Dissertation Type: Research

DEPARTMENT OF COMPUTER SCIENCE

Resource Networks of Pet Scam Websites

Benjamin Price

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Thursday 14th May, 2020

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of BSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Benjamin Price, Thursday 14th May, 2020

Contents

Executive Summary

This paper looks at identifying links between scam websites involved in the fraudulent selling of pets. We will look at a range of techniques that compute the pair-wise similarity of pet scam websites, and use them to construct resource networks that cluster pet scam websites.

> My research hypothesis is that it is possible to cluster pet scam websites based on their shared resources.

As part of the project, I have achieved the following:

- I developed a scraper in Python that crawled more than 11,000 known pet scam websites and downloaded 1,380 of them.
- I analysed 1,380 pet scam websites, compared them to each other using a number of techniques, and clustered them into groups of websites based on their shared resources.
- I used off-site data in the form of WHOIS to validate these clusterings.
- I created a website that hosts an interactive graph which allows users to explore the resource network and see the connections between pet scam websites. The website can be found at `petscams.herokuapp.com`.

Acknowledgements

I would like to thank my supervisor, Dr Matthew Edwards, for all of the guidance he has provided me, and the team of volunteers at PetScams.com for maintaining a public list of pet scam websites.

Chapter 1

Introduction

A pet scam website is a fraudulent website that claims to sell pets. Scammers will create a website that appears to be a legitimate seller of pets, and advertise it on social media and ad websites. Scammers will attract potential victims by advertising pets for far less than the market price. Their aim is to direct potential victims to their website and to get them emotionally invested in a fictitious pet. These fraudulent websites often appear to be legitimate at first glance. Many claim to be associated with real organisations such as the International Pet and Animal Transportation Association (IPATA), and some will have testimonials from previous customers. The website will showcase the pets for sale and have contact information that allows victims to message the scammers so they can purchase the pet. If the victim chooses to purchase the fictitious pet, the scammer will only accept non-refundable payment methods such as Western Union and MoneyGram, which makes it impossible for victims to recover their lost money. However, once the victim has paid the money the scam is not over. Pet scammers have a number of tricks they use to further extort money from their victims who are now emotionally and financially invested in a fictitious pet.

Scammers will create fraudulent pet delivery websites and once a victim has paid for the pet, he or she will be given a fake tracking number and the URL for the fake delivery website. Here, the victim can track the status of the delivery of their pet. Shortly after the purchase, troubles with shipping will arise that can only be resolved by the victim paying the scammer more money. These include logistical and medical issues such as a pet being stuck in customs, or needing emergency veterinary care. The sunken-cost fallacy along with an emotional message to the victims explaining how their poor pet is stuck somewhere or is ill, persuades the victim to pay. Additional fees can be also be created after the initial purchase such as fees for vaccinations or a ventilated cage. It does not matter how many additional fees the victim pays, the scammer will continuously make up new ones to further exploit them. If the victim becomes apprehensive about paying more money, then the scammer can threaten to get law enforcement involved. This can frighten vulnerable people into cooperating. The scam ends when the victim either runs out of money, or realises that they have been scammed.

The number of complaints related to pet scams that organisations such as the Better Business Bureau (BBB) have received are increasing every year. In the three year period from 2017 to 2019, pet fraud complaints to the BBB increased by 39% from 4,664 to 6,466 a year. Victims usually lost between $100 and $1,000, although some lost as much as $5,000. The majority of victims are from the USA and are in their 20s and 30s. This is due to this age group growing up with the internet which makes them more likely to be open to the idea of purchasing a pet online, rather than visiting a local breeder [3].

While some delivery websites represent fictional companies, others take advantage of well-known brands and either pretend to be associated with real companies, or pretend to be the companies themselves. Since these websites are similar to, or exact copies of, legitimate transportation companies' websites, it can be difficult for victims to realise that they are fraudulent. In 2017, Delta Airlines filed a federal lawsuit in the USA against a number of fraudulent delivery websites associated with pet scams, including `DeltaPetTransit.com` and `DeltaPetAirways.com` for breaching their copyright [14]. These sites were designed to look similar to the legitimate Delta Airlines website and even used their trademarked logo in order to trick victims into thinking that they were paying Delta instead of the scammers.

Pet scam websites are often mass-produced in order to target as many types of pets as possible. Sites are taken offline by authorities once victims report them, only to be re-hosted under a new domain name. Online tools designed to make websites quickly and cheaply make this process even easier for the scammers. Websites will be made as cheaply as possible to minimise operating costs, so they often

duplicate images and HTML structure which can then be used to identify links between them. Many will also use services such as WhoisGuard™ by Namecheap Inc. in order to protect their identities, which makes it difficult for law enforcement to identify the perpetrator(s). The pet scam websites can pretend to not be associated with the delivery website, and then the delivery website can be taken down, whilst the pet scam website continues operating.

There have been effects by organisations to keep track of the names of people, websites and emails involved in pet scams, so that potential victims can be warned. PetScams.com is a website run by volunteers that is dedicated to maintaining and hosting the largest public list of pet scam websites. Users are able to report websites via an online form. Volunteers working for PetScams.com will then read this form and decide whether or not the complaint is legitimate. If enough legitimate complaints are made, the website is added to the appropriate list of scam websites. They have two lists of URLs. One for those fraudulently selling pets, we shall refer to these as pet scam websites, and one for those fraudulently delivering pets which we shall refer to as delivery scam websites. This paper uses the two lists of URLs on the website as a source of known pet scam and delivery scam websites.

The International Pet and Animal Transportation Association (IPATA) is another website that hosts a list of URLs belonging to reported pet scam websites, as well as phone numbers and email addresses. Many pet scam websites will falsely claim to be members of IPATA in order to appear more legitimate to victims, hence why IPATA make an effort to display websites that they are not associated with. Many of the URLs on IPATA's list of scam websites, also appear on PetScams.com. However, the IPATA list is only updated every few months, whereas PetScams.com updates its list several times a week. There also aren't as many URLs and they are not classified into pet scams and delivery scams. For these reasons, this project will only look at pet and delivery scam websites identified by PetScams.com.

One group of researchers have created an extension for email applications that attempts to detect pet scam emails [15]. They created a database of keywords that are more likely to appear in pet scam emails. There were a total of 66 keywords which were found by analysing 50 real pet scam emails. Whenever a user receives an email, the text content is extracted and each word is matched against the keywords. A threshold on the percentage of keywords that appear in the email determines the likelihood of the email being a scam, which is then displayed to the user. One limitation of this system is that it does not take images into consideration. Furthermore, the system has a heavy reliance on the small number of keywords that were found by only looking at a small number of scam emails. If a scammer was aware of this publicly available list, then they could make efforts to reduce the number of keywords their future scam emails contained.

If we visit some of the websites on the PetScams.com list, we begin to notice similarities between them. For example, some of the testimonials on different websites are almost exact copies of each other. The only difference is the name of the pet and website mentioned. There are also websites with the same image of pets used. These similarities suggest that multiple websites are made by the same person or group of people. Scammers who find successful techniques and methods for scamming people will want to reuse them on their next website.

There has been no research into finding links between pet scam websites. This paper aims to fill that gap. We will look at four different ways of finding similarities between pet scam websites, and compare clusterings of websites based on these similarities. Plotting the clusters of pet scam websites will allow us to see groups of websites and the types of similarities they have. We shall refer to these graphs as resource networks, since the edges between nodes represent shared resources. We will use off-site data from WHOIS to validate as many of these edges as possible, since not all websites have WHOIS data available. This paper also looks at developing a website that allows users to explore the resource network of pet scam websites.

If a large cluster of websites that were all created by the same criminal(s) was identified, then law enforcement would be able to focus their limited resources on this cluster rather than going after a scammer responsible for one or two sites, in order to have a greater impact. Furthermore, a system that can find and visualise the resource networks of pet scam websites could be applied to additional types of scam websites.

Chapter 2

Technical Background

This chapter looks at techniques that have been used to identify links between scam websites. Although there has been little research into pet scam websites, research into other types of scam websites is useful as the methods used can be applied to this domain.

There are a number of methods that have been used to identify links between scam websites to varying degrees of success. Once links between known scam websites have been found, it is then possible to cluster them into groups based on their similarity. The most trivial method of linking websites is by looking at the hyperlinks on one website that link to another. A small number of pet scam websites will have links to other pet scam websites that sell different breeds of animals, but most choose not to link to other websites as once one website is identified and taken down, the linked websites would also be taken offline. Therefore, we will also look at other methods that have been used to identify links between known scam websites, which look at shared resources including images, text, and HTML structure.

2.1 Perceptual Hashing

A hash function is an function that maps a binary string of arbitrary length to a binary string of fixed length [12]. Hash functions can therefore take images as an input and output a binary string representing this image. Cryptographic hash functions used in generating digital signatures are sensitive to small changes in inputs. Therefore, two images which are the same size but only differ in one pixel will have two very different hash values.

(a) Hash: 84d99927f632316d

(b) Hash: 84d99967d632316d

Figure 2.1: Example showing that the the perceptual hashes of similar images are also similar.

Perceptual hash functions differ from cryptographic hash functions in that they are designed such that two similar bit strings will result in a similar hash value. The greater the difference in bit strings, the greater the difference in the hash values. The perceptual hashing algorithm extracts features from the given image to generate a hash value or a fingerprint. The fingerprint is robust against changes in brightness, contrast and scale [19]. Figure 2.1 shows how changing a small region of pixels will produce a similar hash value. Images replicated across websites will differ slightly due to compression and different file formats, so perceptual hashing provides a way of generating a fingerprint for each image which can be used as point of comparison.

There are number of ways to implement a perceptual hashing algorithm. The perceptual hashing algorithm that we use later in this paper works as follows. First, the image is reduced to 32x32 pixels and converted to greyscale. Then the discrete cosine transform (DCT) of the image is calculated, which

creates a 32x32 value. The 8x8 values in the top-left quadrant of the DCT have the lowest frequency, and the mean of these values is computed. Finally, the hash is created by setting each bit of the hash to be 0 or 1 depending on if the corresponding value from the 8x8 DCT is above or below the mean DCT value. This algorithm generates a 64 bit hash value. Although this example used a 32x32 image, the algorithm can accommodate any sized image as long as it is square and a power of 2. When creating perceptual hashes that will be compared, we can use larger squares to create larger hashes, which means that they are more sensitive to the differences between images.

Hashes, or image fingerprints, can be compared to each other by using the Hamming distance to determine how similar they are. The Hamming distance between two binary strings of the same length is the number of times where two bits at the same position differ. A smaller hamming distance means a greater similarity, and a threshold can be applied to determine if two images are the same.

Perceptual hashing has been used in multiple fields as a method for detecting images. A browser plug-in called SpoofGuard was developed by anti-fraud researchers to protect users from giving away personal information to phishing websites [16]. When a user visits a webpage that asks for personal information, SpoofGuard hashes all the images on the webpage and compares them to a database of known commercial logo hashes. If there is a match, it will then check to see if the URL of the website is associated with that company. If not, users are warned that they may be on a phishing website.

Perceptual hashing has also been used to find groups of fake online dating profiles that use the same images [8]. Furthermore, techniques similar to perceptual hashing that also generate a fingerprint representing the displayed contents of web pages - such as image shingling - have been used to identify duplicate websites [1]. Phishing websites are visually similar to legitimate websites so this is an effective technique.

Considerable effort goes into preventing websites from hosting illegal content such as child pornography. Image recognition software that uses perceptual hashing has been used to detect these kinds of images [20]. Such systems use a content-based image retrieval system that allows for hashes to be quickly compared against a database of known hashes. Furthermore, a group of researchers have used perceptual hashing to identify TOR websites [2].

