effective as the full pattern the user had in their mind when started using the system.

The difficulty of this approach was to find a chart that could encode all this information. To test the several candidates, we used the fraud pattern we found in the preparation for the second prototype (Section 3.3.1) as benchmark to help us choose which was the chart that gave us the best results.

The results of our testing are shown in Figure 3.16. Each filter has its own color so the user know which is the resulting intersection of the last applied filters:

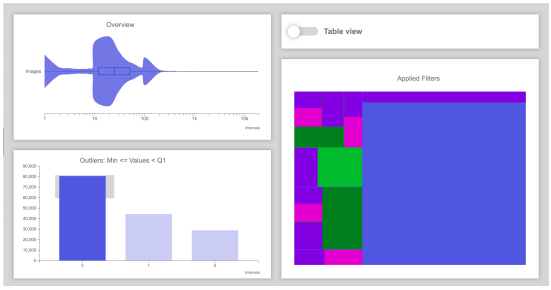
1. Force directed graph (Figure 3.16(a)): The idea was to make a chart as close as our sketch shown in Figure 3.15. We used a library specific of D3.js[10]to produce this chart. However, it had some problems: it did not scale well when we tried to add more filters; the circles could not be proportional to the number of total web pages in each subset, because of the poor space management of this chart; and having a radial approach could make it hard for the user to find the any pattern. We

(a) Force directed graph.

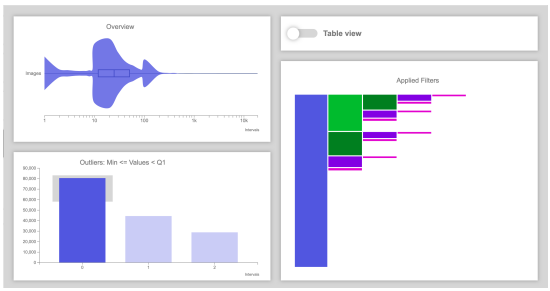


(b) Packed circles.

(c) Radial Dendogram.



(d) Treemap.



(e) Partition Layout.

Figure 3.16: *Selected Filter Plot* candidates.

excluded this chart.

2. Packed Circles (Figure 3.16(b)): This chart supported proportional circles for each subset, but it was not possible to view the pattern and the other filter combinations without having to interact with it. Since we wanted an approach that showed all combinations on demand, we excluded this chart.

3. Radial Dendogram (Figure 3.16(c)): We gave a second change to a radial approach, but since it showed the same problems as with the force directed graph, we also excluded this second (and last) radial attempt.

4. Treemap (Figure 3.16(d)): Unfortunately, the treemap had the same problem as the packed circles: we cannot understand the pattern and all other filters combinations without interacting with it. Once again, we excluded this chart.

5. Partition Layout (also known as icicle, Figure 3.16(e)): This was the chart that met our requirements: it showed differences on the size of the rectangles depending on the subset's size; it was possible to view the full pattern and all other filters combinations; and it scaled well when we added more filters to test the full capacity of this chart.

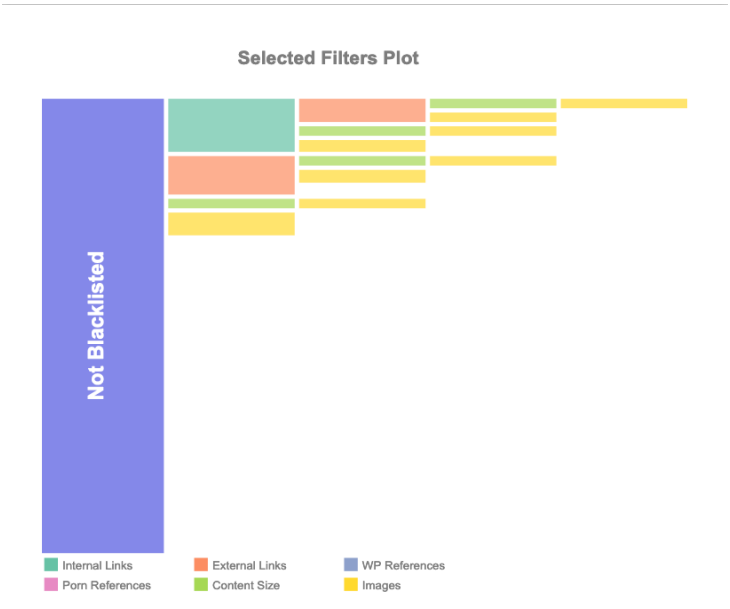


Figure 3.17: Final version of the *Selected Filters Plot*, with the corresponding legend.

We have chosen the chart to encoded the information we wanted and the next step was to make the necessary adjustments to work with our data structure.

The final version of our *Selected Filters Plot* is shown in Figure 3.17. Each rectangle encodes a subset of web pages, either resulting directly from the filters the user has chosen (the rectangles in contact with the active set, represented by the vertical rectangle) or resulting from the intersection with

[10] https://d3js.org/

other subsets (all other rectangles); the height of each rectangle encodes the total number of web pages in each subset; the x-axis encodes how many intersections originated a given subset (in this case, the maximum number of intersections is three, $max.intersections = total filters - 1$, and it is always displayed in the top row); the color scale encodes the type of the applied filter. The functionality of this chart is fully covered in Section 3.4.3.

If we observe the *Selected Filters Plot*, we notice that there are several rows of possible filter combinations. The top row always shows the pattern that the user is investigating, i.e. the pattern generated by the filters the user applied manually. All other rows are smaller combinations of the same user filters created automatically by the data structure (explained in Figure 3.15). Furthermore, the pattern found in Section 3.3.1, generates a total of 15 rectangles in the *Selected Filters Plot*, which means each of those rectangles represents a different subset of web pages that can be viewed as different patterns.

To close the description of all charts that our research offers, we want to explain the used colors and why they were selected.

3.4.2.D Colors

In this research we use a total three main colors across all charts:

- Red: We use the color red (and some variations of it to distinguish elements, such as, charts for different types of the same attribute) to encode everything related to blacklisted web pages, because red is a color often associated with danger [26] and since blacklisted web pages have some fraudulent activity in them, they represent danger to any company/person that wants to place Ads in them. The main shade of red used in our system is the same Phybbit has in the logotype of their advertisement fraud detection tool, SpiderAF.

- Blue: Since NBL is the opposing set of BL, we wanted a color that was in the other end of the color spectrum. Red and blue are often used in both ends of color sequences [27], so it made sense to use blue as the color to encode everything NBL related. The main shade of blue used in our system is the same present in Phybbit's company logotype.

- Grey: We needed a third color to encode everything related to the set resulting of the union of both BL and NBL web pages. We chose grey since it is a neutral color that does not hint anything when analyzing all web pages indiscriminately. The main shade of grey used in our system was randomly picked.

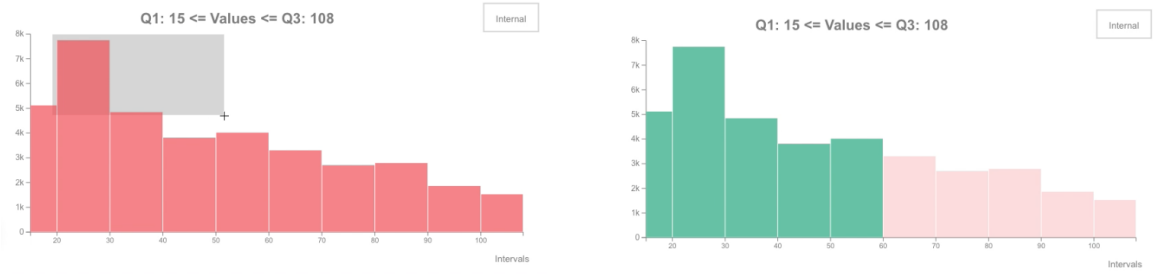
The last colors we want to discuss are the colors used in the *Selected Filters Plot* to encode each type of filter. We were looking for a color scale that did not interfere with the colors of each set of web pages and, at the same time, were color blind friendly. We selected the colors with the help of an online tool developed to provide color scale options following user defined criteria [28].

We have now completed the description of all charts we included in our final solution. We've looked at how and why each chart was chosen. However, we still need to go through all the functionality and interactions offered by each chart and how they work together to achieve our goal: find fraud patterns in web pages.

3.4.3 Interactions & Functionality

In Section 3.4.2, we have described each chart we included in our final solution and the reasons that led us to implement them the way we did. However, we did not cover the interactions and some functionalities of the charts and the system in general. In the next sections we will fully explain the interactions in each type of chart and we will talk about the functionality of our system as a whole.

3.4.3.A Chart Interactions



(a) Brushing mechanism starting.

(b) Result after the user is satisfied with their selection.

Figure 3.18: Example of the brushing interaction.

