Identifying direct relatives of a social network site user and obtaining their personal contact details

Rhodes University *

Darryn Cull Yusuf Motara

ABSTRACT

Usage of social network sites has steadily grown in the past few years and with it so has the amount of cyberbullying and online harassment. This paper explores a potential remedy to the problem of cyberbullying using an internet based practice called "doxxing" in order to determine the bullies direct family members through social network sites. To do so, a Google Chrome extension was developed to interact with Facebook to collect data and identify direct relatives of the targeted user. Based on the results found, it can be seen that the application was successful in determining the direct relatives of a Facebook user while achieving the research goal by ensuring any Facebook user can make use of the application in order to counter cyberbullying.

1. CCS CONCEPTS

Thesis classification under the ACM Computing Classification System (2012 version, valid through 2015):
•Human-centred computing — Social networks •Human-centred computing — Social networking sites •Human-centred computing — Social network analysis •*Human-centred computing — Heuristic evaluations •Human-centred computing — Social network security and privacy* •Human-centred computing — Social recommendation **•Information systems — Web applications**

2. INTRODUCTION

In this section the research problem will be described followed by the goals of this research and the planned approach to obtain said goals.

2.1 Problem Statement

Usage of social network sites has steadily grown in the past few years [10] allowing friends and family to stay connected online. What this means is people are able to communicate more easily by sending direct private messages through these sites as well as communicate more publicly by posting on another persons or your own wall, allowing anyone with access to view and comment on the post. This opens people up to public opinion which can be very rewarding at times when the comments are useful or relevant to the post On the other hand, it also allows for negative behaviour. People are now able to voice inappropriate opinions or thoughts to whomever they wish and have access to, which is not in and of itself a bad thing unless it is at to detriment of someone else.

A problem that has become apparent with online social networks is the ever-increasing amount of bullying or harassment taking place on them. Bullying has always been a problem within schools, although now that social network sites exist, the bullying does not end when school does and can now follow the child home. Cyberbullying and online harassment have become a major problem since there it not always someone policing giving complete freedom to get away with saying anything to anyone. Along with this, the internet trend of "trolling" has also taken off which is an excuse given online when someone deliberately provokes a negative emotional response from the target by posting cynical or sarcastic remarks and in some cases socially unacceptable comments.

2.2 Research Goals

The goals of this research is to find a potential remedy to the problem mentioned above providing a way to counter cyberbullying and online harassment. Currently there are very few remedies to these problems for children; one such remedy is to tell your own guardians what has happened asking for their guidance or assistance on how to deal with the problem and possibly solve the problem. This is still a highly recommended approach, along with guardians not allowing their children to use social network site unless the guardian is allowed to keep tabs on what is happening. In the cases where children are allowed to use social network sites freely and get bullied, there is a high likelihood that they will not tell their guardian in the same way that they do not tell their teacher when bullied at school.

The remedy proposed in this research will make use of an internet based practice usually used by hackers called "doxxing." This involves collecting personal information about a target using social media and any other means possible, usually followed by openly revealing the information to the internet. Doxxing may be used as a remedy for online harassment by altering the search parameters to only deter-

*Submitted in partial fulfilment of the requirements for the degree of BACHELOR OF SCIENCE (HONOURS) of Rhodes University

© 2015 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

mine the harasser's direct family members through social network sites. This will allow the victim to send a private message to the harassers direct family members making them aware of what has been said or done, hopefully remedying the problem. In order to do this, a tool will need to be developed that is easily accessible to all users of the social network site. This generally includes all ages ranging from age 13 and up since most social networking sites require the user to be older than 13, although this is not always true as many children under the age of 13 sign up regardless.

2.3 Research Approach

To identify direct relatives of a social network site user, an application must be developed that has access to the site, is simple to acquire and easy to use. Thus the proposed approach is to develop a web browser add-on that can be used to target a user in order to determine their relatives. The reasons for this approach are the accessibility of add-ons through browser specific web stores, the straightforward installation process of simply clicking a button and finally the inherent connection to the internet. Considering the time constraints of this project, the scope of the application will be limited to a single browser and a single social network site; the browser being Google Chrome and the social network site being Facebook. The reason for using Google Chrome as the selected browser is due to its convenience of automatically synchronising browsers which synchronises workspaces plus its well designed developer mode and the reason for choosing Facebook is due to how widely used it is plus the rich data it provides. Using the internet connection, the application will collect data on the targeted user by requesting the information from the social network site. The data collected will then be analysed using graph theory and heuristics to determine who the direct relatives of the targeted user are. Once done, the application should show the results with the option to contact the resulting direct family members.