2.2 Textual Similarity

There are multiple methods for computing the textual similarity between the text content of two websites. The simplest is to compute the Jaccard index. The Jaccard index between two sets A and B is defined as $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ and will always be a value between 0 and 1 inclusive, where a higher value indicates greater similarity between the two sets. The Jaccard index is also known as the intersection over the union (IOU) and a distance measure can be computed from it. The Jaccard distance measure between two sets A and B is defined as 1 - $J(A,B)$, and will also always be a value between 0 and 1 inclusive, where a smaller value indicates a greater similarity. The text content of a website can be placed into a set and the pairwise similarity of the two websites' text content can be computed. The more similar a pair of websites are, the greater the Jaccard index between them will be. Using this similarity metric is effective when websites are near duplicates of each other, since the majority of the text will be identical. Multiple researchers have used it to cluster groups of similar websites by applying a threshold such that if the Jaccard index is greater the threshold (or the Jaccard distance is less than the threshold), then the two websites belong to the same cluster. It has been included as one of a number of features used in the clustering of spam websites [4] and in the detection of duplicate websites [17] and documents [21]. It has also been used successfully in the detection of plagiarised documents, as plagiarised documents will share a lot of the same words [10].

However, if scammers do not duplicate entire websites and instead only copy-and-paste a sentence or two, then the Jaccard index values will be close to 0, due to there being a small similarity. To overcome this limitation, the longest common substring between two websites can be computed instead. Like with the Jaccard index, a threshold can be used to determine if two websites belong to the same cluster, but rather than the threshold representing the minimum similarity, it instead represents the minimum longest common substring. One group of researchers have used the longest common substring to cluster fake online dating profiles [8].

2.3 HTML Similarity

Several researchers have looked at the HTML structure of websites in order to cluster them. There has been a number of different methods used to compute the HTML similarity between pairs of websites.

Frequency analysis of HTML tags is a method that has been used in multiple papers to cluster websites. Drew et al. developed a system that records the frequency of HTML tags in each spam website and computes the Jaccard index between pairs of websites. They then set a threshold on the Jaccard index and websites that meet that threshold are clustered based on their similarity [4]. They found that combining HTML structural similarity with textual similarity improves the results, which suggests that our system for finding links between pet scam websites should consider multiple features too.

One group of researchers have compared three different methods for computing the similarity of HTML structure between pairs of websites. They measured the performance of each algorithm, and tested them on two samples of websites. One with unrelated websites, and the other with websites that had been partially duplicated. The three methods were computing the frequency analysis of HTML tags, finding the longest common sub-tree and computing the ratio of HTML paths both websites have. They found that computing the frequency analysis of HTML tags was both the fastest method and the better method for finding websites with similar HTML structure [22].

Another group of researchers have developed a system to cluster spam websites that were created using the same tools and HTML templates, by generating a fingerprint of each webpage based on the HTML noise [18]. This is a different approach from other systems which usually remove this noise in a pre-processing step.

2.4 WHOIS

Whenever a new domain name is registered, the person registering it must provide contact information to the registrar including their name and address. The registrar will then keep a record of this alongside other information such as the creation and expiration date of the domain.

The WHOIS protocol was created in 1982, and since 1998 it has been maintained by the Internet Corporation for Assigned Names and Numbers (ICANN). It is a protocol that can be used to retrieve information about domain names. Despite the word being capitalised, it is not an abbreviation. Each registry will have a database storing information which can be retrieved by making a WHOIS request. The information contained in a WHOIS record depends on the top level domain name (TLD). Newer and smaller TLDs provide less information.

Many registrars will offer a service to hide the contact information in the WHOIS data for a fee, which gives the person registering the domain name privacy. However, some registrars offer this service for free in a bid to attract new customers. Namecheap Inc. is a popular registrar that provides this service for free, called WhoisGuard™. Many people who use Namecheap Inc. will then use WhoisGuard™ to hide their contact information from anyone performing WHOIS lookups. This can make it difficult to identify who owns the domain.

Since 2010, ICANN have looked at creating a successor to WHOIS that will standardise queries. This system is called the Registration Data Access Protocol (RDAP). It can access the same information as WHOIS but returns data in a JSON format. This format is less human-readable, but it was designed so that it could be built upon more easily than WHOIS.

WHOIS data has been used to cluster scam websites. One group of researchers analysed thousands of WHOIS records and clustered websites based on similarities in their WHOIS data [13]. They observed that many of the fields had been modified to protect the website owner's privacy. In addition, other researchers have identified websites with intentionally invalid WHOIS data [9] [7]. Criminals can conceal their identity by providing incorrect WHOIS information, which then makes it more difficult for law enforcement to identify them.

2.5 Dunn Index

The Dunn index is a metric used for evaluating clusters [5]. It is a measure of how compact and well-separated a clustering is. It is classified as an internal clustering validation metric, rather than an external clustering validation metric, as it only looks at the data within the clusters and does not use an external reference. It is computed by performing $DunnIndex = \frac{MinimumSeparation}{MaximumCompactness}$ [6]. The minimum separation value is the minimum inter-cluster distance, which looks at the minimum distance between all

pairs of clusters; the aim is to maximise this value, since this means that the clusters are well-separated. On the other hand, the maximum compactness value looks at the intra-cluster distance within all clusters. The cluster with the biggest diameter is the least compact and will have this value. By minimising this value, we end up with clusters that are very compact. Overall, the aim is to maximise the Dunn index, and by comparing the Dunn index for different clusterings, we can find the one with the most compact and well-separated clusters.

The main disadvantage with the Dunn index is that it is computationally expensive, $O(n^2)$. It requires us to compute the pairwise distance between all pair of websites within each cluster, and then the pairwise distance between the websites in all pairs of clusters. Furthermore, the Dunn index is sensitive to noise as it only represents the minimum separation and maximum compactness. This means that one outlier cluster can significantly impact the Dunn index [11].

There are a number of distance measures that can be used when calculating the inter-cluster and intra-cluster distance, such as the single linkage, complete linkage or average linkage distances. These use the closest, furthest, and average distances respectively between two entities in two different clusters.

Chapter 3

Project Execution

There were three main stages to my project execution. The first stage looked at creating a tool to automatically scrape pet scam and delivery scam websites from a list of URLs hosted on PetScams.com. The second stage looked at finding connections between pairs of scam websites based on four features and generate resource networks that cluster them based on these connections. The clusterings are then validated using off-site data from WHOIS. The third stage was to develop an interactive graph that would allow users to explore the complete resource network of pet scam websites, and then host this graph on a website to share the results.

During development, I used Git for version control. The websites I downloaded were saved in a directory, but all the other data such as connection information was stored in a PostgreSQL database. I chose to use the Python 3 programming language as it is quick to prototype with. The Python scripts were run on my laptop which has a quad-core i5-8265U CPU and 8GB RAM. These specifications should be considered whenever any runtimes are provided.

3.1 Scraping Scam Websites

Before any analysis of pet scam websites could begin, I needed to acquire a data set of websites. The website PetScams.com hosts a list of pet scam websites, as well as delivery scam websites involved in pet scams, that users have reported. When reporting a website, users must fill out a form. Volunteers working for PetScams.com will then read this form and decide whether or not the complaint is legitimate. If enough legitimate complaints are made, the scam website is added to the list. Since 2017, over 9,000 pet scam websites and 2,000 delivery scam websites have been identified.[1]

Figure 3.1 shows how the number of pet scam websites identified has varied over time. In 2020, there has been a notable decrease in the number of websites identified. It is unclear if this is due to fewer pet scam websites being created, or if it has been harder to detect pet scam websites, and therefore fewer are identified. Overall, there are fewer delivery scam websites than pet scam websites. This suggests that not all pet scam websites have a corresponding delivery scam website. It is not clear which delivery scam websites are associated with which pet scam websites until a victim has paid money and is told who will be shipping the pet.

3.1.1 Acquiring scam website URLs

In order to get a local copy of the list of URLs, I created a Python script that scraped the two lists of pet scams and delivery scams from PetScams.com and saved each URL, along with its type and the date it was identified, in a database table called scam_websites. During development I encountered an issue where PetScams.com would block my IP from making any more requests for several minutes, due to my script making dozens of HTTP requests every second. To overcome this, I modified the script so it would pause after each request for a number of seconds before proceeding, which significantly increased the run time of the program. To reduce this overhead, I ran experiments to find the smallest size pause that wouldn't result in my IP address being temporarily blocked. New scam websites were added to the list in chronological order. Hence, I made the decision to stop the script from running once it had found a URL that was already saved in the database. This prevented the scraper from scraping the full list every

[1] The first pet scam website was identified by PetScams.com in March 2017. However, it has been omitted from figure 3.1 because it was the only scam website identified that month.

time, and meant that subsequent runs of the script were much faster than the first run. In total, I saved the URLs of 11,518 scam websites which included 9,126 pet scam URLs and 2,392 delivery scam URLs.

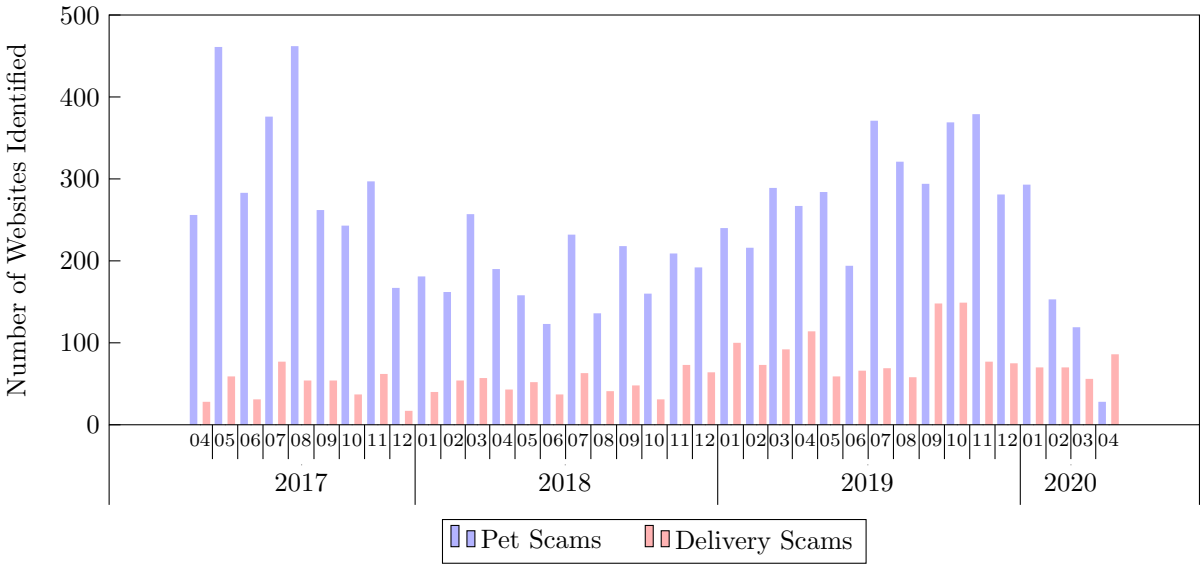


Figure 3.1: Number of pet scam and delivery scam websites identified each month from April 2017 to April 2020.

3.1.2 Downloading scam websites

I then created another scraper in Python that would visit each one of the pet scam and delivery scam websites that I had saved in the database, and download it as long as it was still online. I took an iterative approach to development by adding features one at a time. I started by creating a scraper that was able to visit all of the webpages belonging to a website. To do this, I parsed the HTML and found all of the links to another page, and added each link to a queue if it had not already been downloaded. The scraper then visited the webpage at the start of the queue. This process repeated until the queue was empty.