As mentioned before, the user can select values in any chart and that selection will be part of the pattern the user is investigating. Both charts for the overview and for the interval of interest, share the same basic interactions:

1. Brushing: The selection of data in any chart is done by using a brushing mechanism. The brush starts by pressing down the left mouse button in the white area of a given chart, then the user can move the cursor until the brushing area covers all the data they want to select and, once the user releases the mouse button, the brush is complete. Figure 3.18 shows an example of the brushing mechanism in the histogram for the Internal Links attribute in the BL set of web pages. Notice that, once the brush completes, the chart elements (in this case the bars) change color to match the respective color in the color scale of our *Selected Filters Plot* (Figure 3.17), giving the user a visual connection between charts. In the case of the overview charts, the selection is done in the same manner, but since some attributes (links and references) have more than one chart, each has its

own brushing and they can be used at the same time or just one, depending on the user needs. Figure 3.19 shows an example of a selection in each of the links charts. It is important to mention that for any given bar included in the selection, there is no need to be fully covered by the brushing area, i.e. the brushing area (grey area visible in Figure 3.18(a)) only needs to touch a small portion of the bar.

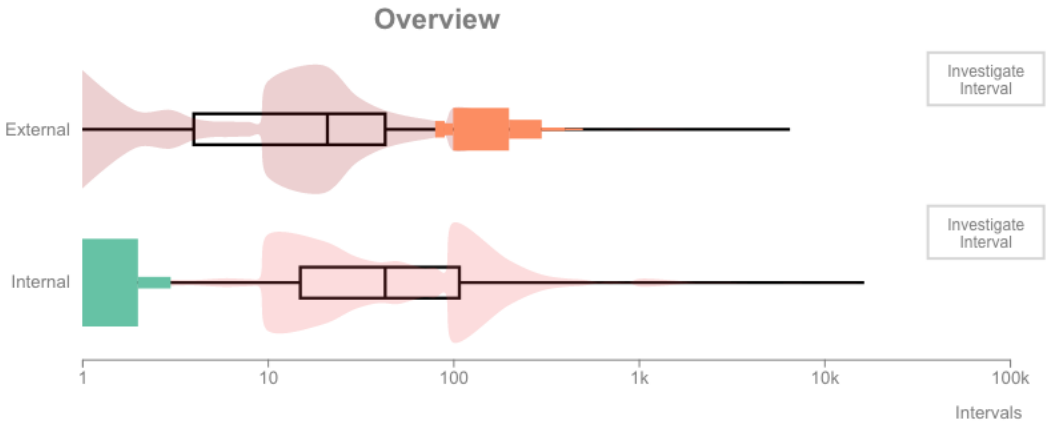
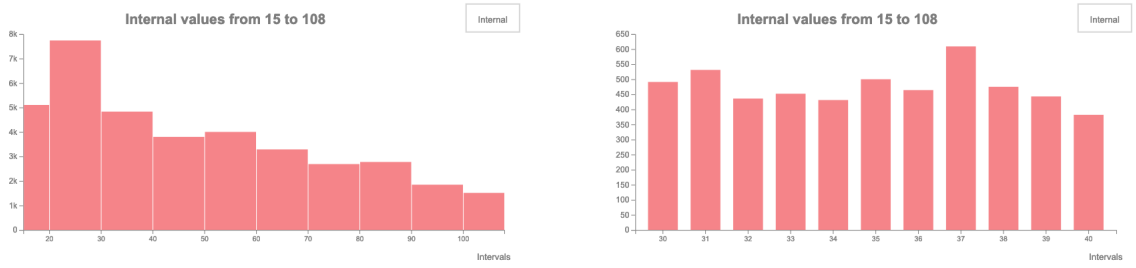


Figure 3.19: Each overview chart has its own brushing.



(a) Histogram for Internal Links in BL set of web pages.

(b) Bar chart for the values between 30 and 40 internal links.

Figure 3.20: Interaction to investigate the values inside an interval of a histogram.

2. Inspect values inside a histogram interval: In the case of the charts for the intervals of interest, they offer other interaction besides the brushing: it possible to click on a histogram bar and analyze the values inside it. Figure 3.20 shows a working example, where the user clicks on the histogram bar for the values between 30 and 40 internal links, and a bar chart appears with encoded the information for the selected interval. When the interval is too big to be encoded with a bar chart, another histogram is used. This means a chart for intervals of interest can generate quite a few other histograms until the user clicks on a interval which information small enough to fit in a bar chart. This

feature is particularly useful if the user wants to apply a filter that is a single value instead of a whole interval.

3. Hovering chart elements: Every chart has a hovering mechanism. For a histogram based chart (violin plot in the overview charts and some charts for the interval of interest), when the user hovers a given bar, a tooltip appears with the interval of values of that bar and the total number of web pages that belong to the hovered interval. For bar charts, the tooltip only shows the total number of web pages with the hovered value. Lastly, in the *Selected Filters Plot*, hovering over a rectangle will show information regarding all the combinations that created the hovered subset (Figure 3.21) of web pages, including the filter (or filters) and the total number of web pages in each intersection that precedes the subset.

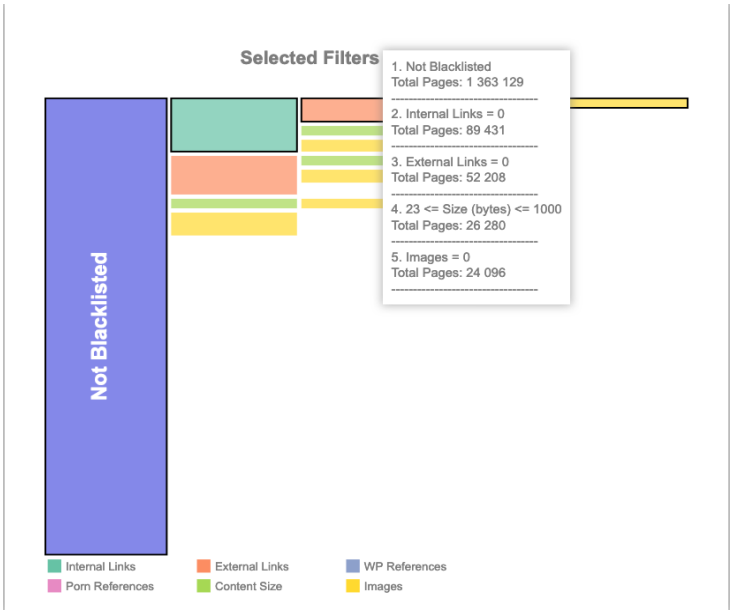


Figure 3.21: Hovering a subset in *Selected Filters Plot* shows all the filters that contributed for the intersections that originated the hovered subset.

4. Resetting the state of the charts: The white area around the element for a chart works like a reset button. If the user wants to clear the brushing, they can click on the white area of the chart with the selection and the brushing is cleared (the charts go back to their original color, all other charts stay the same). In the charts for the intervals of interest, the user can go back to the original histogram with a double click in the white area. This behavior is the same across all other charts: single click in the white area to clear some selection and double click to reset any chart to its original state before any user interaction.

The aforementioned interactions are chart specific. The final step in our system walk through is to explain how we can use the various UI elements and charts to reach a suspicious pattern and, once we

have one, how can we verify if our pattern can be considered as a fraud pattern or not.

3.4.3.B Main Functionalities

In terms of functionalities, we already saw that the user can filter data in the charts and the combination of filters originate a pattern. However, all the processes we have covered in the interactions section were using only the the default views of our system. To find a more complex pattern similar to the one discussed in Section 3.3.1, the user will have to do quite a few interactions that combine charts and UI usage.

The first group of functions include the use of the buttons in both overview charts and the charts for the interval of interest. In the overview chart, there is a button called *Investigate Interval*, of which its function is to allow the user to change the interval of interest in the chart below, by selecting an interval in the overview chart.

Figure 3.22: It is possible to change the chart of interval of interest by using the overview chart.

What the *Investigate Interval* button does is engage a different operation mode (button changes color) that removes the smoothing function of the violin plot, which allows the user to view the underlying histogram (behavior visible in fig. 3.22). Then, we take advantage of the brushing mechanism to select an interval. While in this mode, the brushing only alters the chart below, and the selection is not considered a filter to be included in *Selected Filters Plot*. The new chart for the selected interval works exactly the same way as the default one does. The user can go back to the regular violin plot by clicking again

Figure 3.23: Top stats are updated when a subset of web pages is selected in *Selected Filters Plot*.

in the *Investigate Interval* button. It is important to notice that going back to the default violin plot will not erase the newly drawn chart for the selected interval. To reset the chart for the interval of interest, the user can use the dropdown menu that is on the top right of the chart and look for a button called *Default*.

The dropdown menu in the charts for intervals of interest is used to switch the type of the attribute being encoded (visible in Figure 3.20). As with the violin plots, if an attribute has more than one type, the dropdown menu allows the user to switch between them. This means that, for example in the case of the links, the dropdown menu has the option of viewing the chart for the internal links or for the external links. This switching method is used in our fraud pattern to select the value of internal and external links.

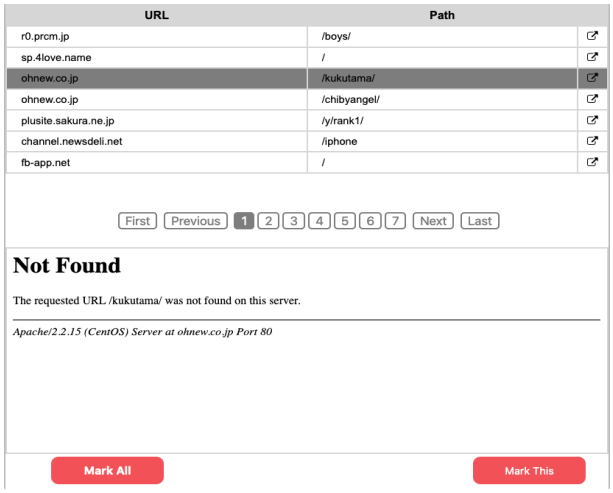


Figure 3.24: In *Table View*, the user can analyze the web pages that match the selected pattern and verify its contents.

