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 known fraud patterns. In the ever evolving world of the web and ad 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.

1. Introduction

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) [3]. 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 [10].

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 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 Blacklist² 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

¹https://spideraf.com/intl/en
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 [8]. InfoVis takes advantage of a high bandwidth communication channel humans have: vision. Being able to transmit information through visual elements, allows the discovery of new approaches for problems that cannot be fully comprehended and specified by a computer algorithm [8]. 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 appropriated. 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 finding fraud patterns and, ultimately, providing insights that enables the refinement of the techniques used in their business.

2. Related Work
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
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.

CORG [7] is the next iteration of a research named ELVIS. CORGI solved one of the main issues of its predecessor: the views of two or more log files could be combined for exploration. CORGI offered 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 was scalability. CORGI depended on the web browser performance and, in cases where considerable amounts of log files were loaded, the system might lag and some visualizations might 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).

Zhang et al. [14] 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 1:** Dashboard of Network Awareness Visualization Tool [14].

Figure 1 shows the dashboard of the visualisation. 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 was not scalable. If one looks closely at the charts, they will 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.

LongLine [13] 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 shows a dashboard of LongLine, with the calendar view selected. 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 [11]. MOOC (Massive Open Online Courses) are a particular case where analyzing the clickstream can be useful. 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.

PeakVizor [4] tried to display all the available visualizations in the same dashboard view.

Figure 3 shows a general view of PeakVizor. 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.

DropoutSeer’s [5] main goal is to visualize one of the most critical component of MOOC: students’ dropout rates.

Figure 4 shows the DropoutSeer system. 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 did 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.

MatrixWeave’s [15] 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.

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 5 shows the dashboard of MatrixWeave for the aforementioned datasets. 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, losing expressiveness.

2.3. Discussion
We have decided to evaluate the aforementioned researches based on the following criteria: existence of multiple views; existence of connected views; how scalable the systems was; available exploration mechanisms; and chosen colors, evaluated using a test called Black and White test which is a procedure where all the images from the different approaches are converted into black and white images.

CORG1 [7], Zhang et al.’s [14] approach, PeakVizor [4] and DropoutSeer [5] are the most complete tools because they offered multiple connected views with several exploration mechanisms and retained their expressiveness when all idioms became black and white, but none of them could scale when the dataset had millions of entries and still encode high granularity of information. On the other hand, MatrixWeave was the only one that could scale effectively when the dataset increased, but it offered a single view with limited visualization options and only worked in specific scenarios.

3. System Development
As stated in Section 1, the main goal of this research is to study the application of InfoVis techniques in the website analysis context, in order to help find fraud patterns in web pages.

Before starting to develop 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:

1. Visualize the data distribution in a given set of web pages;
2. Discover fraud patterns (if they exist);
3. Verify if a web page has fraudulent activity;
4. Save fraud patterns and fraudulent web pages.

With the requirements defined, we could focus on establishing what the tasks and questions of our InfoVis system were [8]. Our tasks and questions were approved by Phybbit:

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

Example Question 1: How many Not Black-listed (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 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 (both BL and NBL) have no images and content size less or equal to 1000 bytes?

3.1. Prototypes
We have followed an iterative and incremental design. We developed two working prototypes before reaching our final solution. Each prototype and final solution were informally evaluated by Phybbit. This evaluation was done by Eurico Doiradó (CTO) and Gonçalo Pereira (Team Leader and Senior Software Engineer). The evaluation consisted in a remote meeting where we would describe the current state of the system, explain what the process of finding fraud patterns was and what our next steps would be. After the description, Eurico and Gonçalo would give their feedback about the state of the system.
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. After consulting with Phybbit, we realized that we had too much information on the landing page. So, we began our prototyping stage with a reanalysis of the available data. We excluded some of the indicators we used and added others. The first prototype also solved a problem we had in the concept: mixing data of hosts with data of the web pages themselves.

Figure 6 shows the view of our system, in this case when comparing BL and NBL sets of web pages. The first prototype allowed switching from viewing data for hosts or web pages. It also offered the possibility of investigating what the most common and least common values were for each indicator and analyze each set of web pages individually.

The informal evaluation revealed two main issues with this prototype: we still had information that was not relevant (mainly information about hosts, since it could be extremely inaccurate) and the process of finding fraud patterns was not clear. Phybbit suggested that we manually find a fraud pattern that worked and adapt the process that led us to a functional system using InfoVis techniques.

The process of finding 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 and Phybbit’s feedback for the first prototype, we chose to focus solely on these attributes:

- **Links (external and internal);**
- **Content Size;**
- **Header Content-Type;**
- **Images;**
- **References (word press and pornographic).**

The reasoning behind finding fraud pattern is straightforward: find combinations of web page attributes that lead us to a subset of web pages, in which the majority of web pages have evidence of fraudulent activity. However, considering that there are millions of possible attribute combinations, we decided to choose extreme cases of some combinations of attributes that revealed suspicious activity inside a web page using knowledge we acquired while performing the data analysis. The discovered fraud pattern was:

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

The discovered pattern finds web pages that are either empty, default pages from server applications such as Apache Server\(^3\) or are not active/reachable. 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, so we removed it from our analysis. The process of finding the fraud pattern was the following:

1. Select attributes that could show suspicious activity in a web page;
2. From the chosen attributes, investigate their values in the NBL set and look for fraudulent activity;
3. Combine attributes;
4. Verify if the web pages found, actually showed evidence of suspicious activity;

Since we had a process that could effectively find fraud patterns, the next step was to develop a prototype that included the improvements from the first prototype and incorporated the process itself.