Once the scraper had a list of webpages, it was able to visit each one in turn and download the HTML and images on that page. Downloading the HTML was simple, but downloading the images required a more sophisticated approach. This was due to the variety of ways in which a webpage can display an image. In order to aid development, the scraper would save its progress to the scam_websites table. This progress information included the number of HTML pages downloaded, the number of images downloaded, and if an error had been thrown during the scraping. With this information, I was able to run an SQL query that would find all the websites that had either downloaded at least one HTML page, but zero images, or had thrown an error. I would then manually check the HTML of all these websites and update the scraper to support the scraping of images displayed in this format.

When designing the scraper, I considered the ethics of repeatedly sending HTTP requests to websites and decided that it would only request data at a reasonable rate. I added a pause of 100ms after each request so I wouldn't place a burden on the website. I also added spoof data to the request, to make it appear as if it were coming from a browser. Before this change, some websites were blocking the request.

Many of the sites my scraper crawled were online, but were no longer functional and had been replaced with a single webpage displaying an error. By recording the number of HTML pages and images downloaded, I was able to avoid using any websites that weren't operating in further analysis. I set a threshold of 4 for the number of HTML pages and images required for a website to be considered operational. In total, the scraper downloaded 1,380 websites including 1,064 pet scam websites and 316 delivery scam websites.

3.1.3 Pets sold on pet scam websites

The majority of pet scam websites will target a specific type and breed of animal. I looked at around 1,000 of the scraped websites and identified the types of pets that they were selling. Figure 3.2 shows that the most popular pet is dogs. Only a fraction of websites sell multiple types of pets. PetScams.com have done their own research on the most popular dog breeds and found that French Bulldogs and Yorkshire Terriers are the most popular. Figure 3.3 shows the most popular dog breeds.

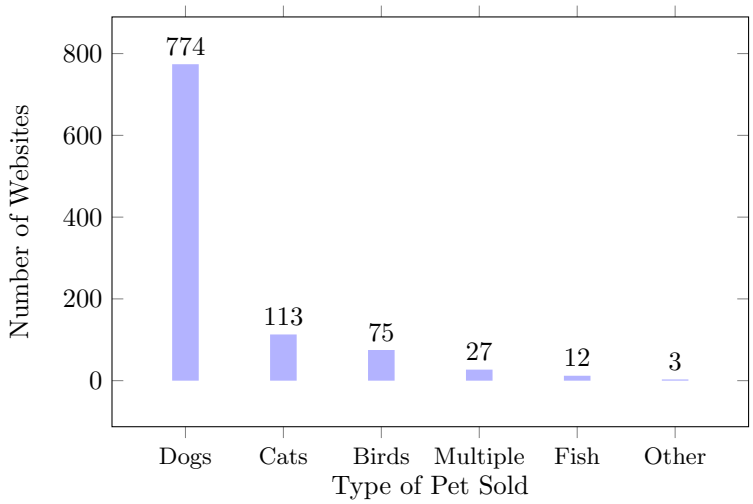


Figure 3.2: Number of scraped websites selling different types of pets.

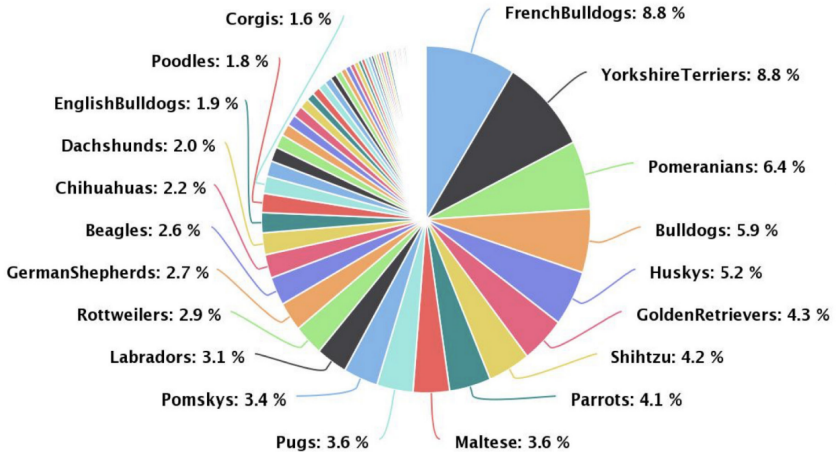


Figure 3.3: Popularity of dog breeds on pet scam websites. Taken from PetScams.com.

3.2 Finding Direct Links in HTML

The first type of connection I looked for was direct links in HTML. I made the assumption that if one scam website had a direct link to another scam website, then both websites were likely to have been created by the same person(s). To find these connections, I created a Python script to search the HTML of each website for any occurrence of a URL from the list of scam websites saved in the scam_websites table, whilst ignoring the URL of the website itself. When a URL was found, I stored the pair of websites in a new table called direct_links. This script was run whenever new URLs had been scraped from PetScams.com, or when new websites had been saved by the website scraper.

After analysing the HTML of 1,380 websites, I found 140 pairs of websites connected with a direct link in HTML. Some of these were linking to images that were hosted on another pet scam website's domain. By plotting all the websites in the direct_links tables onto a graph, and drawing edges between the nodes with a direct link in HTML, we end up with the resource network shown in figure 3.4. Almost 20% (237/1,380) of all scraped scam websites appear in this resource network. The vast majority of edges are only between websites of the same type. There is only one edge between a pet scam website and a delivery scam website. This link is because an image of a dog on a pet scam website is hosted on a delivery scam website.

Figure 3.5 shows the sizes of clusters in this resource network. It shows how the majority of clusters have two websites in them. The average degree (average number of edges each node has) is 1.18, which shows how few websites had more than one other pet scam URL in them. The largest cluster only has nine websites, which contains around 4% of all the websites in the resource graph.

Converting HTML into a string and performing a string search to find other pet scam URLs was a relatively quick task. It took approximately 75 minutes to complete with 1,380 websites scraped and 11,518 URLs. However, as the number of websites scraped and URLs increases in the future, so too will the run time. One optimisation I applied was multi-threading. I split the websites into a number of lists so that each thread would have a list of websites to search, and then the threads would run in parallel. This resulted in a large decrease in run time proportional to the number of threads running.

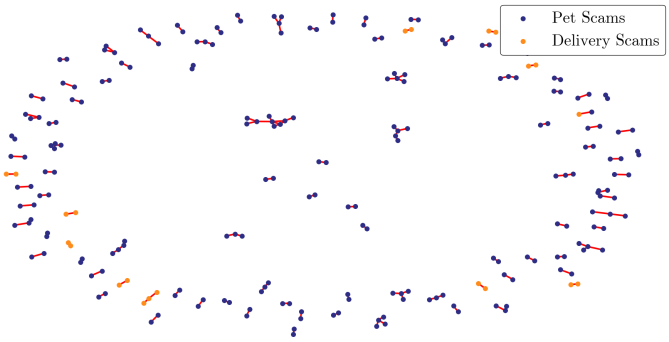


Figure 3.4: Resource network showing websites with a direct link in HTML.

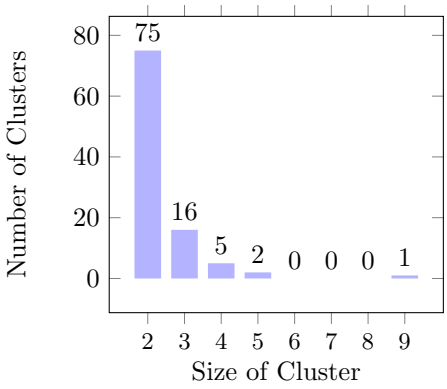


Figure 3.5: Size of clusters in the resource network generated by finding direct links in HTML.

3.3 Comparing Images

The second type of connection I looked for was shared images. My scraper had downloaded over 59,000 images so I needed to find a fast and robust method of comparing these images. I chose to use perceptual hashing and found an easy-to-use Python library for computing the perceptual hash of an image. I then wrote a Python script to compute the perceptual hash of every image and stored it in a table. This allowed me to write an SQL query that would group images based on their hash.

The library I used for computing the perceptual hash took the length of the desired hash as an input. I experimented with different lengths of hash as smaller hashes were faster to compute and resulted in more matches, but were less accurate. If the hash length was too small, then two images that are clearly of different objects can have the same hash. I found 64 bits of hash to be the optimal length. The script that computed and saved perceptual hashes would compute hashes of multiple lengths and save them all so I could later modify the length used if needed.

I also considered an alternative approach where the Hamming distance between all pairs of hashes would be compared. A threshold would be set for the distance, and if the distance was less than the threshold, I would count those two images as being the same. This would result in more image similarities found, since the current approach essentially uses a threshold of one. I chose not to take this approach as there are over 1.7 billion pairs of image hashes. Even though computing the hamming distance between two hashes is fast, it would still take too much time and require too much storage space to save the results.

Running an SQL query to return all hashes that were equal to at least one other hash took less than a second and out of 59,000 image hashes, there were 7,300 hashes that were shared by at least two images. These 7,300 hashes represented a total of 20,000 images, however, not all of these images were useful for finding comparisons between websites. Some of the most common hashes were from images that would not be useful, such as completely black images, or images of logos of popular companies that deal with transactions such as PayPal and Western Union. I wanted to exclude these types of images from consideration since two websites with the same payment processor logo isn't a good indication of the websites being created by the same person(s).

In order to exclude unwanted images, I added a field to the image_hashes tables to mark images as valid or invalid. The meant that invalid images could be excluded from being considered as shared images. I created a Python script to help me check the validity of images with matching perceptual hashes. This script iterated over the most common image hashes and displayed several images with each hash. I would then decide if the images were valid or invalid. Invalid images, such as logos of payment processors and images that are all one colour, were marked as invalid and this was saved in the database alongside each hash. The validity checker script also marked all images with a height or width less than 64 pixels as invalid automatically. With the help of this script, I was able to determine the validity of the images with the most popular images hashes, and I excluded over 2,000 invalid images and 300 invalid hashes from further consideration, leaving us with 18,000 images and 7,000 hashes.

When generating the resource network, an SQL query is run to find the list of perceptual hashes that at least two images have. For every hash in this list, another SQL query is made to find the names of the websites which have images with that hash. A GROUP BY clause is used to get the distinct website names as some of the shared images appear on the same website. An edge is then created for each one of the pairs from the pair-wise combinations of all websites in that list.

Image Threshold	Number of Websites	Size of Largest Cluster
1	913	359
2	701	134
3	617	37
4	571	27
5	506	26

Table 3.1: The effect of applying a threshold to the number of shared images on the number of websites and size of the largest cluster in the resulting resource network.

I experimented with setting a threshold for the number of shared images required to form a link between two websites. Table 3.1 shows that as the threshold increases, the total number of websites and size of the largest cluster decreases. A higher number of shared images provides more evidence that the sites were create by the same person(s), but results in fewer connections. I chose a threshold of 2.

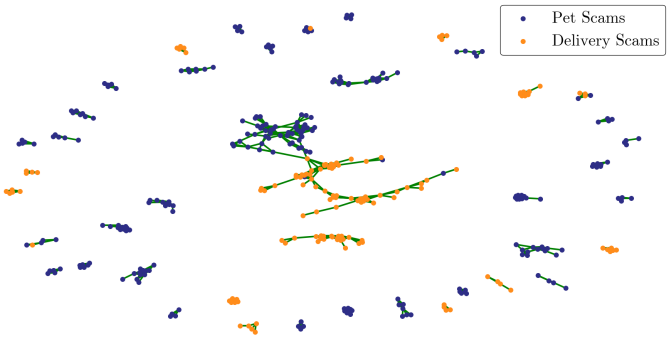


Figure 3.6: Resource network showing websites with at least two shared images images and a minimum cluster size of 4.