Lets switch our attention to the *Selected Filters Plot*. To select a pattern to investigate, the user can click on the corresponding rectangle. This will trigger two changes in our system: the pattern will be highlighted in the *Selected Filters Plot* (black line around all rectangles i.e subsets that are part of the pattern) and, on the top of the UI the values for *Filters Stats* will match the number of web pages in the selected pattern (Figure 3.23). It must be mentioned that the white area of this chart works the same as specified before, with an important difference: double clicking the white area will trigger a reset on the *Selected Filters Plot* but it will also reset all other charts to their default value, this being an effective way to reset the state of our system without having to reload the entire page.

Once we have a pattern selected, the next step in the process of finding a fraud pattern is to verify the existence of suspicious activity in the web pages. To achieve this, the user can change to the *Table*

View and start opening web pages to analyze their contents.

Figure 3.24 shows us the *Table View* in detail. Once a user selects a web page, it changes color (from white to grey) in the table to give visual feedback about what is the opened web page. To close its preview, the user can click again on the corresponding table entry. If the user finds some suspicious activity happening, they can be mark the web page, using the button *Mark This*. Marking a web page is to simply put them in a staging blacklisted list that can be saved or erased, depending on the user needs. After the user finds a number of suspicious web pages they consider to be representative of the whole pattern they are analyzing, the user can use the button *Mark All* to mark all web pages that belong to the selected subset without having to mark each web page individually. Clicking again on one of the buttons, it is possible to either unmark a given web page (click on the table entry and then in the *Mark This* button, now called *Unmark This*) or to unmark all web pages (click again in the *Mark All* button, now called *Unmark All*, which erases the staging list of marked web pages). Figure 3.25 shows all web pages from our studied fraud pattern marked, where the color encodes the state of web page (red for marked, white for not yet marked).

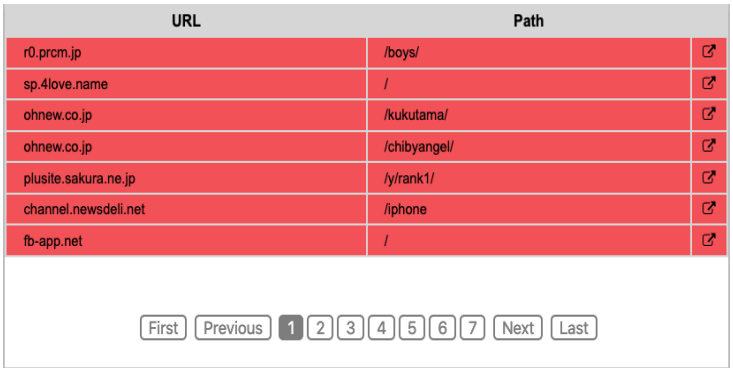


Figure 3.25: When the user marks one or more web pages, their entries change to red in the table.

Table View also offers a search function that allows the user to check whether a given website has web pages in the selected pattern.

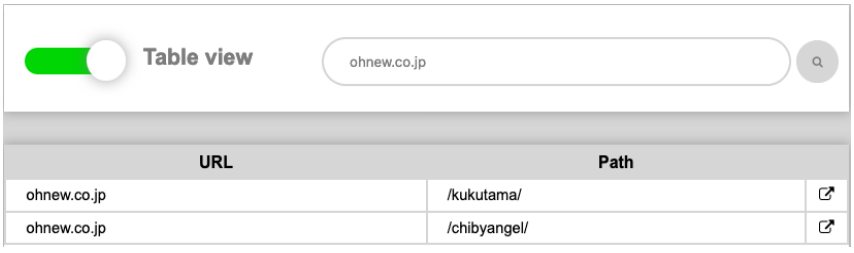


Figure 3.26: Search result for the website ”ohnew.co.jp”, with the fraud pattern described in Section 3.3.1.

Figure 3.26 shows the search result for the website ”ohnew.co.jp”. The search result revealed that a total of two web pages from the website ”ohnew.co.jp” were inside the fraud pattern described in

Section 3.3.1. This functionality can be particularly useful if the user suspects about the content of a given web page and wants to check if the website has other web pages in the pattern being analyzed.

We have reached a point in our process of finding fraud patterns where we have a pattern that worked and we marked all the web pages inside it. If the user considers that they want to reuse the pattern later (for example, they may want to reapply the previously found fraud pattern to a new set of web pages), the user can save the discovered fraud pattern. The *Save Options* menu offers that possibility to the user. Furthermore, it also offers two extra options: *Save Marked Pages*, which as the name suggests, allows the user to save the web page/pages they marked blacklisted; and an option to *Load Filters* that loads and applies a pattern previously saved using our system. All three options are available for the both NBL and all (union of BL with NBL) set of web pages. For the BL set, only the *Save Filters* option is available since we considered that it would be the most useful functionality for someone that started our system by choosing the BL set (it would be interesting to add a functionality to check if all web pages caught by a fraud pattern are already blacklisted, but as we discussed earlier, this is not a high priority feature and so it will be included as future work in Chapter 5).

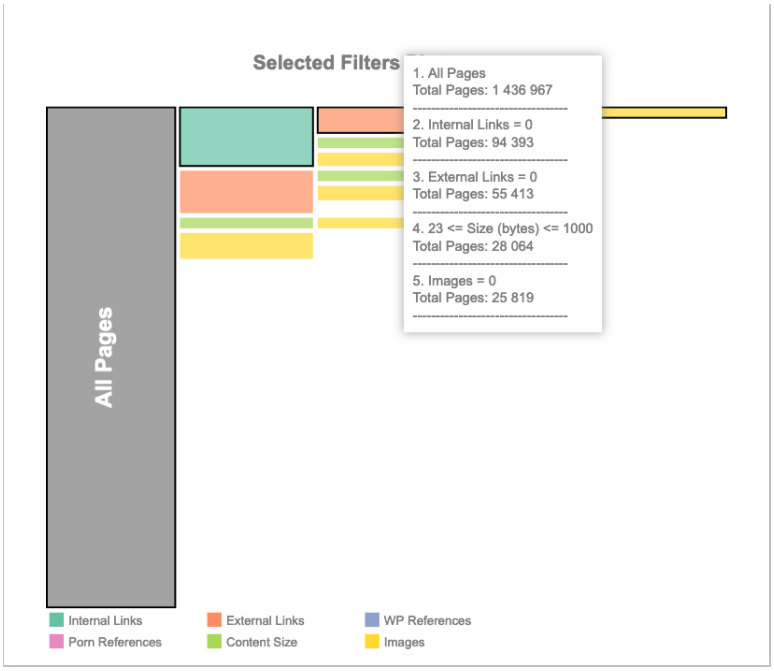


Figure 3.27: *Selected Filters Plot* after loading our fraud pattern in the set of all web pages.

To demonstrate the use of the *Load Filters*, we applied our fraud pattern to the set of all web pages (it is not practical to create a set with new web pages to analyze, so we used this set as an example). First, the user would use the *Change Set* button on the left of our UI and they would see a dialog very similar to the landing (Figure 3.11), with the addition of a *Close* button. It is important to mention that changing the active set of web pages will result in the reset (and consequently, the loss) of all progress

made until that point. That is why we included a *Close* button in the dialog to allow the user to go back without losing any work. Figure 3.27 shows the *Selected Filters Plot*. The combinations are the same as in Figure 3.17 (but with different total values for each subset, as expected, since, the number of web pages in the active set changed), the original chart when the user was investigating the pattern.

We brought this version to Phybbit and they thought our system was complete and met the goal of this research: finding fraud patterns. That being said, the system was ready for the next step: user evaluation. Since we had a system that worked, we needed to know how well it works and what other people think of it. In the next chapter we will explain the user evaluations and the results we obtained.

4

Evaluation

Contents

In Chapter 3, we explained the steps of the iterative and incremental design of our system. Even though Phybbit considered our final version as a workable and useful solution, we still needed to perform a formal evaluation and understand what the strengths are and what could be improved in our solution. The evaluation focused in two different components: usability and utility. In this chapter we will thoroughly describe both usability and utility evaluations procedures as well as the obtained results.

4.1 Usability Evaluation

According to Jakob Nielsen, usability is a quality attribute that assesses how easy user interfaces are to use [29]. To assess how easy to use a given system is, it is important to establish relevant quantitative metrics. In the context of InfoVis, we chose two quantitative metrics: the time it takes for a user to complete a given task and the number of errors they made while performing it [29]. Both these metrics allowed us to measure the efficiency and the efficacy of our system.

4.1.1 Participants

Since usability evaluates how easy to use our UI is, we wanted our participants to focus solely on this quality attribute, which means our participants should not have had previous contact with similar systems that were developed to solve problems in the same context as in this research. However, the participants should have had some contact with UI in general.

With these criteria in mind, we performed our usability evaluation on a total of 20 different participants. Each participant was requested to answer a questionnaire whose main objective was to characterize our test group (see appendix A, Figure A.1 to Figure A.4).