3. RELATED WORK

In this section, literature relating to the design and development of applications to analyse, visualise and extend social networks will be reviewed. What did they do? What are their primary results? Which challenges did they overcome? Which challenges remain? Which algorithms were developed? What analysis was done? What previous work did they rely upon? How scalable are their techniques? These are some of the questions to be answered.

3.1 Social Network Analysis and Mapping

[1] designed an application which collects and analyses home pages of users to build a social network using the text on the page, the outgoing links, the incoming links and the mailing lists they are subscribed too. The text on the page provided a "semantic insight into the content of a user's page" [1] which helped identify the types of links produced by the incoming and outgoing links to other pages. The mailing lists a user was subscribed to provided a community structure the user was already a part of. The information gathered after collection and analysis was then used to visualise a user's social network. [1] found that the application produced many false negatives, meaning a user gets matched with someone they know except there is no explicit link confirming the relationship. Another problem encountered by the authors was that not all students had personal home pages, which causes missing nodes in the mapped network, creating a lack on information. The authors noted that for the application to scale from a university setting to the entire globe would require changes to the analysis and assumptions made; this included adding more data sources such as demographic information, as well as accounting for the number of possible relationships where a user can be a fan of someone (one of thousands) or family member (one of a few). The conclusion [1] came to was that "not only is it possible to find communities, but we can describe them in a non-obvious way."

[5] used social network analysis to map networks of terrorist cells using data collected from news articles, search engine results and publicly released documents. The three major issues covered in their work were

- Incompleteness – the inevitability of missing nodes and links that the investigators will not uncover.
- Fuzzy boundaries – the difficulty in deciding who to include and who not to include.
- Dynamic – these networks are not static, they are always changing. Instead of looking at the presence or absence of a tie between two individuals, looking at the waxing and waning strength of a tie depending upon the time and the task at hand.

[5] came to the conclusion that in order to successfully map a terrorist network, the agencies and/or countries involved in combating the terrorist cell need to share information and knowledge so "a more complete picture of possible danger can be drawn."

3.2 Recommender Systems

Given a snapshot of a social network at a given time, it is possible to predict likely interactions between users. The "link-prediction problem" as [6] put it, is "based on measures for analysing the proximity of nodes in a network." The authors show how this can be done in many different ways using already existing techniques found in graph theory and social network analysis producing varying results. They used methods based on node neighbourhoods such as common neighbours [8], which scores two users similarity based on the number of connections they have in common, and Jaccard's coefficient [9], which measures the probability that two users have a given neighbour from a randomly selected neighbour from either user. Other methods based on the ensemble of all paths were used such as the Katz coefficient [3], which defines a measure that directly sums the collection of paths, exponentially damped by length to count short paths more heavily. Some higher-level approaches were also used, including low-rank approximation, unseen bigrams and clustering. [6] found that "there was no single clear winner" but "there is indeed useful information contained in the network topology alone." [6] conclude by saying "there is clearly room for improvement in performance" considering the highest performing method (Katz clustering) is "correct on only about 16% of its predictions."

A similar study was done by [2], focusing more on the similarity between two users based on the knowledge of social ties existing among users and the analysis of activities users are involved in. They use this data to draw a local measure of similarity between the two users, then consider

the network as a whole to obtain a global measure of similarity based on the Katz coefficient. Finally they combine the scores of similarity into a unique value by applying linear regression. [2] found that applying a global measure of similarity can "partially correct errors produced by each single activity" implying that a collection of techniques working together outperforms any single technique. The authors mention future work will include applying their research to the link-prediction problem, using similarity as a measure of proximity in the network.

[4] designed an application, called ReferralWeb, similar to that of [1]. However, instead of focusing solely on analysing and visualising the network, a recommender system was built on top. ReferralWeb is "an interactive system for reconstructing, visualising, and searching social networks" [4]. The system is used specifically on academia for searching through papers, articles and the like to build a graph of references, allowing a user to search for academic writing by a particular person, referenced by a particular person, or referencing a particular person. [4] state that "by instantiating the larger community, the user can discover connections to people and information that would otherwise lay hidden."