Figure 3.6 shows the resource network generated by connecting websites with at 2 shared images. Only clusters with 4 or more websites are shown. When exploring this graph, colour coding reveals that the majority of clusters only contained images from one type of website. However, the largest cluster which has 134 nodes is an outlier. It contains both pet scam and delivery scam websites. Figure 3.7 shows this cluster in more detail. There are two halves to the graph. One half contains pet scam websites, and the other half contains delivery scam websites. We can see that there is a single delivery scam website that joins the two halves as it shares images with both pet scam and delivery scam websites.

The smaller clusters in figure 3.6 are more likely to have nodes where all of the nodes are strongly connected. These complete sub graphs were a result of groups of websites that all shared the same 2 images. Viewing the websites names revealed that the majority of these websites had similar names and were selling the same breed of animal.

The largest complete sub graph had 8 nodes and the highest number of websites a shared image appeared on was 29. There is one pair of websites with more than 900 images in common. Both websites sell parrots and equipment needed to keep birds.

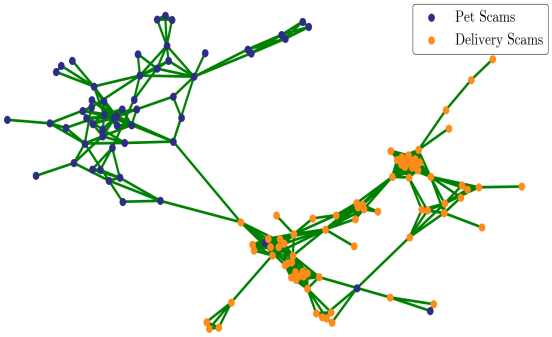


Figure 3.7: The largest cluster from figure 3.6 containing 134 websites.

Figure 3.8: One of the most popular shared images that appeared in 7 websites.

3.3.1 Combining Shared Images With Other Connections

We can generate resource networks that take into account two types of connections, which are direct links in HTML and shared images. By combining types of connections, we can see if there is any overlap.

Figure 3.9 shows the resource network generated by combining pairs of websites with either a direct link in HTML or at least two shared images. It appears to be very similar to the resource network generated by pairs of websites with at least two shared images, as this resource network contained 1,330 pairs, which is far more than the 140 pairs seen by just looking at direct links in HTML. The number of websites in the new resource network increased from 701 to 847 and the size of the largest cluster increased from 134 to 145.

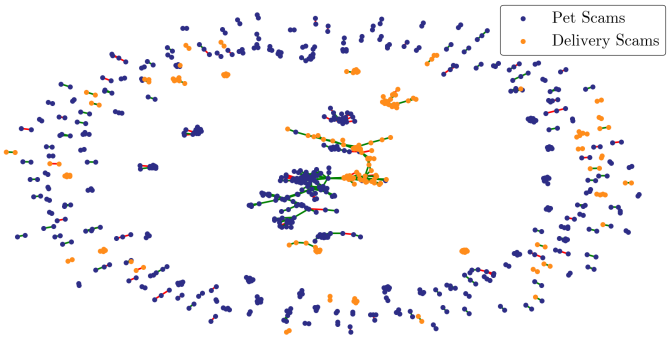


Figure 3.9: Resource network showing websites with a direct link in HTML or at least two shared images.

Figure 3.10 shows the resource network generated by looking at pairs of websites with both a direct link in HTML and at least two shared images. It looks similar to the resource network generated by just direct links in HTML (figure 3.4). 20% (28/140) of pairs of websites with direct links in their HTML also have at least two shared images, which supports the earlier claim that the reason for some direct links appearing in HTML is that they link to the same image.

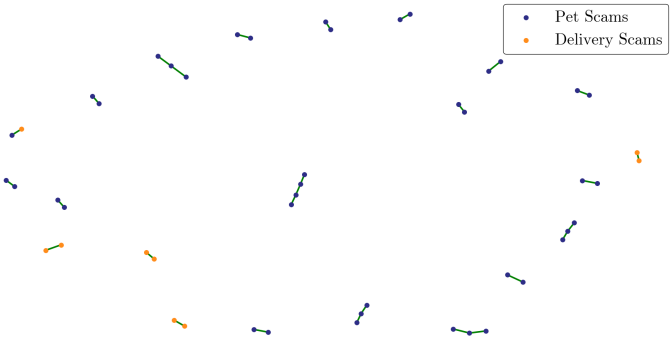


Figure 3.10: Resource network showing websites with a direct link in HTML and at least two shared images.

3.4 Comparing Text

There were a number of methods I could use to compare the text between two websites. This chapter looks at using the Jaccard index and the longest common substring. Both methods only look at the text contained within ⟨**P**⟩ tags. As there are almost one million pairs of websites to compare, I needed to find a method that was capable of finding results, but was also fast.

3.4.1 Jaccard Index

The first method I looked at was using the Jaccard index to quantify the similarity in the words and sentences between pairs of websites.

I first extracted all of the text in the ⟨**P**⟩ tags from each website, before using a tokenizer from NLTK to split the text into either a list of words or a list of sentences. I then computed the intersection divided by the union of both lists for every pair of websites to find the Jaccard index. I ran an initial experiment on 1,000 pairs before reviewing the results. Figure 3.11 shows that the distances measures for words were all very small numbers close to zero. This made it difficult to set a threshold. Websites selling the same breed of animal naturally had a higher Jaccard index. Computing the Jaccard index using sentences provided even worse results. The largest Jaccard index found after analysing 1,000 pairs was 0.38 for words and 0.15 for sentences. The Jaccard index will only be high if the majority of the text from two websites are similar. If a scammer copy and pastes one paragraph then the Jaccard index between the two sites will still be small.

In terms of performance, the Jaccard index for both words and sentences was relatively fast to compute. It was slightly quicker when looking at sentences. It took 75 minutes to compute the Jaccard index between 1,000 pairs of websites when using words, however it only took 60 minutes when using sentences since the number of sentences per website is much lower than the number of words.

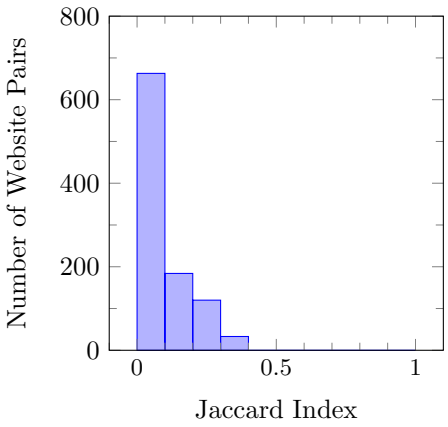


Figure 3.11: Distribution of Jaccard Index for words from 1,000 website pairs.

3.4.2 Longest Common Substring

The second method I looked at was finding the longest common substring (LCS). This method produced more interesting and varied results than the Jaccard index, but it was much slower to compute.

I started by timing a number of Python libraries that computed the LCS in order to find the fastest. The difference in run times was small, but since this function was going to be called hundreds of thousands of times, even a small percentage difference would add up to make a big impact. Table 3.2 shows the results of these experiments. The fastest function I found was the find_longest_match function from the difflib library.

The Python script I wrote iterates over all pairs of websites and creates two lists of strings from the text content within all the ⟨**P**⟩ tags from each website. The LCS between all pairs of strings is then found and the greatest LCS is saved in a database. The run time is dependant on the amount of text each website has. It takes an average of 10 seconds to load the HTML from two websites and find the LCS between their text. The average pet scam website I scraped had 29,438 characters in ⟨**P**⟩ tags, and the average pair of websites had 33 characters as the LCS.

Library	Time taken (s)
difflib	1031
pylcs	1081

Table 3.2: The time taken to compute the LCS between 100 pairs of websites.

LCS Threshold	Number of Websites	Size of Largest Cluster
300	1072	966
400	977	700
500	851	570
600	689	338
1000	336	46

Table 3.3: The effect of thresholding the length of the LCS on the number of websites and size of the largest cluster.

In order to reduce the run time of the algorithm, I excluded ⟨**P**⟩ tags with a length less than 200 characters of text. This reduced the size of the list and reduced the run time by approximately 20%, although it depended on how many small ⟨**P**⟩ tags the website had. A drawback of this optimisation is that if the LCS was less than 200, then this algorithm won't find it and will instead say it is 0. I decided that this was acceptable since I planned on setting a threshold much greater than 200. In the future, it would be possible to disable this optimisation and re-run it on previously saved pairs of websites with a LCS of 0 to see if it makes any difference.

Another optimisation I implemented was multi-threading. I used four threads which significantly improved performance. I also pre-fetched the ⟨**P**⟩ tags text and saved it in a dictionary, which increased start-up time, but decreased run time overall as it reduced the number of times data was loaded from HTML files from the number of pairs multiplied by two (1.9 million), to the number of websites (1,380). After all these optimisations are implemented, the script can compute the LCS between 1 pair every 2 seconds, or 40,000 pairs of websites a day, which means it took approximately 24 days to complete for all 950,000 pairs of websites. Without these optimisations, it would have taken hundreds of days.

I experimented with applying a threshold to the length of the longest common substring required to form a connection in the resource network. Table 3.3 shows that even with large thresholds, there were a large number of scam websites sharing text. When viewing the LCS in the database, I could see commonly used paragraphs that appeared in multiple websites. I decided upon a threshold of 400 characters.

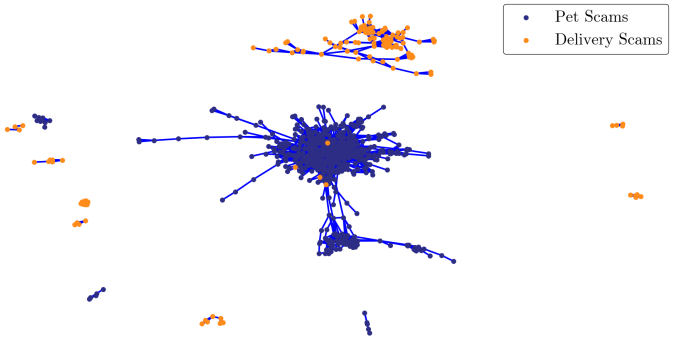


Figure 3.12: Resource graph of websites with a longest common substring of at least 400 characters and cluster size of at least 4 websites.

Figure 3.12 shows the resource network generating from looking at shared text between websites with a longest common substring greater than or equal to 400 characters. There were 977 websites in the graph with a large cluster of 700 websites. Over 70% of all scraped websites appeared in this graph which shows that the LCS is a good method for identifying similarities. There are two main clusters, one mostly consisting of pet scam websites and the other mostly consisting of delivery scam websites. The most common LCS was a paragraph of 586 characters (110 words) that was found in 59 websites and discusses the logistics of shipping a puppy by air. The substring was:

> Shipping a puppy by itself to a new location always sounds cruel and embarrassing, but actually I think it is harder for us than the puppy(s). With my many years of shipping experience, I know for a fact that all of the pups are well taken care of. So if you stop and

think about it, the airlines are not going to mistreat the puppy(s) for fear of lawsuit and customer dissatisfaction. I tape puppy(s) food and feeding instructions to the top of the crate and put frozen water in the crate, so it will gradually thaw out for the puppy(s) and the puppies are offered food along the ride.

3.4.3 Combining Textual Similarity With Other Connections

Figure 3.13 shows all 713 pairs of websites with a shared text of 400 characters and at least 2 shared images. This resource network no longer had single giant cluster; the largest cluster contained 25 websites. Approximately 12% (713/5925) of pairs of websites with a LCS of at least 400 characters also have at least two shared images. Pairs of websites that share two different types of resources are more likely to have been created by the same criminal(s), than pairs of websites with just one type of shared resource.