The first couple of questions focused on assessing basic information about our participants. From the 20 participants, 16 of them were males (80%) and 4 were females (20%). Regarding the age group, 14 participants were between 18 and 24 years old (70%), 3 participants were between 25 and 34 years old, 2 participants were between 45 and 54 years old and the remaining participant was between 55 and 64 years old. After these base questions, we wanted to know how familiar our participants were with UI and computer usage in general. To assess this, we asked them how many times they used their computer: 12 (60%) participants claimed they used the computer at least every hour, 5 participants (25%) used it at most every day, 2 (10%) participants only used the computer a couple of times in a week and the remaining participant claimed that they only used the PC once a week.

The remaining questions focused on understanding if the participants had contact with elements of our context (mainly online Ads) and if they had some idea about the online advertisement fraud and its impact in the economy of the companies involved in the online Ads industry. We started by asking if they had ever seen online Ads: only 1 participant had never seen online Ads. The next question assessed

how exposed our participants were to online Ads. Firstly we asked them if they used adblock [30]: only 2 participants said they did not use adblock. For these 2 participants, a new set of questions appeared to better understand their interactions with online Ads. We realized that both participants were not likely to click on online Ads and none of the participants ever ended up on a fraudulent web page in the few times they clicked on them. The remaining two questions focused on the context of advertisement fraud itself. Firstly, we asked the participants to guess what the total cost of advertisement fraud was in the year of 2019: only 3 participants (15%) guessed correctly, the vast majority of the participants (70%) underestimated the cost of advertisement fraud and 3 participants (15%) overestimated the total cost. In our final question, we asked the participants to guess the percentage of traffic of online Ads interactions that did not come from humans, since this is one of the most common fraud technique (not our focus in this research but extremely important in the overall context [31]): only 2 participants (10%) guessed correctly the percentage of non-human actors, the majority (65%) of the participants overestimated the value and the remaining 5 participants (25%) underestimated the correct percentage.

4.1.2 User Evaluation

In the context of InfoVis, it is important that the questions the participants had to solve in the usability evaluation were as close as possible to the specified set of tasks and questions in Section 3.1.1. The order of the evaluation questions is also extremely important: we should not start with the more complex questions, since we did not want our participants to feel demotivated and overwhelmed right at the start of the evaluation. However, we wanted to add another requirement: we wanted to give the participants a sense of completeness while doing the evaluation, i.e. we wanted to order the questions so the participants would observe how a working fraud pattern is found. To do that, we used the found pattern in Section 3.3.1 and ordered the questions as follows:

1. **How many Not Blacklisted web pages have more than 1000 Word Press (WP) references (fraud pattern 1)?**
   This is a simple fraud pattern, that worked as the participants first discovered fraud pattern using our system.
2. **How many web pages not classified as fraudulent, have no internal links, no external links, no images and content size less or equal than 1000 bytes (fraud pattern 2)? Mark all the web pages.**
   The pattern found in Section 3.3.1. More complex than the previous one, since it required more interactions.
3. **What is the percentage of web pages marked as blacklisted from fraud pattern 2?**
   This is a straightforward question that required the participants to simply look at our system's top menu bar.

4. Given all Not Blacklisted web pages of fraud pattern 2, how many of them appear empty when previewing them (consider only the first page of *Table View*)?

In this question we wanted to evaluate the implementation of the verification phase of our system.

5. Consider the first 3 pages of the *Table View*. How many marked web pages appear in fraud pattern 2 and in the subset of web pages with no images and content size less or equal than 1000 bytes (fraud pattern 3)?

The main goal of this question was to evaluate how easy the *Selected Filters Plot* was to use to find smaller fraud patterns generated from other combinations of the filters applied by the participants in previous questions.

6. How many web pages (both Blacklisted and Not Blacklisted) have the fraud pattern 3?

The final question was used to evaluate the possibility of reapplying previously found fraud patterns to a new set of web pages (simulated here by changing the set of web pages).

4.1.3 Apparatus

Our initial plan was to conduct the evaluation in a controlled environment that would consist in a room with only the evaluator, the participant and the required material to evaluate our system. However, when we were about to start the usability evaluation, the world was struck by a global pandemic that forced the national authorities to impose measures of social distancing that completely denied any intention to perform the evaluation as we defined early.

Given the circumstances, we had to adapt the evaluation to be performed remotely. We asked the participants to perform the evaluation in their personal computers (either desktop or laptop), with an internet connection available, so they could communicate with the evaluator and access our system, locally hosted in the evaluator's computer. We needed software that allowed us to communicate directly with the participants. The software should also allow screen share and file share. We chose a set of communication software and each participant chose the one they were more familiarize with: Discord[1], Skype[2] or Google Meets[3]. Finally, each participant would chose their usual web browser to run our system.

Giving so many choices to the participants is far from the ideal evaluation scenario, but with all the restrictions we had in terms of social distancing, this was the best compromise we found so we could still evaluate our system. By allowing the participants to choose both the communication software and web browser, we reduced to the minimum the hassle this evaluation could cause to each participant and minimized the number of mistakes caused by the used communication software.

[1] https://discord.com/new
[2] https://www.skype.com/en/
[3] https://meet.google.com/

4.1.3.A System Usability Scale (SUS)

The SUS is a subjective measure of usability of the system that should be used after the participants completed the tasks they had [32]. It consisted of ten questions scored on a six-point scale, from *strongly disagree* to *strongly agree*. We modified the original five-point Likert scale, since we wanted the participants to choose a side [33] and not remain neutral in their assessment. With this change, the maximum score of each SUS form was 50 (with the original scale it was 40). To convert the SUS score into the final 100 points scale, we had to decrease the multiplicative factor from the original 2.5 to 2. The final score of the SUS ranges from 0 to a total of 100 points, where the average score is 68 [34]. The form used in our evaluation is available in Appendix A (Figure A.5 to Figure A.7).

4.1.4 Procedure

The evaluation procedure started by establishing a call with the participant and checked whether the connection was good enough to communicate or if we needed to start over or change the software in use. The remaining procedure went as follows:

1. Made a brief introduction of the evaluation procedure;
2. Asked the participant to fill the form for user characterization;
3. Fully explained the context of this research, what the problem was and how could our research solve/mitigate the problem;
4. Started a screen share with the participant to introduce them to the UI and the available interactions in our system;
5. Let them ask some questions about the context and/or usage of the system (if they had any);
6. Sent them a file with the questions and a link to the server hosting our system;
7. Asked the participant to start sharing their screen;
8. Asked the participant to start answering the question;
9. Recorded time and number of errors the participant made, until the correct answer was verbally given;
10. Repeated step 8 and 9 for all the questions;
11. Asked participants for qualitative feedback (both what they like/dislike about our implementation and suggestions on how it could be improved);

12. Asked the participants to fill the form for the SUS;

13. Thanked the participant for their precious help and time.

4.1.5 Results

Upon the conclusion of our tests we had three sets of data: response time for each question, the number of errors for each question and the scores of the SUS.

For each set of data, we have included descriptive statistics [35]: mean, median, standard error (SE), standard deviation (STD) and confidence interval (CI). The tables that show the individual results of each participant are represented in Appendix B.

The first step in our results analysis is to look at time and errors individually. Then, we will perform a more extensive statistical analysis in order to formally investigate if the UI is easy enough to use or if it should be further improved. The SUS will have its own section, where we will analyze the score the participants gave to our system. Lastly, we will discuss all the obtained results in our usability evaluation.

4.1.5.A Response Time

Descriptive Statistics						
Statistics	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
Mean	72.7	166.45	19.65	58.6	98.15	105.55
Median	70,5	160	5	52	93	97
SE	8.28	12.71	9.64	5.62	7.52	12.32
STD	37.02	56.84	43.09	25.11	33.63	55.10
CI	17.33	26.60	20.17	11.75	15.74	25.79

Table 4.1: Descriptive statistics for the response time results of the usability evaluation.

Table 4.1 shows the results obtained for the response time in the usability evaluation. Observing the response times, we verified that Q3 was the one that took less time to solve. In contrast, Q2 was the question that took the most time to solve. Q2 and Q6 were the questions where the STD was higher, which means they had a flatter distribution curve. To better visualize the results, we plotted the data into a box plot, in order to have a visual representation of the results (Figure 4.1). The response time box plot also showed us that the distributions for Q1, Q5 and Q6 are visually similar. It also shows that both Q3 and Q6 have the most outliers (2 outliers in each one). However, Q3 was the one where the outliers were the furthest away from the upper whisker of the box plot. This visual finding was corroborated by Table B.1, where we could observe the outliers as being users 1 and 18, with response times of 180 seconds and 96 seconds, respectively. This discrepancy in the values explains why the mean (19.65 seconds) is almost four times higher than the value of the median (5 seconds) in Q3.