3.3 Third Party Applications

[7] developed and launched three applications using the Facebook Developer Platform in order to collect data on the usage of these applications. The applications gained a combined user base of more than eight million users, analysing the rich data procured based on geographical distribution of users, user interactions and modelling these interactions through interaction graphs. The three applications developed included a social gaming application called "Fighter's Club" in which users pick virtual fights with their Facebook friends that last from 15 to 48 hours and allow the aggressor and defender to request help from friends to become supporters. The other two applications developed were non-gaming applications; "Got Love" was designed to allow users to pick and display a distinct set of 'loved' friends on their profile page, "Hugged" was designed similar to that of Facebook Pokes where a user can send a virtual 'hug' to a friend (this can be done multiple times). One of the key findings is that "application dynamics can significantly affect the structure of interaction graphs, hence weakening the association between them and the underlying real-world (friendship) relationships between users" [7]. This finding is based on the fact that users of the social gaming application would often Friend strangers in order to increase the number of supporters they can gain distorting the natural community structure. On the other hand, they found that non-gaming applications tend to exhibit strong community structures.

3.4 Facebook Friend Mapper

Facebook Friend Mapper[1] is a Google Chrome Extension designed to generate the friends list of a user who is using privacy settings to hide their friends list. In order for it to work, a user must have at least one mutual friend with the target user. It works by using the mutual friend to see which friends they have in common with the target user, generating a partial list of the targets friends. It then uses the generated list to find users whose privacy settings allow the application to view their friends list, applying the previous method again to generate more of the targets friend list. The application continues doing this until a defined depth is reached, and once completed it outputs the list produced.

[1]https://chrome.google.com/webstore/detail/facebook-friends-mapper/ikfdhlkcdllmkklmdbhfjkofjmehionn

4. IMPLEMENTATION

In this section, the design choices made will be implemented and high level descriptions of how they were achieved will be given along with changes to the design made during implementation. Obstacles encountered during implementation will also be mentioned together with the solution found to overcome the obstacle. The section will follow the path of implementation taken by the developer; starting with input, moving to data collection, then heuristics and finally output.

4.1 Input

In order to get user input working, buttons needed to be added to every post and comment. To accomplish this within the content script, four functions were written; two of which respond to events calling the other two for adding the buttons. The two event listeners used both come from the `window.*` library; the first being an `onload` event which gets called every time a Facebook page is loaded and the second event being a timed event which occurs every 2000 milliseconds. Due to the use of the manifest file included with the extension, these functions only run on pages served by Facebook based on the permissions set. The contents of these two functions are similar, they both search the current page for "Like" buttons, one as part of a post and the other as part of a comment passing the found nodes to the other two functions mentioned. These functions then add a "Dislike" button providing a unique class name to posts and to comments respectively allowing the application to tell where a button press originates from. The reason for searching for "Like" buttons on the page simplifies the problem of knowing where buttons are needed or not needed based on whether the current user has enough privileges on the current page. With this done, buttons have been added everywhere needed, but there is no way to respond to a button press. So to do this, another two functions were written using the jQuery libraries `.on()` method which given the body of the page as the origin of the event along with the type of event, a selector for each unique class created when generating the buttons and finally a callback function which will be executed when the event occurs. With all of the above implemented, the application can dynamically add buttons to any Facebook page and begin execution once a button is pressed. A benefit to this approach is that when the callback function runs in response to a button press, the function knows where on the page the button got pressed based on the unique class and so can capture the post and comments which includes the abusive post/comment being targeted along with the targets Facebook user name.

4.2 Data Collection

After getting input working, the application already had some relevant data which could be used to gather more. The information gathered through user input included the target users, as well as all users included in the post, user ID and name. To gather more information, the application needs to make requests to Facebook. To do so, the application

uses jQuery's `.get()` method which, when supplied with a sub domain of the current domain and a callback function, asynchronously makes the request, allowing the application to continue running until the response is received and the application is idle, at which point the callback function gets run, parsing the response. For each page containing needed information that is unique to the others, a new function will have to be written since parsing the page will vary based on the given page. The following subsections detail the different functions written to gather data.