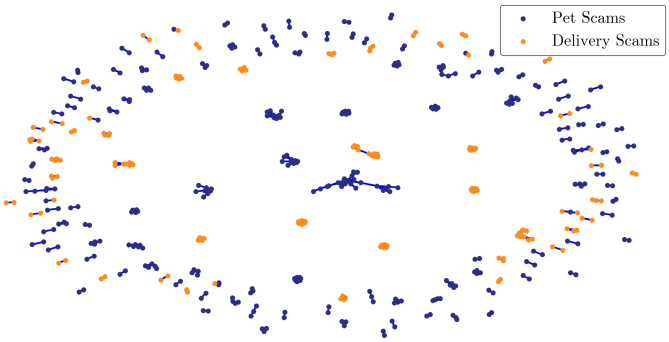


Figure 3.13: Resource graph of websites with shared of text of at least 400 characters and at least 2 shared images.

Figure 3.14 shows that 26 pairs of websites had a direct link in HTML and shared text of at least 400 characters. Around 19% (26/140) of pairs of websites with a direct link in HTML, also had a LCS of at least 400 characters.

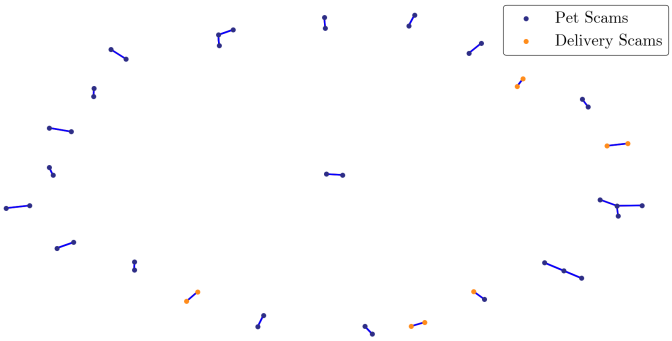
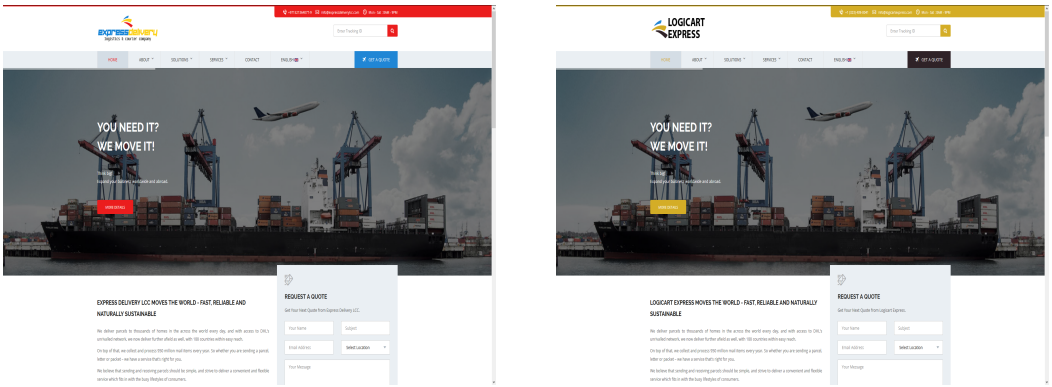


Figure 3.14: Resource graph of websites with shared of text of at least 400 characters and a direct link in HTML.

3.5 Comparing HTML Structure

I hypothesised that some pet scam websites would have a similar HTML structure due to being created with the same online tool. To determine the validity of this, I used the Jaccard index to compare the HTML structural similarity of pairs of websites by looking at the frequency of HTML tags. This method was very quick. I created a dictionary containing the number of times each tag occurred in each website, and then compared each dictionary to each other using the Jaccard index. I set a threshold of 0.9, which 369 pairs of websites met. The resource network shown in figure 3.16 contained 209 websites and the largest cluster contained 12 websites. There were 57 pairs of websites with a Jaccard index of 1.0. Figure 3.15 shows an example of two websites that have a HTML similarity of 1.0. Both of these websites were part of a cluster of 8 delivery scam websites, which suggests that all 8 websites were created by the same people.



(a) expressdeliveryllc.com (b) logicartexpress.com

Figure 3.15: An example of two websites with a HTML tag similarity of 1.0.

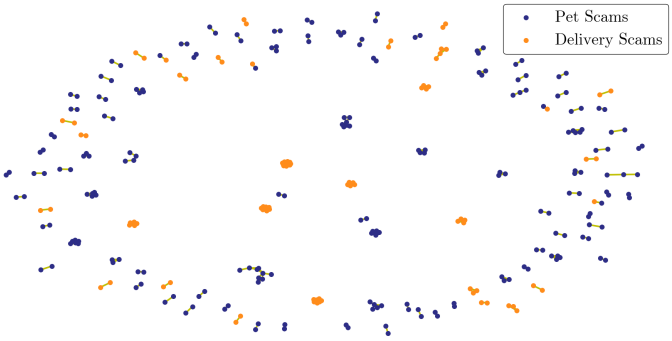


Figure 3.16: Resource network showing websites with a HTML tag similarity of at least 0.9.

Computing the similarity of HTML tags between two websites is much faster than the other methods of comparison we have looked at. It takes less than a second to find the Jaccard index between the HTML tags of 1,000 pairs of websites, provided the tags have been pre-fetched and saved in a dictionary beforehand. This means it took around 15 minutes to compare the HTML similarity of all 950,000 pairs of websites.

3.5.1 Combining HTML Similarity With Other Connections

We can combine HTML similarity with the other three connections we have looked at in this paper.

By combing HTML similarity and direct links in HTML into a single resource network, we can see that there is very little overlap. Figure 3.18 shows this resource network. There are only 7 edges in this resource network which means that 5% (7/140) of websites with a direct link in HTML have a similar HTML structure. This shows that websites with a direct link in HTML, are unlikely to also have similar HTML.

Figure 3.18 was generated by finding pairs with both a similar HTML structure and a minimum of 2 shared images. There are more edges and the clusters are larger in size, compared to the previous resource network.

Finally, we can combine a high HTML similarity with a LCS of at least 400 characters. This resulted in the largest resource network, containing 340 edges. By looking at figure 3.19, we can see that this resource network has the largest clusters out of the three. 92% of edges with a high HTML similarity, also had a LCS of at least 400 characters.

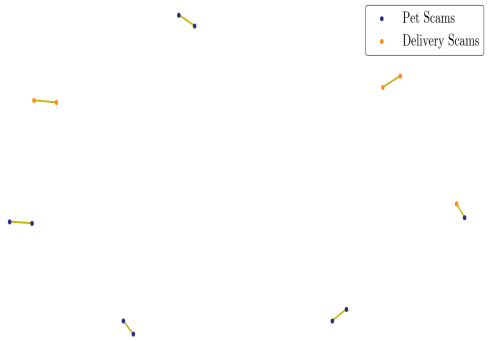


Figure 3.17: Resource network showing websites with a HTML tag similarity of at least 0.9 and a direct link in HTML.

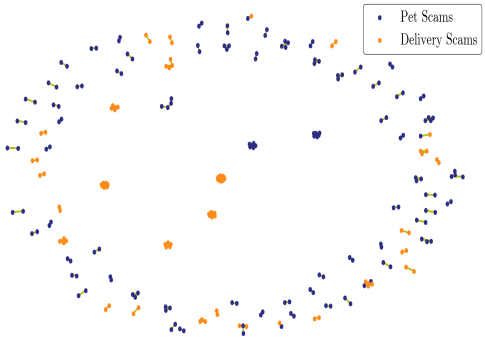


Figure 3.18: Resource network showing websites with a HTML tag similarity of at least 0.9 and at least 2 shared images.

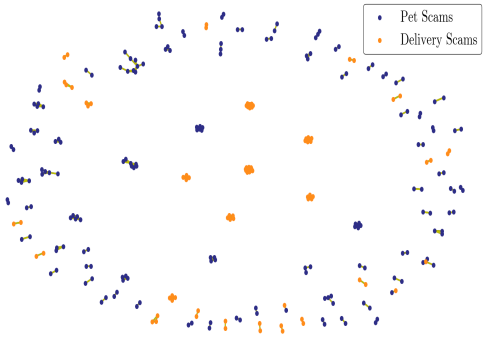


Figure 3.19: Resource network showing websites with a HTML tag similarity of at least 0.9 and a LCS of at least 400 characters.

3.6 Validating Clusters With WHOIS Data

So far, all of the methods for finding connections between websites have used on-site data. We will now look at using WHOIS data to provide further evidence that pairs of websites are connected.

3.6.1 Acquiring WHOIS data

I created a Python script to save the WHOIS data for all of the scraped websites to a database. This data included the IP address, registrar, date of registration and address. I then looked at all the pairs of websites in the resource network and compared their WHOIS data. I directly compared the name of the registrar, the address and the IP address and checked for exact matches. If there were any matches, then there is further evidence of these websites being created by the same group of people. Not every website I had scraped had complete WHOIS data. Some were missing values in fields such as the address, or the website had been taken down so the WHOIS data was no longer available. However, I was able to retrieve and save the WHOIS data for more than 1,000 websites. I saved the WHOIS data in a CSV file alongside where I had saved the HTML and images of websites. I also saved the WHOIS information that I retrieved to a database table for ease-of-access.

I noticed that there was little variety in registrars. There were only 60 different registrars, with the average registrar having 23 websites. Namecheap INC. was the most popular registrar with 500 different pet scam websites using them, and NameSilo LLC. was the second most popular registrar with 183 websites using them. Both of these companies offer services which hide the address of the person who registered the website. Many of the addresses given by the WHOIS data are therefore hidden.

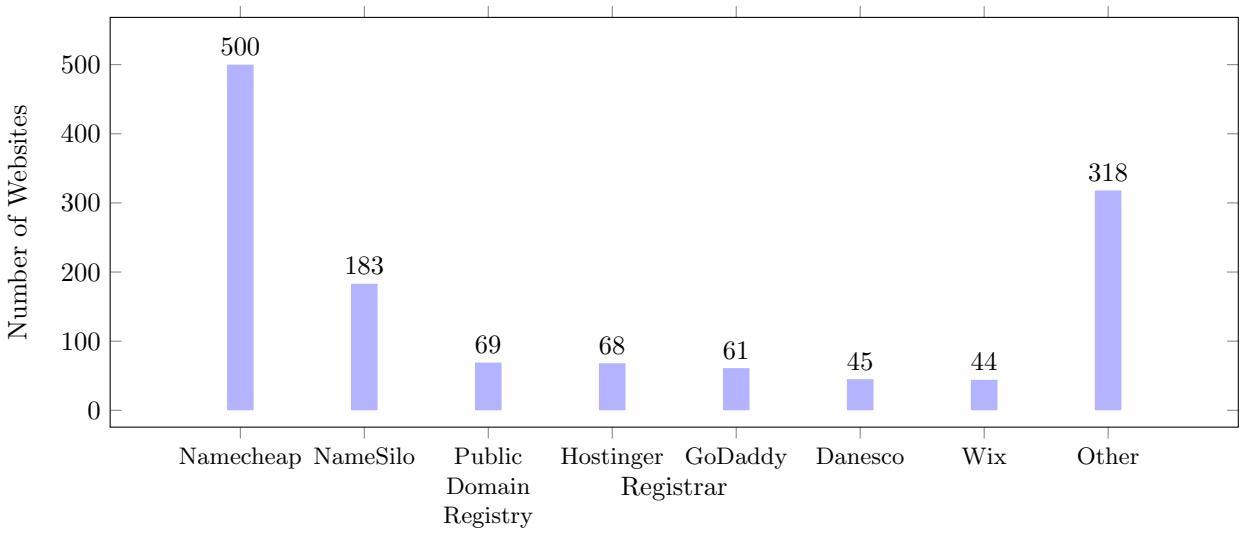


Figure 3.20: Number of websites registered with the most popular registrars.