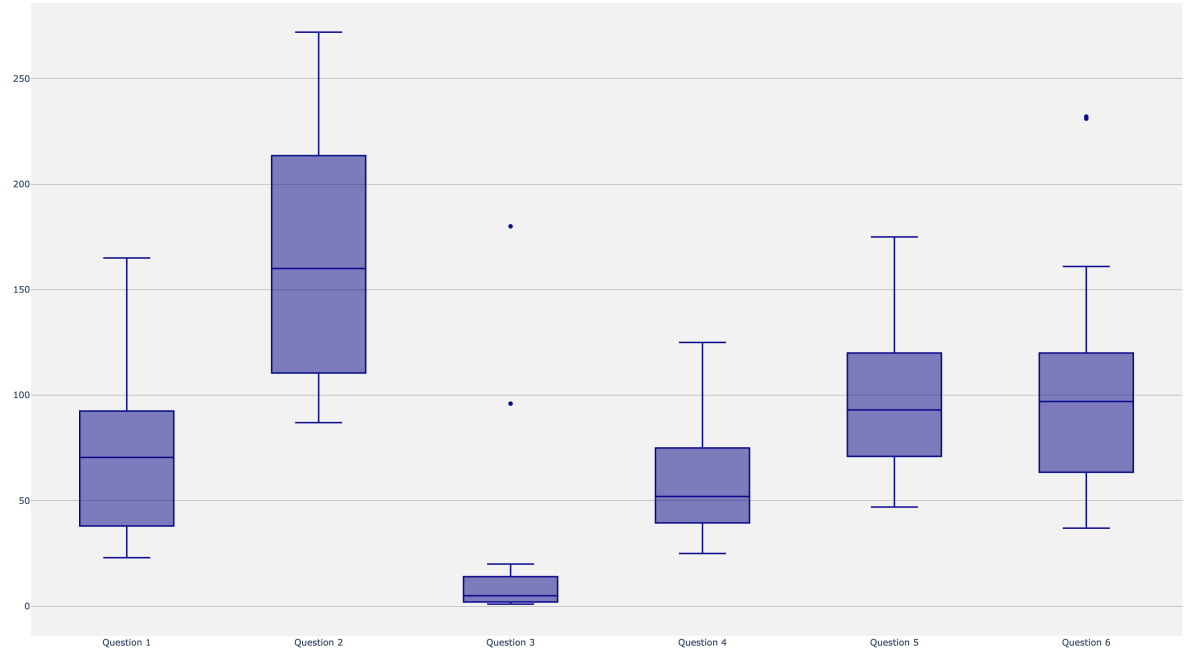


Figure 4.1: Box Plots for each question response time.

4.1.5.B Errors

Descriptive Statistics						
Statistics	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
Mean	1.05	1.65	0.25	0.4	1.1	0.85
Median	1	2	0	0	1	1
SE	0.21	0.28	0.16	0.13	0.19	0.18
STD	0.94	1.27	0.72	0.60	0.85	0.81
CI	0.44	0.59	0.34	0.28	0.40	0.38

Table 4.2: Descriptive statistics for the number of errors of the usability evaluation.

Table 4.2 shows the results for the number of errors in the usability evaluation. Observing the number of errors, we verified that Q2 was the question with a higher number of errors by a considerable margin (mean of 1.65 whereas the second highest mean is 1.1 from Q5). It was also the flatter distribution, since it has the highest standard deviation (1.27). Our analysis also showed us that both user 6 and user 8, did not make any mistake in the entire usability evaluation (Table B.2). Lastly, both Q3 and Q4 presented a mean of 0 errors and median of close to 0, which means they were the questions where the participants made less mistakes.

As we did with the response time, we included a box plot for the number of errors, to visually represent the data in the table. Figure 4.1 allows us to understand that with the exception of Q2, that had an outlier with 5 errors, all the number of errors are bound by 0 errors at the lower bound, and by 3 errors at the

upper bound.

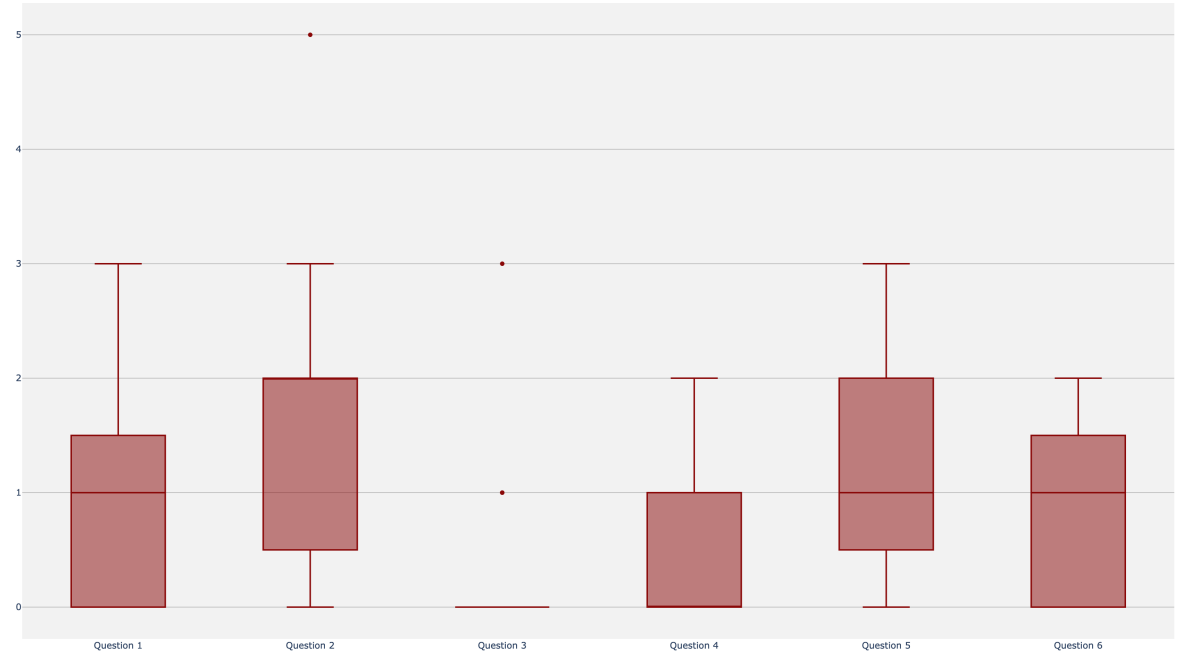


Figure 4.2: Box Plots for each question number of errors.

4.1.5.C Statistical Hypothesis Testing

As we mentioned before, usability evaluation analyzes the easiness of use of a system's UI [29]. However, ease of use is a concept that is subjective and depends on the people that are using the system and their own definition of easy to use, so, we had to use a more scientific method to perform that analysis.

The problem of ease of use can be viewed as if we increase the complexity of a given task in our system, the user can manipulate the UI the same way as in a simpler task and they will not have more difficulties, i.e. the response time and number of errors will have a similar distribution. The statistical null hypothesis will be based on this concept.

Before starting to test the hypothesis, we needed to gather more information about our response time and number of errors distribution. More precisely, we needed to check if the distribution for these metrics per question followed a normal distribution or not. To do that, we applied the Shapiro-Wilk test. The Shapiro-Wilk showed us that all distributions for the number of error on every question did not follow a normal distribution. In terms of response time, all distributions were normal distributions except for Q3 and Q6. The next step in the hypothesis testing was comparing distributions of metrics. In order to compare distributions, we divided the questions of the usability evaluation into three difficulty categories:

1. **Easy Questions: Q4:** We consider Q4 to be the easiest question in our evaluation since it only

required to preview the web pages, which consisted in simply selecting them in the respective table entry of the *Table View* (Figure 3.24).

2. Medium Difficulty Questions: Q1 and Q6: These questions were harder than Q4, because they required more interactions in several elements of the UI.

3. Hard Questions: Q2 and Q5: Both these questions required, respectively, a high number of interactions and a high level of understanding of how the *Selected Filters Plot* worked (Figure 3.17)

Note: We did not include Q3 in any category, since it did not require any direct interaction by the user.

In order to compare distributions we needed to apply a statistical test that depends on the distribution of our metrics: if the metrics followed a normal distribution, we would apply a parametric test, which was Student T-Test; on the other hand, if metrics did not follow a normal distribution, we would apply a non-parametric test, which was the Wilcoxon Signed-Ranked test. Since, we had a few metrics that did not follow a normal distribution, we decided to apply the non-parametric test to all of them.

A – Hypothesis 1 - The complexity of a question does not influence the usage of the UI. To test our main hypothesis, we needed to split it into two sub-hypothesis: one that focused on the response time behavior and another that focused on the number of errors. The rejection of any of these sub-hypothesis results in the rejection of our main hypothesis.

A –.1 Hypothesis 1.1 - The response time has the same distribution regardless of the complexity of the question. A Wilcoxon test was carried out to compare the response time between difficulty categories.

	Statistic	p-value
Q4 vs Q1 (easy vs medium)	65.00	0.140
Q4 vs Q6 (easy vs medium)	14.00	**0.001**
Q1 vs Q2 (medium vs hard)	1.00	**0.000**
Q1 vs Q5 (medium vs hard)	40.50	**0.017**
Q6 vs Q2 (medium vs hard)	4.00	**0.000**
Q6 vs Q5 (medium vs hard)	99.00	0.837

Table 4.3: H1.1 - Results of Wilcoxon-Signed-Ranked test for the response time.

Table 4.3 showed that by increasing the difficulty of the questions, the distribution of the response time had significant differences between each other. The exceptions were the case where the response time distribution was not significantly different for pair Q4 and Q1 ($p = 0.140$) and for pair Q6 and Q5 ($p = 0.0837$). Considering that there are significant differences in the distribution when the difficulty increased, this hypothesis was rejected.