4.2.1 Friend List

To be able to determine a users family members, a list of candidates would be needed. Firstly, the application will make the assumption that the target user is friends with their relatives on Facebook since finding a direct relative with no connection to the target would be more difficult task. If the target's friend list is retrieved, the application can use it as the list of candidates. To do this, the application will use a feature on Facebook which allows a user to see the mutual friends between two users that can be found by navigating to a URL, https://www.facebook.com/[ID1] ?and=[ID2]&sk=friends, which includes the user IDs of the two users in question. Using this, the application can generate the majority of the target user's friend list by using the target's user ID along with a known friend of the target. If the target user made the original post, it can be assumed that the user of the application is a friend of the target and so can use "me" as the known friend. Otherwise, if the target user only commented on a post, it can be assumed that the target user is friends with the poster and so the poster can be used as the known friend. These assumptions were made based on the general format of a user's Facebook feed and what post make it into said feed. Having the target users ID and a known friends ID, the application can request the mutual friends they have using the already described method. Once the response has been parsed and the IDs of all their mutual friends have been stored, the application can use these new IDs to request more mutual friends. To do this, once the function is completed, it will recursively call itself passing a new ID to be requested every time until all unique friends have been used as the known friend in the request. Once the entire process is complete, the application has gathered a list of the targets friends, although the list is not necessarily complete. The reason the list isn't complete is due to outliers in the targets friend list, outliers being friends or groups of friends that are connected to the target user by a single link. In other words, outliers are people who are friends with the target user, but not friends with any of the target users other friends. However, this is not likely a disadvantage or obstacle, since it is unlikely that the target's direct family members fall into this category although it is possible.

4.2.2 Profile

With more information, better heuristics can be run and a more correct solution can be found. On Facebook, the best source of information about a particular person is their profile page. So to gather this information a function was written which, given a user ID, will request the profile page of the user and parse it for as much information as possible. A problem with collecting data from Facebook profile pages is that, depending on the relationship between the target of the function and the application user as well as the privacy settings of the target, much of the information available of a profile page could be unavailable to the application. The function written to do this uses the same method previously described at the start of this section. One section of a given profile page stands out from the rest, which includes details about a user's family and relationships. This page includes the names and IDs of every user the target user claims to be their family. However, this does not guarantee these users are actually family members since there is a known trend on Facebook to add a good friends as a family members on this page.

4.2.3 Obstacles Encountered

After implementing the data collection, it was found that the application was synchronising the asynchronous requests sent. In other words, the application would send a request asynchronously, but instead of continuing computation the application waited for the response to arrive and be dealt with before moving to the next piece of code. This is a waste of time and resources, and so was fixed allowing the application to continue running other logic while waiting for responses. This allowed the application to start running heuristics based on what information had already been collected while waiting for more information to arrive.

When implementing the parsing of mutual friends and and the target users family and relationships profile page, it was found that the data relating to other users arrived commented out within the HTML of the page. When requested by a browser, Facebook hides the sensitive information contained in the HTML within a comment during transport; after the browser receives the HTML, a script is run to uncomment the HTML just before the browser displays the HTML to the user. The reason behind this might be to prevent generic parsers from being able to read the information. However, using methods from jQuery, the application was able to read this information by searching the HTML received for the HTML tag denoting a comment and manually uncommenting it creating a sub-string producing the HTML within. This HTML was then passed to jQuery allowing the application to parse it normally.

Due to the possibility of privacy settings being enabled on the targets profile page and the chance that the user did not specify particular details, measures had to be taken to catch the possibility of the information not being available. If a page is requested and parsed for a particular element, and this element isn't there; depending on the specifics of the operation the application could through an error or assign the data as undefined. To prevent these issues, the application was programmed to deal with all possibilities of information availability. As an example; on a users family and relationships profile page, they can set the privacy settings to not show their relationship but still show family members or vice versa meaning if the application looks for both and only one is found it must still know which section the information came from while not causing errors immediately or further in computation.

4.3 Heuristics

During the data collection process, heuristics are run simultaneously due the asynchrony of JavaScript requests. This makes the application more efficient by allowing data received to be processed while more data is being collected,

and if more data is needed, the application will continue to wait until the data is received moving to an idle state allowing the AJAX callback functions to be called. The heuristics included in the application are run on each user, and if the heuristic is successful, a score is added to said user based on the strength of the heuristic and the strength of relation between the users. These score will help the user of the application decide whether family members outputted are more or less likely to be actual family members. Intimate relations can be found by the application based on the information retrieved from the targets profile page which include marital relationships and non-marital intimate relationships, although if no information is found the application will not look any further. The heuristics used in the application are applied to both the generated friend list as well as the family members found on the targets profile page. If no family members are found on the user's profile page, the chances of finding a direct family member amongst the friend list is likely to be reduced.