Geolocation of IP addresses

I used a free API called `freegeoip.app` to find the geolocation of IP addresses belonging to pet scam websites and saved them in the scam_websites table. I then plotted them on a world map, seen in Figure 3.21. This map shows that the vast majority of IP addresses were based in the USA. Websites are able to spoof their IP address. Therefore, just because an IP address comes from one country, doesn't mean the creators are based in that country. Furthermore, many cloud hosting service providers are based in the USA, so although the IP address is based there, the creators may be based in a different country.

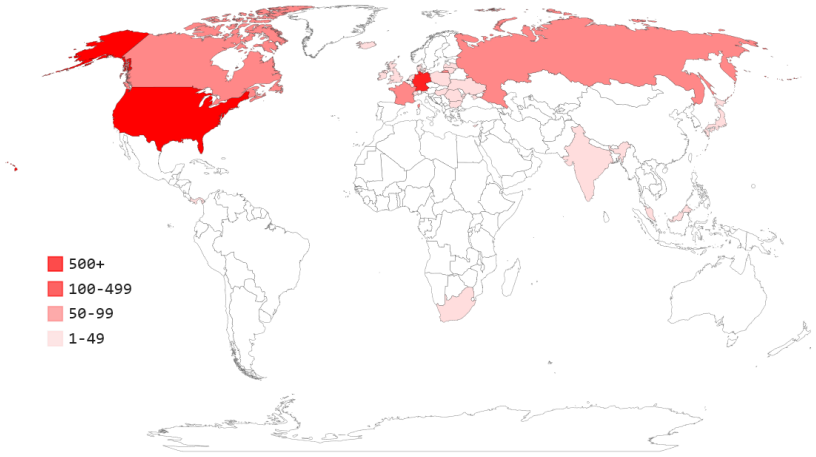


Figure 3.21: Number of pet scam IP addresses in each country.

3.6.2 Using WHOIS to validate clusters

Now we have acquired WHOIS data, we can validate the clusters generated by the different resource networks. We will look at all pairs of websites within a network, and compare the two pieces of WHOIS data.

I went back to the resource networks generated by comparing just one type of resource and compared the WHOIS data between all pairs. I found that a high HTML similarity was the best indicator of two websites having a match in their WHOIS data. For instance, out of the 304 pairs of websites with a HTML tag similarity greater than 0.9 and WHOIS data, 73% had an exact match.

From the two resource networks generating by looking at direct links in HTML and shared images, the pairs of websites with matches in their WHOIS data was 41% and 48% respectively. Although this match rate is lower than the one given by setting a threshold on the HTML similarity, it shows that almost half of these pairs have an additional piece of off-site evidence showing that they may have been created by the same people. Shared text has the lowest validation rate of any single connection type, with only 34% of pairs of websites with WHOIS data having a match. However, it is also the single connection type with the greatest number of matches, with 1,674.

Connections used in resource network	**Number of pairs of websites with WHOIS data**	**Number of pairs of websites with matches in WHOIS data**	**Percentage of pairs of websites with matches in WHOIS data**
Direct links only	34	14	41%
Shared images only	1080	517	48%
Shared text only	4908	1674	34%
Similar HTML only	304	223	73%
At least one connection	5772	1860	32%
At least two connections	677	400	59%
At least three connections	242	167	69%

Table 3.4: The number of pairs of websites with matching WHOIS data, given the type of connection that was used to generate the resource network.

We can set a threshold for the minimum number of connections required to form an edge in a resource network. Table 3.4 shows that setting the threshold to one results in the lowest validation rate of 32%. If we increase this threshold, we increase the validation rate, but reduce the number of edges in the resource network.

Scammers might not always use the same registrar when creating a website. In section 3.4.2 we found the longest common substring between pairs of websites and discovered 59 puppy-selling websites with the same 586 character substring. When looking at the WHOIS data for 51 of these websites, we can see a range of 9 different registrars. 30 of them use Namecheap, 7 use Hostinger and 6 use NameSilo. The rest are only used by 1 or 2 websites.

3.7 Development of an Interactive Graph and Website

In the introduction, we discussed the lack of research into identifying connections between pet scam websites. By creating a website, we can share our results and give users the opportunity to explore the resource networks of pet scam websites. This tool could be used by law enforcement to see clusters of websites that have multiple connections and are therefore likely to have been created by the same group of criminals. This will help the law enforcement to focus their resources and have a greater impact in preventing pet fraud.

So far, we have generated static graphs that display scam websites as nodes and the connections between them as edges. The node and edge colour can only convey a limited amount of information about the node and edge. In this case, the type of scam website and connection. The library that we use to generate them can show the names of all the nodes, but as there are hundreds of nodes in close proximity to each other, it becomes unreadable. This problem can be overcome by only displaying the name of the node when clicking on it, however this requires using a different Python library called Dash that allows the user to interact with the graph. We will use Dash to create an interactive graph that allows uses to click on nodes and edges to learn more about the connections we have identified between pet scam websites. We will host this interactive graph on a website so it can be easily shared and seen by as many people as possible.

When exploring options for hosting the website, I decided to use Heroku as it provides us with free HTTPS and a database with up to ten million rows. I also used a free service for hosting the images called Cloudinary which supports up to 300,000 images which was sufficient for our needs.

The interactive graph needed to display information related to the feature-based connections we have discovered between websites. Therefore, when an edge is clicked, it will display if there is a direct link in HTML between the two websites, if they have a high HTML similarity or if they share any WHOIS data. If there is any shared text, then it will be displayed. Images shared between websites also need to be displayed side by side which will allow users to visually compare them themselves. Edges are still coloured based on the type of connection, and nodes are still coloured based on whether they are a pet scam or delivery scam website. Hovering over a node will display the name of the website and the number of other scam websites it is connected to. Clicking on a node will display these scam websites as a list.

Information about the connections can be stored in the Heroku database, such as the longest common substring between websites. Whenever the user clicks on a node or edge, an SQL query is made to fetch the relevant information to be displayed, such as the longest common substring, or the URLs of the shared images which have been uploaded to the Cloudinary image hosting service. In total, I uploaded 13,828 images.

The biggest difficulty I encountered in development of the interactive graph and website was the Heroku 30 second boot-up limitation. If the application could not boot-up within 30 seconds then it would fail. There were initially two tasks to be completed at boot-up. The first was to extract the pairs of websites from the various tables that met the thresholds for each feature. The second task was to generate the graph which took time as it needed to compute the position for each one of the thousands of nodes and edges. The first task was the longest and took a couple of minutes, but even the second task took slightly longer than 30 seconds. In order to reduce the number of computations being performed by the Heroku machine, I created a Python script to generate the graph offline and then save the coordinates of all the nodes and edges in the Heroku database. This meant that there was only one job to do during boot-up, which was calling SQL queries to get the pre-computed positions of the nodes and edges. This resulted in a large speedup in runtime, and the boot-up now only take 3 seconds. The Python script to generate the graph can be called whenever new analysis of the scam websites has been done, and keeps the online graph up to date with the offline graph.

Figure 3.22 shows the full system diagram. The system can be run at a set time interval, such as once a week. It will scrape all newly identified pet scam websites, find comparisons between them and then update the resource network on the website.

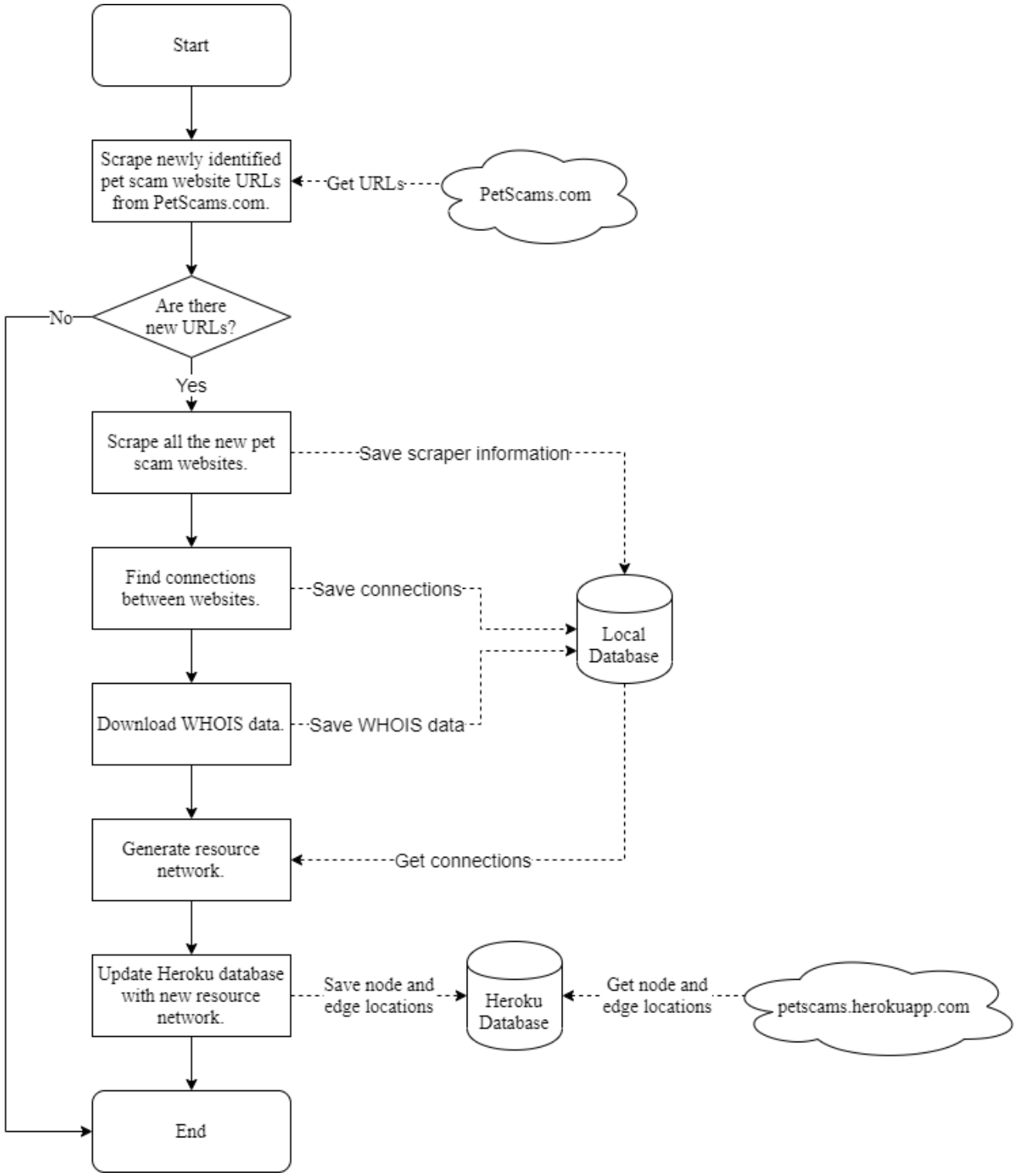


Figure 3.22: System diagram.

Chapter 4

Critical Evaluation

4.1 Cluster Evaluation

4.1.1 Combining All Four Types Of Connections

We have now looked at finding four different types of connections between pet scam websites. We can combine them all together to generate resource networks that takes into account all four features. We can then set a threshold on the number of connections required to appear in the resource network.