A –.2 Hypothesis 1.2 - The number of errors has the same distribution regardless of the complexity of the question. This sub-hypothesis focused on how the distribution of the number of errors changed with the various question difficulties.

	Statistic	p-value
Q4 vs Q1 (easy vs medium)	24.00	**0.015**
Q4 vs Q6 (easy vs medium)	28.00	**0.048**
Q1 vs Q2 (medium vs hard)	39.00	0.063
Q1 vs Q5 (medium vs hard)	40.50	0.815
Q6 vs Q2 (medium vs hard)	27.00	**0.014**
Q6 vs Q5 (medium vs hard)	50.00	0.205

Table 4.4: H1.2 - Results of Wilcoxon-Signed-Ranked test for the number of errors.

The results of Table 4.4 showed that 75% of the cases where we compared medium with hard difficulty questions did not present any significant differences while all the cases where we compared easy to medium difficulty questions presented significant differences ($p = 0.015$ and $p = 0.048$). Since, one or more cases showed significant difference, the hypothesis 1.2 is also rejected.

Considering that both sub-hypothesis 1.1 and sub-hypothesis 1.2 were rejected, this mean our main hypothesis "The complexity of a question does not influence the usage of the UI" was also rejected.

4.1.5.D SUS scores

Descriptive Statistics											
Statistics	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Total Score**
Min	1	1	2	0	3	4	1	2	1	0	62
Max	5	5	5	5	5	5	5	5	5	5	94
Mean	3.85	4	3.5	3.5	4.45	4.5	3.85	4.1	3.8	3.65	78.4
Median	4	4	4	4	5	4,5	4	4	4	4	83
STD	1.09	1.03	0.76	1.61	0.69	0.51	1.09	0.97	1.15	1.35	10.33

Table 4.5: Descriptive statistics of the results of the usability evaluation SUS.

The last metric that remains to analyze are the results of the SUS. Table 4.5 shows the results for the SUS. Since the scale used ranged from 1 to 6, the minimum score for each question was 0 and the maximum score was 5. It was possible to observe that the questions with the highest STD were Q4 ("I think that I would need the support of a technical person to be able to use this system.") and Q10 ("I needed to learn a lot of things before I could get going with this system."), which means they were the questions in which the participants were more divided in the score they should give our system's UI. The total score was bound by 62 points in the lower bound and by 94 points in the upper bound. The average total score of our system's UI was 78.4, which, according to Bangor et al. [34], is above the average, translating to a "C" (above "Good") score, as shown in Figure 4.3.

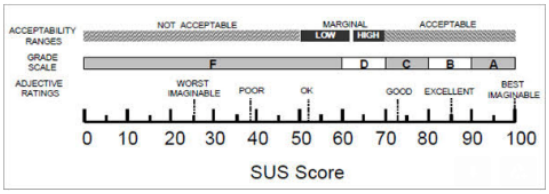


Figure 4.3: SUS scores scale according to Bangor et al.

4.1.5.E Discussion

When we looked at the results of the response time and the number of errors, we noticed that Q2 was the hardest question to solve. However, we have to take into consideration the fact that the usability evaluation was done remotely and depended on the personal computer each participant had. This variability in the equipment helps explain some of the dispersion of the values of response time (and it was especially noticeable when watching the screen share of the participant, while they were trying to solve each question). Another question that caused difficulties with the participants was Q5. Even though the response time was on par with Q1 and Q6, the number of errors was, on average, higher.

The reasons why we noticed this pattern in both Q2 and Q5 are: in the case of Q2, the fact that we employed a brushing selection mechanism on all charts (including bar charts), caused some participants to miss selections in both bar charts (they tried to click on a bar to select a given value) and histograms (where the problem was more noticeable in histograms with a logarithmic scale in the x-axis); and in the case of Q5, the main problem was that participants had trouble in understanding how the *Selected Filters Plot* worked, which is highly likely due to some failure in the evaluator's explanation of its functionality.

The rejection of the hypothesis "The complexity of a question does not influence the usage of the UI" could be explained by the fact the UI itself might need some polish and because of the interactions we used were not obvious at first for the participants (for example, the brushing as selection mechanism in bar charts). This fact could also explain why the STD was higher in Q4 ("I think that I would need the support of a technical person to be able to use this system.") and Q10 ("I needed to learn a lot of things before I could get going with this system.") of the SUS.

Even with these difficulties, the participants rated the UI as being above average and provided exceptional qualitative feedback including comments as "We found the system very intuitive.", "Pretty cool and organized system.", "I can see how this system solves the initial problem." and "The more I use the system, the better and easier to understand it gets."

4.2 Utility Evaluation

Utility is another quality attribute that assesses if the user interface provides the features the user needs [29]. Evaluating utility is different than evaluating usability. The most important feature in this evaluation

is to understand what the participants think about the overall utility of our system.

Since this research is supported by Phybbit, the utility evaluation was performed by two collaborators of Phybbit: Eurico Doirado (CTO) and Gonçalo Pereira (Team Leader and Senior Software Engineer). Both work daily with the context of advertisement fraud and are experienced in finding signs of fraud in websites. With the utility evaluation, we wanted to check whether our participants could find new information that they did not know before or, at least, offer our participants a new perspective about the context of advertisement fraud in websites.

Eurico started by looking at the set of all web pages. He wondered why there were so many web pages with 1, 10 and 100 links. Then, Eurico applied filters for word press references, content size and images. He found an interesting subset of web pages that reminded him of parked domain web pages [36] and a lot of blogs. The parked domain web pages ”should definitely be blacklisted”, Eurico said. The blogs were a different story. Some of them appeared legitimate while others did not, so, Eurico refined his search even more. He reached a subset of web pages with a considerable amount of repeated URLs but different paths, ”This is the common behavior of web pages with WordPress content” Eurico stated. During the evaluation, Eurico considered it was nice to view previous applied filters, i.e. previous subset state in the *Selected Filters Plot*. Another feature Eurico thought to be valuable was the possibility of previewing the web pages without having to leave the interface, ”That’s nice and saves time!”.

Gonçalo took a different approach and decided to investigate the set of NBL web pages. He started by noticing the similarities of the internal link and external link distributions. Gonçalo decided to apply a filter to investigate web pages with less than 10 external links. He found an enormous subset of web pages, with a mix of legitimate and suspicious web pages, so the next step was to refine the search to include a low number of internal links. A subset of about 300,000 web pages was generated with ”very bad content” Gonçalo stated. Gonçalo repeated the process for the web pages with around 100 external links and/or 100 internal links. Here, Gonçalo did not find anything relevant that showed fraud. The same happened when investigating word press references: no clear signs of fraud were found. Investigating the pornographic references revealed a totally different story: with more than 90 pornographic references, all web pages should be blacklisted, since Phybbit’s clients do not want their Ads in adult websites. ”This parameter has a huge impact and we can quickly find potentially blacklisted websites with it” Gonçalo concluded. The same conclusion was obtained when looking at web pages with combinations of low content size and low images count. Gonçalo considered that the main highlight of the system was the *Selected Filters Plot*, ”The selected filters plot is very interesting to quickly understand the dimension of the several filters and its importance in the results I was trying to obtained.” and ”I was not expecting it to be so useful and interesting!”. Gonçalo also praised the *Investigate Interval* functionality by saying ”I was having some difficulties in selecting the exact interval in the violin plots, so I clicked

in the *Investigate Interval* and I was able to easily select the intervals I wanted."

The utility evaluation showed that our system allowed the participants to discover thousands of web pages with highly suspicious content in a matter of minutes. Furthermore, it offered the possibility of viewing how the attribute distributions were in each set, and what the most common values were, giving a new perspective on how similar the web pages were (which can be explained by the fact they were built following a specific template).

4.3 Discussion

The usability evaluation showed us what we needed to improved in our system. We noticed that some of the participants were having some difficulty in understanding the available interactions and how they were connected. This conclusion was supported by the more dispersed SUS score in Q4 ("I think that I would need the support of a technical person to be able to use this system.") and Q10 ("I needed to learn a lot of things before I could get going with this system."). The participants of our utility evaluation also had similar difficulties, especially the first time they tried to make a specific interaction.

Even with the aforementioned difficulties, the participants of the usability evaluation enjoyed the overall user experience, giving a huge amount of positive feedback and an above average classification in the SUS. The participants from both evaluations understood the usefulness of our system. In the usability evaluation, a lot of participants were impressed by the amount of potentially fraudulent web pages they could find when applying the pattern we found in Section 3.3.1. The same feeling was present in the utility evaluation. Being able to quickly find a subset of thousands of web pages that showed signs of fraudulent activity, was one of the features our participants found most helpful when freely exploring our system in the utility evaluation. Furthermore, they both found appropriate the inclusion of the preview of the web page in the UI itself and, one of them, was particular impressed with how advantageous the *Selected Filters Plot* was in the process of analyzing the various subsets of web pages created by the combinations of the applied filters.

The utility evaluation also revealed that is was possible to find large subsets of web pages that showed signs of fraudulent activity, meaning that our system can scale in order to find fraud patterns that included thousands of web pages.

With the results of both the usability and the utility evaluation, it was proven that it is possible to use InfoVis techniques to find fraud pattern in web pages, meaning we fulfilled our main objective. We have selected the most adequate InfoVis techniques to build a system that allows the Phybbit's researchers to find new fraud patterns while offering the possibility of detecting a large number of suspicious web pages at once.