The first heuristic is used during data collection, which is determining family members using the targets family and relationships profile page. This heuristic produces a list of possible family members along with their type of relation to the target user, although in some case the type is omitted. The next heuristic used involves checking whether the target user has the same last name as any of the friends in the generated friends list. If someone is found, they are likely to be a family member although this not guaranteed since friends may have the same last name with no relation, a user may have taken a use name or may not have a last name. Also, this excludes direct family with different last names, such as divorced or remarried parents taking different names. Another heuristic to help solve this problem is checking the profile page of a known family member for their family relations which, depending on the users privacy settings, could produce more possible direct family members since the connection is known.

4.4 Output

In order to get output working, two functions were written. The first function creates an output dialog the moment the application is run styled to match Facebook; this was done so that a loading icon can be shown while the application runs since the computation time can be quite long depending on the number of friends the target user has; thus, something was needed to show the application is actually running. Once the application is finished, the loading icon is removed from the dialog box and the output is added. The output shown includes the application's guesses about who the target user's direct family members are as well as the friend list generated. Each guess provided by the application will be accompanied with a scoring to give an idea of how likely they are to be related. If no information is known about the direct family of the user, the output will reflect this by showing the targets friend list along with zero or no scoring.

5. TESTS AND RESULTS

In this section, the application will be tested on 16 Facebook users where the total number of friends for each user is known. The results produced will be analysed in three categories including coverage, efficiency and accuracy. In each of these subsections, a description will be given on what the category incorporates along with data and an analysis of that data.

5.1 Testing

In order to test the application, data will be needed. To get the data, code was added to the application to produce the necessary data needed for the three categories mentioned. The data required for analysis includes the number of requests sent for family members, the number of requests sent to gather friends, the number of friends found, the number of family members found from the users profile page, the total number of nodes found, the total number of friends the user has, the number of family members found, the number of false positives and finally the number of false negatives. This data will be used in each category of analysis based on its relevance to that category. The users chosen for testing include users with little to no privacy settings as well as users with high privacy settings and was carried out from the perspective of a user friends with all targeted users. The reason the application was not tested on users that are not friends with the application user is so that the values obtained through testing can be checked against actual values only known by being friends with target user.

5.2 Coverage

The application requires nodes in the social network graph of the target user in order to determine the targets direct relatives. If some nodes are left out then the likelihood of obtaining the targets direct family will decrease. So, to determine whether the application covers enough of the graph its coverage needs to be determined. This is done by comparing the number of friends generated by the application to the total number of friends the target user has producing a percentage of friends found for each user targeted during testing. This percentage only includes direct friends found using mutual friends and does not include the total number of nodes found during data collection.

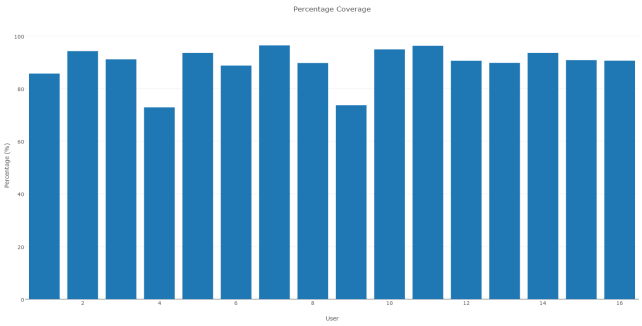

Figure 1: Graph of coverage of friends found.

As can be seen in Table 1, or in Figure 1, the application scores in the top 20 percent coverage on all users bar two with a maximum reaching 96.42 percent. This implies the application is capable of covering a high proportion of the target users social network graph, although, the application did score a minimum of 72.88 percent which means the coverage can vary based on the current user being targeted. With a mean of 89.54 percent and a standard deviation of 6.96 percent, the application's coverage varies between 82.58 percent and 96.50 percent which includes the maximum and

excludes the two lowest users. This means that the application has on average a 10 percent chance to not find the targets direct family members since they may not be covered by the applications data collection. This does not include all data collection, only the traversal of the target user's direct friends and does not include the data collected from users profile pages. The two lowest scoring users also happen to be the two users with high privacy settings which implies privacy may play a role in the application coverage since they are the only two to fall outside one standard deviation.