At Least One Connection

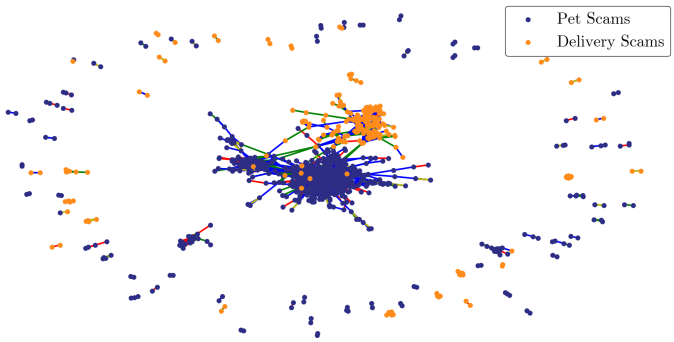


Figure 4.1: Resource network showing websites with at least one of the following: A direct link in HTML, at least two shared images, a LCS of at least 400 characters, or a HTML tag similarity of at least 0.9.

When looking at the resource network of websites with a direct link in HTML, at least two shared images, a LCS of at least 400 characters, or a HTML tag similarity of at least 0.9, we end up with the resource network shown in figure 4.1. It contains 1,212 nodes and the largest cluster consists of 967 websites. Almost 90% of all scraped websites appear in this resource network and the majority of edges are caused by shared text.

However, two websites having one type of connection may not be enough evidence that the websites are connected. Instead, we can require multiple thresholds to be met before drawing a connection between two websites in the resource network.

At Least Two Connections

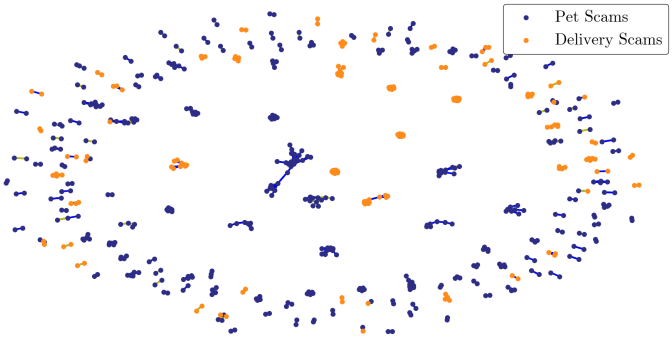


Figure 4.2: Resource network showing websites with at least two of the following: A direct link in HTML, at least two shared images, a LCS of at least 400 characters, or a HTML tag similarity of at least 0.9.

Figure 4.2 shows the resource network when we set a threshold for 2 connections. It contains 595 websites and the largest cluster has 27 websites. Although this resource network contains half the number of nodes as the previous network, the largest cluster is approximately 35 times smaller. These websites have a higher likelihood of being created by the same person(s) due to the range in the types of resources they share.

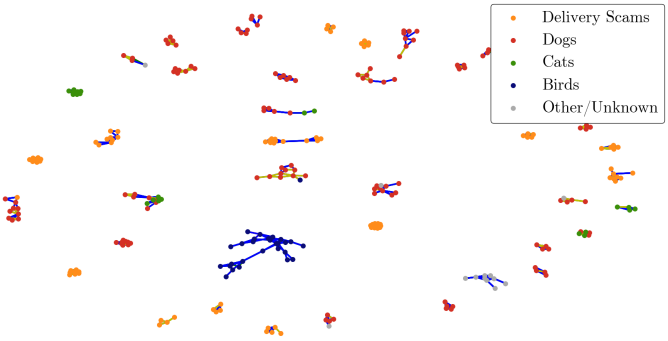


Figure 4.3: Resource network showing websites with at least two of the following: A direct link in HTML, at least two shared images, a LCS of at least 400 characters, or a HTML tag similarity of at least 0.9. Pet scam websites are colour coded based on the type of pet they sell, and only clusters with at least 4 websites are shown.

All of the resource networks we have generated are able to find lots of connections between websites of the same type, but struggle to find connections between websites of different types. Figure 4.3 colour codes the pet scam websites based on the type of pet they sell. We can see that clusters mostly sell the same type of animal, and that all of the websites in the largest cluster of 27 websites sell birds.

At Least Three Connections

If we set a threshold of 3 types of connections being required to form an edge, then we create the resource network shown in figure 4.4 which contains 246 websites. Although the majority of clusters contain 2 or 3 websites, there are 7 fully connected clusters with at least 5 websites. The largest cluster consists of 12 delivery scam websites. 11 out of the 12 websites have WHOIS data available, and 10 of them were registered with Namecheap, whilst 9 have the same IP address. This suggests that this cluster of 12 websites were all created by the same criminal(s).

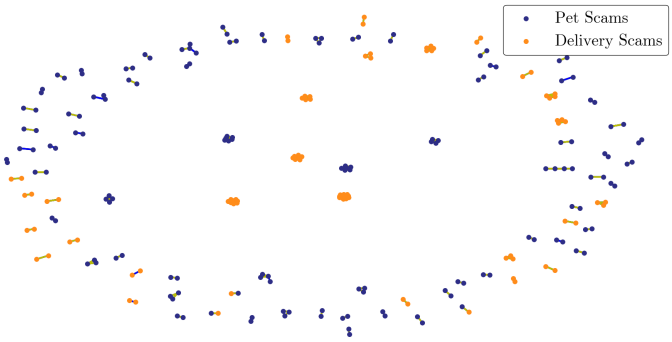


Figure 4.4: Resource network showing websites with at least three of the following: A direct link in HTML, at least two shared images, a LCS of at least 400 characters, or a HTML tag similarity of at least 0.9.

4.1.2 Connections That Support Each Other

	Percentage of edges that have another connection			
Type of Connection	**Direct Link**	**Shared Images**	**Shared Text**	**Similar HTML**
Direct Link	-	20.0%	18.6%	5.0%
Shared Images	2.1%	-	53.6%	22.1%
Shared Text	0.4%	12.0%	-	5.7%
Similar HTML	1.9%	79.7%	92.1%	-

Table 4.1: Percentage of edges with one type of connection that have an additional type of connection.

Throughout this paper, we have looked at generating resource networks that combine pairs of features. Table 4.1 shows the percentage of edges which have one type of connection that also have an additional type of connection. It shows us that websites with a high HTML similarity have a 92.1% chance of also having a LCS of least 400 characters, and a 79.7% chance of having at least two shared images. Since pairs of websites with a high number of shared resources are more likely to have been created by the same person(s), we can see that just looking at HTML similarity as the only connection would provide an accurate resource network. Further evidence can be seen in table 3.4, which shows that the resource network generated with only similar HTML connections have 73% of their edges validated with WHOIS data.

Shared text edges are unlikely to have any other types of connections. The most likely connection is shared images with a 12.0% chance. This suggests that the LCS threshold is too small, or perhaps there are paragraphs of text that appear in multiple websites, but don't indicate that they were made by the same person. Validating with WHOIS also reveals that shared text is the weakest connection, as only 34% of pairs of websites with shared text have matches in their WHOIS data. This reveals that there are lots of false positives. However, shared text is the most numerous type of connection, with over 5,000 edges.

There is a positive correlation between connection types that share edges with other connection types, and the WHOIS validation rate of connections types. Connections types that are more likely to share a second type of connection, were more likely to have a higher WHOIS validation rate. The WHOIS validation rates for the shared text, direct links, shared images, and similar HTML were 34%, 41%, 48% and 73% respectively. The maximum percentage of edges with a second connection type were 12%, 20%, 53% and 92% respectively.

4.1.3 Dunn Index

Connection Threshold	Minimum Separation	Maximum Diameter	Dunn Index
At least one connection	0.75	0.89	0.84
At least two connections	0.75	0.81	0.93
At least three connections	0.50	0.38	1.32

Table 4.2: The Dunn Index for resource networks with different connection thresholds.

We can compute the Dunn index for a resource network by giving each connection type a weight of 0.25. A distance measure can be achieved by subtracting the total weight from 1, as this gives us a value between 0 and 1. There is a choice to be made for the distance measures used when computing the inter-cluster and intra-cluster distances. We chose to use the average linkage distance, as the single or complete linkage distance values would always be one of 0, 0.25, 0.5, 0.75 or 1.

Table 4.2 shows the Dunn index for the three resource networks that have a connection threshold of 1, 2 and 3. The table shows that as the connection threshold increases, so does the Dunn index. We aim to maximise the minimum separation between clusters and to minimise the maximum diameter within a cluster. The maximum diameter decreases as we increase the threshold which shows that clusters are getting more compact. Despite the Dunn index being a computationally expensive calculation, it only takes a few minutes to calculate for the largest resource network as it only has around 1,900 edges.

4.1.4 Website Registration Dates

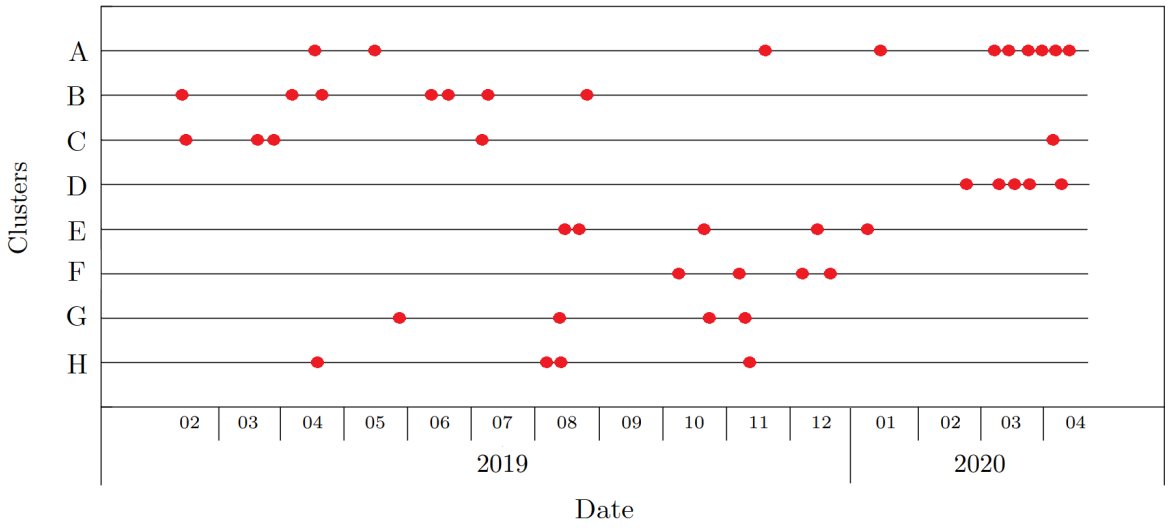


Figure 4.5: The date when websites in the largest 8 clusters were registered according to WHOIS data. Clusters are from the resource network generated from websites with similar WHOIS data and at least three of the following: A direct link in HTML, at least two shared images, a LCS of at least 400 characters, or a HTML tag similarity of at least 0.9.

We can look at the registration date of websites in the WHOIS data to see if there is any pattern. Figure 4.5 shows the dates when websites in the 8 largest clusters were registered. These clusters come from the resource network generated by pairs of websites with at least three different types of connections and similar WHOIS data. Most of the websites in a cluster were registered over a long time frame; only a few clusters had websites registered at a similar time. Cluster D had all 5 of its websites registered in a 6 week period in early 2020, and cluster A had 6 of its 10 websites registered within a five week period. Since these clusters share at least three types of resources and have matches in their WHOIS data, we expect them to have been created by the same person(s). Websites being created and registered around a similar time supports this.

4.2 Scraper Evaluation

The scraper we have created depends heavily on PetScams.com. It is the only source of pet scam URLs we have used. During development of this scraper, they renamed the path of the URL that leads to the list of pet scam websites. This temporarily broke the scraper and required me to update the URL. Although this problem was simple to fix, it highlights how vulnerable the system is to changes outside of my control.