5

Conclusion

Contents

In this document we addressed the problem of finding fraud patterns in web pages. This is a subject not yet very well studied in the context of advertisement fraud. There are tools that can detect and take action against advertisement fraud based on the information of the interactions with Ads themselves (number of clicks, which identity clicked on them, when those clicks happened and more). SpiderAF is one of those tools that also has a feature for detecting fraud in web pages. Its goal is to provide a reliable list of websites that should be blacklisted. This list is part of its *Shared Blacklist* service. Researchers on their team have the arduous task of exploring websites looking for new fraudulent trends, a process that can benefit greatly from the use of InfoVis techniques.

Information Visualization systems are built to help people carry out more demanding tasks, leveraging the use of visual elements to represent the data to be analyzed [17], so, it is logical to consider the application of InfoVis techniques in the context of advertisement fraud, in order to improve the efficiency and efficacy of fraud detection. This is the reasoning behind our main objective in this research: **study the application of Information Visualization techniques in the website analysis context, in order to help find fraud patterns in web pages**.

We developed a solution that utilizes InfoVis techniques in order to detect fraud patterns at a large scale. Our system aims at providing a more efficient and effective alternative to the exploratory process currently being used by SpiderAF's team when researching new fraud patterns. We followed an iterative and incremental design until we reached a solution where the process of finding fraud patterns was clearly defined. The iterative process began by analyzing all attributes from a web page that were collected by Phybbit, through crawling. Each developed prototype helped us reduce the number of attributes considered relevant in this context. Furthermore, they served the purpose of refining the overall process of finding fraud patterns. The idioms used in our system also benefited from the incremental design. We began by including only bar charts to encode every attribute. Then, we turned those bar charts into box plots with heat maps. This solution did not work as intended and so, we improved it with custom made violin plots that were able to encode the same information in a more coherent fashion. The final version of our system leveraged the use of InfoVis techniques to display the distribution of the attributes in charts that supported filtering operations. The combination of one or more filters would originate a pattern. We included a plot called *Selected Filters Plot* that encoded all possible combinations between the filters applied by the user. Once the pattern was found, it was possible to preview each web page and look for signs of fraud. If the web pages showed signs of fraud, then, the user could mark those web pages as blacklisted, which means the investigated pattern was, indeed, a fraud pattern. With our system it is possible not only to detect fraud in web pages, but also save the found fraud patterns so they can be used later to analyze new web pages.

The system was evaluated by both a group of participants that focused their attention on the usability and by two collaborators of Phybbit that evaluated the utility of the system. In the case of the usability, our

participants approved our system with an above average score in the SUS, provided useful feedback about what could be improved and complimented how the overall system was built and how it could solve the initial problem. In terms of the utility, it was proven that our system could find large subsets of web pages that showed signs of fraudulent activity. This conclusion means that introducing our tool to support Phybbit's exploratory processes could help them quickly discover new fraudulent website patterns. It would then result in the blacklisting of their corresponding websites. The newly blacklisted websites would be added to the *Shared Blacklist* which means SpiderAF's clients would, in turn, reap the benefits by being protected against more fraud patterns. As such, we proved that it is possible to use InfoVis techniques to find fraud patterns in a efficient and effective manner, with our system being a step forward in the current process of finding fraud websites patterns.

5.1 Future Work

The future work should focus on improving our current solution. The main feature that can extend our solution would be the possibility of allowing the direct comparison of the similarities of two or more subsets of web pages, either the ones we have as defaults (the BL and NBL set of web pages) or others that could be found by the user when investigating a given pattern.

During the feedback stage of the usability evaluation, the most frequently mentioned suggestions of the participants were: add more interactions to the *Selected Filters Plot*, mainly a dropdown menu to show the several combinations in a list form and an option to undo the last applied filter; add a checkbox option in the table of the *Table View* to mark web pages without opening them; add a help menu in case the user has difficulties with any functionality/interaction. We considered these suggestions to be extremely important and that they would certainly improve our system.

Lastly, we would like to make the system more customizable depending on the user's needs. This means that the user could choose which attributes to investigate based on all the parameters Phybbit collects through crawling. To achieve this, we would need to integrate our system in the companies' infrastructure, leveraging the direct communication with the DB and the more available computational power, in order to make the system more fluid and more capable of performing more complex operations.

Bibliography

[1] S. Eick, J. L. Steffen, and E. E. Sumner, “Seesoft-a tool for visualizing line oriented software statistics,” *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 957–968, 1992.

[2] J.-E. Stange, M. Dörk, J. Landstorfer, and R. Wettach, “Visual filter: graphical exploration of network security log files,” in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*. ACM, 2014, pp. 41–48.

[3] J. Landstorfer, I. Herrmann, J.-E. Stange, M. Dörk, and R. Wettach, “Weaving a carpet from log entries: A network security visualization built with co-creation,” in *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2014, pp. 73–82.

[4] C. Humphries, N. Prigent, C. Bidan, and F. Majorczyk, “Elvis: Extensible log visualization,” in *Proceedings of the Tenth Workshop on Visualization for Cyber Security*. ACM, 2013, pp. 9–16.

[5] ——, “Corgi: Combination, organization and reconstruction through graphical interactions,” in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*. ACM, 2014, pp. 57–64.

[6] M. Alsaleh, A. Alarifi, A. Alqahtani, and A. Al-Salman, “Visualizing web server attacks: patterns in phpids logs,” *Security and Communication Networks*, vol. 8, no. 11, pp. 1991–2003, 2015.

[7] T. Zhang, Q. Liao, and L. Shi, “Bridging the gap of network management and anomaly detection through interactive visualization,” in *2014 IEEE Pacific Visualization Symposium*. IEEE, 2014, pp. 253–257.

[8] S. Yoo, J. Jo, B. Kim, and J. Seo, “Longline: Visual analytics system for large-scale audit logs,” *Visual Informatics*, vol. 2, no. 1, pp. 82–97, 2018.

[9] C. Shi, S. Fu, Q. Chen, and H. Qu, “Vismooc: Visualizing video clickstream data from massive open online courses,” in *2015 IEEE Pacific visualization symposium (PacificVis)*. IEEE, 2015, pp. 159–166.

[10] Q. Chen, Y. Chen, D. Liu, C. Shi, Y. Wu, and H. Qu, "Peakvizor: Visual analytics of peaks in video clickstreams from massive open online courses," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 10, pp. 2315–2330, 2016.

[11] Y. Chen, Q. Chen, M. Zhao, S. Boyer, K. Veeramachaneni, and H. Qu, "Dropoutseer: Visualizing learning patterns in massive open online courses for dropout reasoning and prediction," in *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2016, pp. 111–120.

[12] Z. Liu, Y. Wang, M. Dontcheva, M. Hoffman, S. Walker, and A. Wilson, "Patterns and sequences: Interactive exploration of clickstreams to understand common visitor paths," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 321–330, 2017.

[13] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson, "Matrixwave: Visual comparison of event sequence data," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 259–268.

[14] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao, "Unsupervised clickstream clustering for user behavior analysis," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 225–236.

[15] Business of Apps, "Ad fraud statistics," 2020, [Online; accessed 20-July-2020]. [Online]. Available: https://www.businessofapps.com/ads/ad-fraud/research/ad-fraud-statistics/

[16] Patrick Kulp, "Ad fraud could become the second biggest organized crime enterprise behind the drug trade," 2016, [Online; accessed 20-July-2020]. [Online]. Available: https://mashable.com/2016/06/09/ad-fraud-organized-crime/?europe=true#s7JF3SEGqsqG

[17] T. Munzner, *Visualization Analysis and Design*, 1st ed., ser. AK Peters Visualization Series. A K Peters/CRC Press, 2014.

[18] A. D. D'Amico, J. R. Goodall, D. R. Tesone, and J. K. Kopylec, "Visual discovery in computer network defense," *IEEE Computer Graphics and Applications*, vol. 27, no. 5, pp. 20–27, 2007.

[19] M. Sweiger, J. Langston, H. Lombard, and M. R. Madsen, *Clickstream Data Warehousing*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[20] L. Pappano, "The year of the mooc," *The New York Times*, vol. 2, no. 12, p. 2012, 2012.

[21] I. C. Cuthill, "Color perception," *Bird coloration*, vol. 1, pp. 3–40, 2006.

[22] Wikipedia contributors, "Web hosting service — Wikipedia, the free encyclopedia," 2020, [Online; accessed 02-July-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Web_hosting_service

[23] ——, “Web page — Wikipedia, the free encyclopedia,” 2020, [Online; accessed 02-July-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Web_hosting_service

[24] ——, “ads.txt — Wikipedia, the free encyclopedia,” 2020, [Online; accessed 02-July-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Ads.txt

[25] Mozilla and individual contributors, “¡iframe¿: The inline frame element,” 2020, [Online; accessed 03-July-2020]. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

[26] DESIGN: AlwaysWithHonor.com and David McCandless. RESEARCH: David McCandless, Pearl Doughty-White, Alexia Wdowski, “Colors in culture,” 2020, [Online; accessed 01-August-2020]. [Online]. Available: https://www.informationisbeautiful.net/visualizations/colours-in-cultures/

[27] C. Ware, *Information Visualization: Perception for Design: Second Edition*, 04 2004.