5.3 Efficiency

For an application to be successful, it must also be efficient by not wasting time or resources. Given that the application collects data by sending requests to Facebook, time is not a good measure of the efficiency of the application due to the non-deterministic number of friends a user can have and instead its efficiency can be determined by comparing the amount of data collected to the number of requests sent. In this case, the amount of data collected is reflected by the number of nodes found in the complete social network graph including the targets friends and the family members found during data collection. The measure of efficiency will be done by dividing the total number of requests sent by the total number of nodes found producing a percentage. This percentage shows number of requests needed per node, thus the lower the percentage is the better the result since less requests were needed to produce more nodes where 100 percent implies one node was found per request sent.

As can be seen in Table 2, or in Figure 2, most users scored between 50 and 70 percent, though four fall outside this boundary. This implies that the application was able to find fewer than two nodes per request sent for these users which also include the two users who scored above 70 percent. On the other hand, the application also managed to score less than 50 percent on two user implying it was able to find more than two nodes per request sent. It should also be noted that the application sends a request for every unique friend found, thus the number of friend requests sent and the number of family requests sent will always be similar to the number of friends found. With a mean of 60.25 percent and a standard deviation of 7.86 percent, the requests per node varies between 52.39 percent and 68.11 percent. It seems efficiency varies significantly depending on the current target user with a quarter of the users falling outside one standard deviation from the mean.

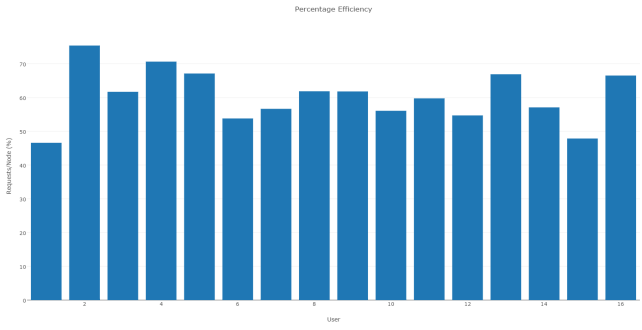

Figure 2: Graph of efficiency of requests per nodes found.

5.4 Accuracy

To determine the correctness of the results produced by the application, its accuracy must be calculated. Accuracy is the degree to which the result conforms to the correct value; in this case accuracy will be determined by summing the number of true positives and true negatives dividing this result by the sum of true positives, true negative, false positives and false negatives. True positives will be considered as correctly determined family members, true negatives as all friends determined not to be direct family, false positives as those incorrectly determined to be direct family and finally false negatives as direct family members incorrectly determined to be not related to the target user. This calculation will produce a ratio out of one where achieving a value of one implies complete accuracy of results.

As can be seen in Table 3, or in Figure 3, the accuracy of the application is exact for just over half the users since seven out of the twelve scored a perfect ratio of one while the remaining five scored within 0.02 of one. These ratios show that the application can accurately reduce a large friend list to a short list of likely family members. However, they do not highlight the incorrect results well enough to show how likely the results are to be family members due to the difference in size between the targets total number of friends and the total family members found. In order to more closely examine the results, an alternative ratio was calculated to try highlight the incorrect results by removing true negatives from the calculation. The alternative ratio can be seen in Table 3 which shows the variation in accuracy for the less than exact results ranging from 0.50 to 0.90. This shows that when the application fails to produce completely accurate results, it will only fail on a maximum of half the results produced. Another point to note is that the application only produces one false negative between all 16 users showing that it is highly improbable to miss a direct family member and instead is more probable to produce false positives. Also, the false positives produced by the application are ranked low by the application indicating to the applications user that they are less likely to be related to the target user.