In the second prototype, we included a redesigned landing page: Figure 7(a). The landing page showed the total number of web pages in each set, allowing the user to visualize the differences in each set and choose which to investigate. The main page of the system was divided in two (Figure 7(b)): on the left side there was the sketch of a chart that would exist to encode the fraud pattern the user was investigating; on the right side, the user could choose which attribute to analyze and then, the charts that encoded the attribute information, would appear.

The main reasoning about the way we divided the UI is to allow the user to apply filters in the charts on the right, where the combination combination, i.e. the pattern, would be visible in chart on the left. Then, the user could preview the web

\(^3\)https://httpd.apache.org/
Even though the chart on the left was not fully developed and the process of finding fraud pattern was not yet fully implemented, we showed the prototype to Phybbit in order to receive feedback. Phybbit thought the chosen attributes were more informative than the ones used in the previous prototype. However, there were two main issues: the User Interface (UI) itself had too much wasted space and the chosen charts had some problems, mainly the box plot with the heatmap, since it was not possible to accurately view all the values.

3.2. Final Version

Before improving the process of finding fraud patterns and solving the problems with the charts, we had to redesign the UI.

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 its corresponding charts; the UI needed a region to place the results of the filters made by the user.

Figure 8 shows our system's final UI. The menu bar on the left, had the attributes that could be chosen by the user. The UI had space for two charts on the left that would encode the information for the select attribute: one overview chart, that encoded information for all the selected set; and one chart for an interval of interest. On the right, we had the space for the chart that would show the combinations of the applied filters. It was also possible to switch to the Table View, where the user could preview the web pages and mark them as BL if they showed signs of fraudulent activity. The top bar showed the statistics for how many web pages were being analyzed in relation to the total set of web pages. The user also had the option to save the marked web pages and/or found pattern as well as load previously found fraud patterns.

We kept the UI landing page of the previous prototype, only changing the background color to white, and added an option to select all web pages, regardless of their classification.

One of the problems of the previous prototype were the charts. To solve the problem of the box plot with heatmap, we decided to transform it into a modified violin plot. A simple violin plot could not be used due to the high variety of values for each indicator, so we adapted it to work with our data. We first built a histogram with a logarithmic scale and then applied a smoothing function to it. We also kept the box plot to help the user to identify the values of the quartiles. Then, we also needed a chart to encode an interval of interest, i.e., an interval that could show signs of suspicious
activity. We defined a default interval of interest for each attribute based on data studying we did in the previous prototypes. However, we also included an option for the user to change the default interval called Investigate Interval. We chose a histogram to encode the interval of interest, since it worked well with the huge variety of values that the interval could have. Both charts for the internal links attribute of the NBL set of web pages are shown in Figure 9, as well as the Investigate Interval option. Since we have, for example, two types of links, each one has its own violin plot. The same happens for the interval of interest chart, where the user can use the dropdown menu on the top right to switch the type of the attribute and view the corresponding interval of interest.

On the right side of the UI we had space for a chart we called Selected Filters Plot. The main function of this chart was to encode the filtering results. Once the user finds some value they think could reveal something suspicious in a web page, the user needs to view the subset of web pages that have the selected value. We developed a solution that allowed the user not only to analyze the resulting subset of web pages of the applied filters, but also of other filters combinations. All this information was stored in a custom structure.

Figure 10: 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).

Figure 11: Final version of the Selected Filters Plot, with the corresponding legend, for the aforementioned fraud pattern.

Once the user had a pattern they thought was worth investigating, they could switch to the Table View, where it was possible to view the URL of all web pages from the chosen pattern and preview each one of them individually. The general view of the Table View is represented in Figure 12.

Figure 12: In Table View, the user can analyze the web pages that match the selected pattern and verify its contents.

3.2.1 Interactions

As we mentioned before, the user should use the charts on the left to apply filters. Selecting an individual value or interval of values is always done using a brushing mechanism, independently of the chart where the selection occurs. Once the filter is applied, the selected portion of the chart changes color to match the corresponding color of the legend of the Selected Filters Plot (Figure 11). In the case of the histograms for the interval of interest, it is possible to click on a bar and analyze all the values inside the interval. The white area around the elements of a chart can be used to do reset actions. So, if the user wants to clear the brush,
they click once in the white area. If the user wants to reset the state of a given chart, they can double click the white area of the chart and it will go back to its default state.

In the Selected Filters Plot, hovering over each rectangle, reveals all the filters that led to that subset and clicking on one of them, will select all its web pages, which will appear in the table of the Table View. The reset of this chart is done by double clicking the white area.