[28] Mark Harrower and The Pennsylvania State University, “Colorbrewer 2.0 - color advice for cartography,” 2020, [Online; accessed 04-July-2020]. [Online]. Available: https://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3

[29] Nielsen Norman Group, “Usability 101: Introduction to usability,” 2012, https://www.nngroup.com/articles/usability-101-introduction-to-usability/,Last accessed on 2020-07-13.

[30] Wikipedia contributors, “Ad blocking — Wikipedia, the free encyclopedia,” 2020, [Online; accessed 13-July-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Ad_blocking

[31] Neil Andrew, “What is click fraud? a complete guide to understanding ad fraud,” 2020, [Online; accessed 13-July-2020]. [Online]. Available: https://ppcprotect.com/what-is-click-fraud/

[32] J. Brooke *et al.*, “Sus-a quick and dirty usability scale,” *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[33] M. J. Fonseca, P. Campos, and D. Gonçalves, *Introdução ao Design de Interfaces*. FCA, 2012.

[34] Aaron Bangor and Philip Kortum and James Miller, “Determining what individual sus scores mean: Adding an adjective rating scale,” 2009, [Online; accessed 13-July-2020]. [Online]. Available: https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/

[35] A. Field and G. Hole, *How to Design and Report Experiments*. SAGE Publications, 2003.

[36] Wikipedia contributors, “Domain parking — Wikipedia, the free encyclopedia,” 2020, [Online; accessed 29-July-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Domain_parking

AppendixA

Thesis Testing: User Information

This form only takes a couple of minutes to complete and all I want you to do is to answer some simple questions. Fear not, I'm not going to ask any personal question.

All data is treated anonymously and no one will know your answers.

Thank you for your help and time!

* Required

1. What is your gender? *

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Prefer not to say
- ◯ Other: _____

2. How old are you? *

Mark only one oval.

- ◯ Less than 18
- ◯ Between 18 and 24
- ◯ Between 25 and 34
- ◯ Between 35 and 44
- ◯ Between 45 and 54
- ◯ Between 55 and 64
- ◯ Between 65 and 74
- ◯ More than 74

Figure A.1: User characterization form: first and second question.

3. How many times do you use your computer? *

Mark only one oval.

- ◯ Once a year
- ◯ Once a month
- ◯ Once a week
- ◯ A couple of times in a week
- ◯ Basically every single day
- ◯ Basically every single hour
- ◯ The computer and I are both parts of the same entity

Thesis Testing: User Information (2/3)

Advertisement (ad for short). Does this name ring you a bell? In this section I want you to know if you are familiar with online ads.

All data is treated anonymously and no one will know your answers.

Thank you for your help and time!

4. Have you ever seen an online ad? *

Definition: an online ad is a notice or announcement present in a webpage with the goal of promoting a produ service, or event.

Mark only one oval.

- ◯ Yes
- ◯ No

5. Do you use some kind of adblock (tool that blocks the ads from showing up in a webpage)? *

Mark only one oval.

- ◯ Yes
- ◯ No *Skip to question 8*

Figure A.2: User characterization form: last question from user information and question about online Ads familiarity.

6. In your opinion, what is the total estimated cost of ad fraud in 2019? *

Source: https://www.businessofapps.com/ads/ad-fraud/research/ad-fraud-statistics/

Mark only one oval.

- ◯ Less than 100 Million Dollars
- ◯ More than 100 Million Dollars and less than 1 Billion Dollars
- ◯ More than 1 Billion Dollars and less than 10 Billion Dollars
- ◯ More than 10 Billion Dollars and less than 30 Billion Dollars
- ◯ More than 30 Billion Dollars and less than 50 Billion Dollars
- ◯ More than 50 Billion Dollars

7. In your opinion, what is the percentage of online ads traffic that comes from non-human actors in 2019? *

Source: https://www.businessofapps.com/ads/ad-fraud/research/ad-fraud-statistics/

Mark only one oval.

- ◯ Less than 10%
- ◯ More than 10% and less than 20%
- ◯ More than 20% and less than 30%
- ◯ More than 30% and less than 40%
- ◯ More than 40%

Thesis Testing: User Information (3/3)

If you reach this point, it means you are used to see online advertising in a regula basis and I want to know more details (just a bit, don't worry)

All data is treated anonymously and no one will know your answers.

Thank you for your help and time!

Figure A.3: User characterization form: advertisement fraud context.

8. How often do you click in ads on a webpage? *

Mark only one oval.

	1	2	3	4	5	6	
Never	◯	◯	◯	◯	◯	◯	Everytime

9. When you click in an ad, how often does the ad take you to a webpage that seems suspicious? *

Clicks ad. "What the hell is this page? That is not what I clicked! This looks like a scam. Lets go back!"

Mark only one oval.

	1	2	3	4	5	6	
Never	◯	◯	◯	◯	◯	◯	Everytime

Figure A.4: User characterization form: online Ads exposure questions for users that do not use adbloack.

Thesis Testing: System Evaluation

Ok, so now that you've complete the test, I only need you to take this survey.

There are only 10 simple questions.
Please, be honest in your answers. Don't be afraid to break my feelings!

All data is treated anonymously and no one will know your answers.

Thank you for your help and time!
* Required

1. 1. I think that I would like to use this system frequently. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

2. 2. I found the system unnecessarily complex. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

3. 3. I thought the system was easy to use. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

Figure A.5: SUS form (1/3).

4. 4. I think that I would need the support of a technical person to be able to use this system. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

5. 5. I found the various functions in this system were well integrated. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

6. 6. I thought there was too much inconsistency in this system. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

7. 7. I would imagine that most people would learn to use this system very quickly. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

Figure A.6: SUS form (2/3).

8. 8. I found the system very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

9. 9. I felt very confident using the system. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

10. 10. I needed to learn a lot of things before I could get going with this system. *

Mark only one oval.

	1	2	3	4	5	6	
Strongly disagree	◯	◯	◯	◯	◯	◯	Strongly agree

Figure A.7: SUS form (3/3).

B

AppendixB

User/Time (sec)	Q1	Q2	Q3	Q4	Q5	Q6
User 1	70	210	180	65	66	60
User 2	52	110	10	90	80	120
User 3	30	130	15	90	100	79
User 4	35	155	20	25	120	95
User 5	78	210	15	50	120	62
User 6	37	90	1	31	65	37
User 7	39	111	3	54	89	82
User 8	23	87	13	41	56	42
User 9	68	200	2	31	175	45
User 10	53	165	1	49	47	107
User 11	86	181	2	34	73	161
User 12	71	224	1	79	77	158
User 13	75	105	8	38	87	99
User 14	99	229	2	80	109	231
User 15	111	272	11	71	133	232
User 16	165	247	7	65	115	120
User 17	123	148	3	49	136	103
User 18	118	106	96	44	69	95
User 19	35	217	2	125	149	118
User 20	86	132	1	61	97	65

Table B.1: Response time results of the usability evaluation.

User/Errors	Q1	Q2	Q3	Q4	Q5	Q6
User 1	1	2	3	1	2	1
User 2	1	0	0	2	0	2
User 3	0	1	0	1	2	0
User 4	0	1	1	0	1	1
User 5	2	3	0	0	3	2
User 6	0	0	0	0	0	0
User 7	1	1	0	0	1	2
User 8	0	0	0	0	0	0
User 9	2	2	0	0	2	0
User 10	0	1	0	0	1	1
User 11	1	2	0	0	1	1
User 12	1	5	0	1	0	2
User 13	0	2	0	0	1	0
User 14	1	2	0	0	2	2
User 15	1	3	0	0	1	1
User 16	3	3	0	1	2	1
User 17	3	2	0	1	1	1
User 18	2	2	1	0	1	0
User 19	1	0	0	1	1	0
User 20	1	1	0	0	0	0

Table B.2: Number of errors of the usability evaluation.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total Score
User 1	4	5	4	4	5	5	5	5	3	2	84
User 2	4	4	3	4	5	5	4	4	4	5	84
User 3	5	5	4	5	5	5	5	4	4	5	94
User 4	2	3	3	3	3	4	4	3	2	4	62
User 5	5	2	2	0	5	5	2	4	2	4	62
User 6	3	4	4	2	4	4	4	5	4	0	68
User 7	4	4	4	4	5	4	4	5	4	5	86
User 8	5	4	3	5	4	4	4	4	4	4	82
User 9	3	5	5	2	5	4	5	5	5	4	86
User 10	4	4	4	4	4	5	5	4	5	4	86
User 11	3	4	2	3	4	4	4	2	4	4	68
User 12	3	4	3	0	3	4	4	4	3	3	62
User 13	4	4	3	5	5	5	3	4	4	1	76
User 14	5	1	4	5	5	4	4	5	5	5	86
User 15	4	4	3	5	4	5	4	2	3	4	76
User 16	4	4	3	2	4	4	4	3	1	4	66
User 17	5	5	4	4	5	4	4	5	4	3	86
User 18	4	4	4	3	4	5	2	4	5	3	76
User 19	5	5	4	5	5	5	1	5	5	5	90
User 20	1	5	4	5	5	5	5	5	5	4	88

Table B.3: Results of the usability evaluation SUS.