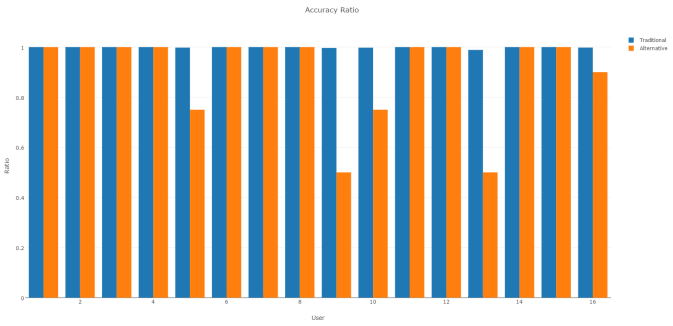

Figure 3: Graph of accuracy ratio.

As can be seen in in Table 3, the application successfully managed to obtain the parents of the targeted users 75 percent of the time. The other 25 percent is made up of four users who are not friends with their parents as can be seen in the table since the only user with a false negative is user nine for whom the application still managed to find a parent. This implies the applications accuracy on finding parents is

100 percent since all users that are friends with at least one parent were found by the application.

5.5 Usability

The ease of use of the application is an important aspect of usability. Since the application is aimed at users of all ages ranging from age 13 and up with the possibility of younger users, the application must be simple to use in order to accommodate all possible users. In order to determine whether the application is usable, the user interface of the application will be analysed to determine how learnable it is, how memorable it is and how easy it would be to make a mistake. These categories will be analysed in comparison to Facebook's user interface as a baseline for a usable interface since all users of Facebook will have a working knowledge of how to use Facebook's user interface.

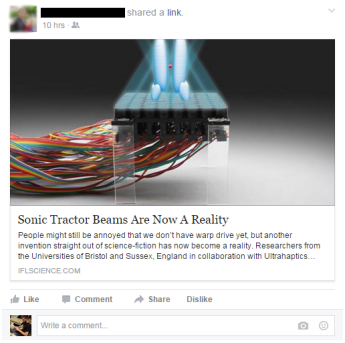

Figure 4: Dislike button added to post.

The application was designed to simplify the process of using it by automatically adding a button to all posts and comments, as can be seen in Figure 4. This simplifies the process by using an already known feature of Facebook, the "Like" button, which users are accustomed to already meaning it should be easy to learn, very memorable and difficult to make a mistake. It also make the application pleasant to use since the "Like" button on Facebook is regarded as a satisfactory feature to which user enjoy using.

After the button is pressed, the output overlay is shown with a moving loading icon known as a GIF. This was done to prevent users from assuming the application is not running since execution time can vary based on the targeted user. This should also prevent users from making errors while using the application by not clicking the added button multiple times which could cause the application to break or produce incorrect results. Once the application finishes executing, the results are shown in the overlay as can be seen in Figure 5. The output produced shows the direct relatives found by the application as hyperlinks to the profile pages of the users with the score produced by the application followed by the rest of the target users friend list. Using a hyperlink shown as the user's name is a common occurrence on Facebook so this should be memorable to users of the application making it simple to understand how to proceed. In order to contact the intended user; the application's user need only click on the hyperlink, which will redirect to the intended user's profile page where the application's user should be familiar enough with general layout of a Facebook profile to find the "Message" button. This will allow the application's user to contact the intended user in a way that is familiar to a Facebook user which should improve the usability of the application by not forcing the application's user to learn anything new and using memorable details of the Facebook user interface.

Figure 5: Output overlay showing output.

6. CONCLUSION

Testing was carried out on 16 users including users with little to no privacy settings as well as users with high privacy settings and was executed from the perspective of a user who is friends with all targeted users. This was done so that values such as the total number of friends each user has can be determined by viewing the users profile page. The results produced fall into three categories: coverage, efficiency and accuracy.

6.1 Coverage

Coverage determines the percentage of a users friend list generated by the application and is used to show the likelihood of the application missing a direct relative by not covering them during data collection. The results for coverage show that on average the application generates 89.54 percent with a standard deviation of 6.96 percent meaning the variation in coverage is quite low although this also implies that the application is in general missing 10 percent of the targeted users friend list producing a 1/10 chance to miss a direct relative. It must also be noted that the friends not being covered by the application are very likely to be nodes in the social network graph that have no other connections to the targets graph whether singular nodes or cliques. This implies that if a family member is found, the chances of other family members falling into this 10 percent are quite low since connections can be made using the known family member. Thus to conclude, the application seems to have a successful coverage of the target users social network.