The scraper makes up just one component of the system. There are a number of Python scripts that each perform a specific role, such as finding shared text between websites. The scripts are modular so they can either be run once individually, or run altogether where we complete the entire process from start to finish. The system can be split into three stages. The first stage scrapes newly identified pet scam websites, the second stage compares the pet scam websites, and the third stage generates a new resource network for the website. A cron job can be set up to run the system once a week which makes the entire process automatic. The time it takes for the first two stages to complete depends on the number of new websites identified. If no new websites are identified, then the system ends prematurely. If new websites have been identified, then the system will only compute comparisons between pairs of websites that haven't yet been computed, which prevents us from redundantly re-computing unnecessary comparisons. The runtime of the third stage is not affected by the number of newly identified sites, unless no new websites were identified, in which case this stage won't run at all. On average, if there have been 10 new sites identified, then it takes 20 minutes for them to be scraped, 400 minutes for new connections to be found, and 30 minutes for the website to be updated with the new resource network. The total runtime is 7.5 hours, however, the majority of the time is spent working out the LCS between pairs of websites. For every new website scraped, the number of LCSs to compute is equal to the number of scraped websites so far. Removing the LCS calculations reduces the total runtime to just 1.5 hours.

4.2.1 Testing The Scraper

When testing the scraper, I wanted to ensure that it was downloading all the webpages and images available. If it had missed any then there would have been less data to analyse. By saving the number of HTML pages and images the scraper had downloaded into a database, I was able to visit a sample of websites and manually count the number of HTML pages and images. If the scraper had missed any, then I checked the HTML of the website and updated the scraper.

4.3 Website Evaluation

Heroku was free for me to use because I was given credits for being a student. When these credits run out, it will cost around £12 a month. As more scam websites are identified and added to the resource network, there will be more rows in the database, and more images uploaded to Cloudinary. If we extrapolate from the number of the scam websites we have downloaded so far, then it would take years to approach the threshold where the cost per month increases.

Generating the resource network for the website offline takes a couple of minutes, but uploading the positions of the nodes and edges to the Heroku database takes up to 30 minutes. This is because there are thousands of nodes and edges, and it is slow to make INSERT SQL queries to the Heroku database.

The website shows 1,296 nodes which represents 94% of all scraped pet scam websites. Edges belonging to all connection types are shown and there are a total of 9,083 edges. The large number of nodes and edges can make the graph look messy, but the interactive graph allows users to zoom in which helps them to look at clusters in more detail.

4.4 Limitations

There are a number of limitations to this research. The first is the fact that all of the pet scam websites came from the same source, PetScams.com. All of these websites have been identified by the same people, and therefore may possess characteristics not seen in other pet scam websites. To resolve this limitation, the scraper could scrape pet scam websites from another source such as the IPATA list mentioned earlier in this paper.

Another limitation is the type of data we have looked at. We have looked at textual, image and HTML structural similarity, but there are other things to look at, such as headers in the HTTP response. We

also rely on WHOIS data for validating the clusterings and this data isn't always available which limits the number of connections we can validate. In addition, we would have been able to better evaluate how accurate the clusterings were, if we had a list of pet scam websites that were confirmed to have been created by the same people.

Finally, there is an issue with the way our system counts shared images. Currently, it computes the perceptual hash of all scraped images, and if there is a match in the hashes, then this counts as a shared image. We made efforts to exclude invalid images such as those of completely black squares, or logos of payment processors such as Western Union or PayPal. To exclude these invalid images, we manually looked at the images with the most popular perceptual hashes and then marked them as valid or invalid. Invalid images were not counted as shared images when the resource networks were being generated. However, we did not manually check all of the images, as there are over 59,000 which means that some invalid images were being included in the resource network. Raising the threshold for a shared image connection to 2 helped to mitigate this, but creating a system to automatically exclude these invalid images would be an even better solution. We could create a database of logos that we want to exclude and then use image recognition algorithms on all the scraped images to see if there is a match. Another solution would be to store the perceptual hashes of these logos and then automatically exclude scraped images that have one of these hashes. Both of these systems could be implemented to reduce the number of invalid images being counted as shared images, which would improve the reliability of our resource networks.

Chapter 5

Conclusion

In this paper, we have looked at developing a website scraper that successfully scraped over 1,300 pet scam websites, and then compared these websites by finding four different types of connections. We have looked at finding direct links in the HTML of one website to another, finding images that are present on multiple websites, finding paragraphs of text that appear on multiple websites, and measuring the HTML structural similarity between pairs of websites. We generated resource networks by setting a threshold on these connections. We then saved the WHOIS data for as many pet scam websites as possible and used this data to validate the connections. Finally, we looked at creating a website to display all of the connections we found between pet scam websites, and automated the entire system so that new pet scam websites can be scraped, compared and added to the website as soon as they have been identified by PetScams.com.

Our research hypothesis was that it is possible to cluster pet scam websites based on their shared resources. Throughout this paper, we have discovered that it is possible to find shared resources between websites, and that many websites share multiple types of resources. The resource networks we generated include a large proportion of the scraped pet scam websites and show multiple large clusters. Validating with off-site data provided further evidence that these websites are linked, and suggests that there are criminals responsible for a large number of pet scam websites.

5.1 Outcomes

The scraper that we created is able to scrape the URLs of newly identified pet scam websites from PetScams.com, and then download the HTML and images from these websites. It crawled all 11,518 of the websites identified by PetScams.com and downloaded 1,380 of them that were still online.

Out of the four connection types we looked at, we discovered that comparing the similarity of HTML structure by using HTML tag frequency analysis had the highest rate of validation. At 73%, this was slightly higher than the 69% rate achieved by requiring at least three connection types. The connection type with the lowest validation rate was the longest common substring, with only 34% of these edges being validated by WHOIS data.

By setting a threshold on the number of connection types required to draw an edge, we generated resource networks that show pairs of websites that share multiple resources. When setting a connection threshold of 3, we generated a resource network with 295 pairs of websites, and the largest cluster contained 12 websites. Furthermore, 69% of the edges with WHOIS data have similar WHOIS data which suggests that they were made by the same people. Clusters contained websites that have the same type, either pet scams or delivery scams, and if they sold pets then the pets were all the same animal.

The website I created (petscams.herokuapp.com) hosts an interactive graph that shows all 1,296 pet scam websites that have a connection with another pet scam website. There are a total of 9,083 edges in this resource network. The websites are colour coded based on whether they are a pet scam or a delivery scam website, and clicking on a website or an edge between websites reveals more information about it. For example, if two websites share 10 images and have a LCS of 586 characters, then clicking on this edge will show 20 images (10 from each website side by side) and the LCS to the user.

5.2 Future Work

There are several areas that future work could look at.

Having a collection of websites that we know for certain were created by the same people will help us to evaluate our resource networks, and determine if they are good at finding links. Currently, we have relied on using WHOIS data, which isn't always available, to validate connections.

We could improve the scraper by including pet scam websites that were identified from other sources such as IPATA. This would increase the number of pet scam websites we can scrape. When scraping websites we could also use Selenium as this would allow use to scrape JavaScript generated content.

When comparing images, we noticed that some images appear in multiple websites, but are not good indications of them being made by the same person(s). These invalid images were often logos of payment processors such as Western Union. By creating a database of these images, we could use image recognition algorithms or perceptual hashing to detect if any of the images that we had scraped matched these invalid images. This would automatically exclude them from being considered as shared images, and would improve the reliability of our shared image connections.

When validating with WHOIS data, the longest common substring approach to textual similarity was the connection type with the lowest proportion of validated edges. As this was the only connection type that looked at the textual content of websites, future work could look at other methods that focus on text. There exist multiple algorithms that can be used to compare the stylometry of two pieces of text. We could use these algorithms as an additional way of measuring the textual similarity of two websites. This method would help us to find connections between websites that don't have the exact same substrings, but do have a similar writing style.

Finally, the system we have created that scrapes websites and finds connections between them is not specific to pet scam websites. The system could use URLs from another source and generate resource networks for any other kind of scam website. It has applications outside of pet fraud and future researchers could use it to help prevent more people from becoming victims to scams.

Bibliography

[1] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, USA, 2007. USENIX Association.

[2] R. Biswas, E. Fidalgo, and E. Alegre. Recognition of service domains on tor dark net using perceptual hashing and image classification techniques. In *8th International Conference on Imaging for Crime Detection and Prevention (ICDP 2017)*, pages 7–12, 2017.

[3] Better Business Bureau. Puppy scams: How fake online pet sellers steal from unsuspecting pet buyers: A BBB study, 2017.

[4] Jake Drew and Tyler Moore. Automatic identification of replicated criminal websites using combined clustering. In *Proceedings of the 2014 IEEE Security and Privacy Workshops*, SPW '14, page 116–123, USA, 2014. IEEE Computer Society.

[5] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[6] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 01 1974.

[7] Benjamin Edelman. Largescale intentional invalid whois data: A case study of 'NicGod productions' / 'domains for sale'. cyber.harvard.edu/archived_content/people/edelman/pubs/Edelman-Judiciary-052002.pdf, 2002. Accessed: 08-05-2020.

[8] Matthew Edwards, Guillermo Suarez-Tangil, Claudia Peersman, Gianluca Stringhini, Awais Rashid, and Monica Whitty. The geography of online dating fraud. 5 2018. Workshop on Technology and Consumer Protection : Co-located with the 39th IEEE Symposium on Security and Privacy, ConPro ; Conference date: 24-05-2018 Through 24-05-2018.

[9] Kathryn Elliott. The who, what, where, when, and why of whois: Privacy and accuracy concerns of the whois database. *SMU Sci. & Tech. L. Rev.*, 12:141, 2008.

[10] Timothy C. Hoad and Justin Zobel. Methods for identifying versioned and plagiarized documents. *J. Am. Soc. Inf. Sci. Technol.*, 54(3):203–215, 02 2003.

[11] Csaba Legány, Sándor Juhász, and Attila Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, AIKED'06, page 388–393, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS).

[12] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., USA, 1st edition, 1996.

[13] Najmeh Miramirkhani, Oleksii Starov, and Nick Nikiforakis. Dial one for scam: A large-scale analysis of technical support scams. In *NDSS*, 2017.

[14] CBS News. Delta says bogus website tricks people who put pets on jets. cbsnews.com/news/delta-says-mysterious-bogus-website-tricks-people-who-put-pets-on-jets/, 2017. Accessed: 10-05-2020.

[15] Nor Sa'datul Aqma Norazman and Norshuhani Zamin. Development of scammed posts detector: A case study of pet scammed posting.

[16] Neil Chou Robert Ledesma Yuka Teraguchi and John C Mitchell. Client-side defense against web-based identity theft. In *Proceedings of the Network and Distributed System Security Symposium, NDSS, 2004*, 2004.

[17] Martin Theobald, Jonathan Siddharth, and Andreas Paepcke. Spotsigs: Robust and efficient near duplicate detection in large web collections. In *31st annual international ACM SIGIR conference on Research and development in information retrieval 2008 (SIGIR 2008)*, 2008.

[18] Tanguy Urvoy, Emmanuel Chauveau, Pascal Filoche, and Thomas Lavergne. Tracking web spam with HTML style similarities. *TWEB*, 2, 02 2008.

[19] Li Weng and Bart Preneel. A secure perceptual hash algorithm for image content authentication. In Bart De Decker, Jorn Lapon, Vincent Naessens, and Andreas Uhl, editors, *Communications and Multimedia Security*, pages 108–121, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[20] Richard K. Wortley and Stephen Smallbone. *Internet child pornography: causes, investigation, and prevention*. Praeger, 2012.

[21] Hui Yang and Jamie Callan. Near-duplicate detection by instance-level constrained clustering. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, page 421–428, New York, NY, USA, 2006. Association for Computing Machinery.

[22] Jiří Štěpánek and Monika Borkovcová. Comparing web pages in terms of inner structure. *Procedia - Social and Behavioral Sciences*, 83:458–462, 07 2013.