In the Table View, the preview of a web page can be done by clicking on the corresponding table entry and its preview will appear on the bottom or the user can click on the symbol next to the path column and open the web page in a new browser tab. If the user finds signs of fraud, they can use the buttons on the bottom to mark web pages individually or to mark all web pages within the pattern as BL. Table View also offers a search mechanism, where the user can search for a given website and check if any of its web pages are in the selected pattern.

Once the user is satisfied with the found fraud pattern and/or with the web pages they marked as BL, they can use the Save Options menu to save their findings. The Save Options also allow the user to load a previously found fraud pattern and apply it to the set of web pages they are investigating.

4. Evaluation
A system is considered useful if it has easy to use features that are needed to fulfill the objective of a given system [9], so the evaluation of this research should focus on usability and utility, respectively.

4.1. Usability Evaluation
According to Jakob Nielsen, usability is a quality attribute that assesses how easy user interfaces are to use [9]. In the context of InfoVis, we chose two quantitative metrics: the time it takes for a user to complete a given task (which is an efficiency metric) and the number of errors they made while performing it (which is an efficacy metric) [9].

The usability evaluation consisted on recording the time and number of errors each participant did while answering the following questions:

- Q1: How many Not Blacklisted web pages have more than 1000 Word Press (WP) references (fraud pattern 1)
- Q2: 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.
- Q3: What is the percentage of web pages marked as blacklisted from fraud pattern 2?
- Q4: 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)?
- Q5: 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)
- Q6: How many web pages (both Blacklisted and Not Blacklisted) have the fraud pattern 3?

We also asked each participant to rate our UI using System Usability Scales (SUS) [2] after completing all the questions.

Upon 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 [5]: mean, median, standard error (SE), standard deviation (STD) and confidence interval (CI).

Observing the response times, we verified that Q3 was the one that took the least 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.

In terms of the number of errors, we verified that Q2 was the question with a higher number of errors. It was also the flatter distribution, since it had 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. Lastly, both Q3 and Q4 presented a mean of 0 errors and median of close to 0, which mean they were the questions where the participants made the least mistakes.

The problem of ease of use can be viewed as such: 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. To better analyze the results from the usability evaluation, we defined a null statistic hypothesis:

- Hypothesis 1: The complexity of a question does not influence the usage of the UI.
- **Hypothesis 1.1**: The response time has the same distribution regardless of the complexity of the question.

- **Hypothesis 1.2**: The number of errors has the same distribution regardless of the complexity of the question.

The next step was to divide the questions of the evaluation into difficulty categories. Then, we applied the Shapiro-Wilk test and verified that some distributions did not follow a normal distribution. Given this finding, we decided to apply the Wilcoxon Signed-Ranked test to compare all the distributions. Once the comparison was finished, we verified that by increasing the difficulty of the questions, both the distribution of the response time and number of errors had significant differences between each other. This means, each sub-hypothesis was rejected, and so, the main hypothesis was also rejected.

Regarding SUS results, 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 these were the questions in which the participants were more divided on the score they should give to our system’s UI. The average total score of our system’s UI was 78.4, which according to Bangor et al. [1] is above the average, which translates to a “C” (above “Good”) score.

The rejection of the hypothesis "The complexity of a question does not influence the usage of the UI" could be explained by the fact that the UI itself might need some polish and because the interactions we used were not obvious at first for the participants. This fact could also explain why the STD was higher in Q4 and Q10 of the SUS.

Even with these difficulties, the participants rated the UI as being above average and provided positive feedback such as "I can see how this system solves the initial problem."

4.2. Utility Evaluation

Utility is another quality attribute that assesses if the user interface provides the features the user needs [9]. Evaluating usability is different than evaluating usability. The most important thing 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).

Eurico applied filters for word press references, content size and images and reached a subset of web pages that reminded him of parked domain web pages [12] and a lot of blogs. The parked domain web pages "should definitely be blacklisted", Eurico said. Eurico also investigated a subset of web pages that were blogs. 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. Eurico found two functionalities of our system especially useful: being able to go to a previous state with the Select Filters Plot and being able to preview the content of the web pages without leaving the UI.

Gonçalo followed a different approach and investigated values he found intriguing just by looking at the charts. Gonçalo reached, first, a subset of about 300k web pages with "very bad content", Gonçalo stated. Then, Gonçalo decided to investigate pornographic references, which revealed a totally different story: with more than 90 pornographic references, all web pages should be blacklisted. "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 was surprised by how useful the Selected Filters Plot was and praised the Investigate Interval functionality, since it allowed him to correctly select the desired interval of interest.

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.

4.3. Discussion

The participants of the usability evaluation enjoyed the overall user experience, giving positive feedback and an above average classification in the SUS. The participants from both evaluations understood the usefulness of our system.

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

With the results of both usability and 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.

5. Conclusion

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 [8], 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.

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. 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 regarding usability and utility. In the case of the usability, our participants approved our system with an above average score in the SUS. 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 an 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 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.

During the feedback stage of the usability evaluation, the participants suggested us some features that could be improved. We should take into consideration all suggestions in order to improve even further our system.

Lastly, we would like to make the system more customizable depending on the user needs. This means 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 prone to perform more complex operations.

References