6.2 Efficiency

Efficiency is calculated by determining the requests per node found and is used to show the amount of work needed to produce a social network graph. The results of efficiency show that on average the application sends 0.6025 requests per node found which implies the application is very efficient considering this means it takes three requests to produce five nodes. Although, with a standard deviation of 0.0786 requests per node, the efficiency of the application does not vary by much implying the applications efficiency is deterministic. This can also be shown by considering that the application sends a request for more friends and for family information every time a new unique friend is found meaning the number of friend requests and the number of family requests will always be close to that of the total friends found. The two lowest scoring users, the only two to fall outside one standard deviation from the mean, also happen to be the two users with the highest privacy settings. Although,

due to the low number of test cases, this can only imply that privacy settings affect the applications efficiency and further research would need to be done to prove this. To conclude, the applications efficiency is not optimal, however, efficiency was not the goal of the application.

6.3 Accuracy

Accuracy is used to determine the correctness of the results produced by the application and is the degree to which the result conforms to the correct value. To calculate accuracy, two formulas were used. This includes the conventional calculation of accuracy determined by summing the number of true positives and true negatives dividing this result by the sum of true positives, true negative, false positives and false negatives. Also included is an alternative calculation used to highlight false negatives by removing true negatives from the aforementioned formula. Using the conventional calculation of accuracy it has been shown that the application is highly capable of correctly reducing a large number of friends into a short list of likely family members. Although, using the alternative calculation of accuracy it has also been shown that the application produces many false negatives with some users scoring only 50 percent accuracy. However, the application gives each family member generated a scoring of their likelihood to be a direct family member, and in these cases, the false positives produced always score low in this respect. Thus, the applications accuracy is in general relatively high especially when considering that over half the users tested scored perfect accuracy using both calculations.

6.4 Final Statement

Based on the results described above, it can be seen that the application was successful in determining the direct relatives of a Facebook user. The application can cover enough of the targets social network graph to accurately determine their direct relatives albeit not efficiently. The application also seems to be sufficiently usable by the target audience by making use of known features within the Facebook user interface to simplify the application's user interface. This makes the research goal more achievable by ensuring any Facebook user can make use of the application in order to counter cyberbullying or online harassment.

6.5 Future Work

There are many possible additions that could me made to the application produced as well as much more research to be done in this field. Firstly, the application could be improved by collecting more data about each individual user which can then be used to add more heuristics. The amount of possible data that could be collected is near endless including information about work and education, places lived and personal information such as date of birth or religious views. This can all be found on a users profile page depending on privacy settings and/or the relationship between the application and the target user. More heuristics could be added using this information such as checking age differences, similar religious views or whether they have lived in similar locations. On top of this, the user's wall could be scraped for information on who they communicate with and what they communicate about. A more in-depth addition that could be made to the application is to integrate other social network sites to gather even more information such as LinkedIn which contains plenty of freely available information. Further research that could be done it to possibly field testing the application to see how well it remedies online harassment or cyberbullying using a test group such as a high school or a university campus. Other future research to be done could be to further test the boundaries of privacy settings by testing the application on a group of willing strangers with little to no connection to the application user to see if the application maintains its accuracy.

7. ACKNOWLEDGMENTS

I would firstly like to thank my supervisor, Mr. Yusuf Motara, for his mentorship and guidance throughout this research. His advice and counsel have been invaluable to me and his knowledge and expertise played a key role in this research. I also thank my family for their emotional and financial support throughout the year. Without the support of them I would not be where I am today giving me the strength and will I needed to complete this research. I would like to acknowledge the financial and technical support of Telkom SA, Tellabs, Easttel, Bright Ideas 39, THRIP and NRF SA (UID 75107) through the Telkom Centre of Excellence in the Department of Computer Science at Rhodes University.

8. REFERENCES

[1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[2] P. De Meo, E. Ferrara, and G. Fiumara. Finding similar users in facebook. *Social Networking and Community Behavior Modeling Qualitative and Quantitative Measurement, IGI Global*, pages 304–323, 2011.

[3] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[4] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[5] V. E. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

[6] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[7] A. Nazir, S. Raza, and C.-N. Chuah. Unveiling facebook: a measurement study of social network based applications. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 43–56. ACM, 2008.

[8] M. E. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):25–102, 2001.

[9] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1983.

[10] Statista. Statistics and facts about social networks. http: //www.statista.com/topics/1164/social-networks/, Feb 25 2015.

APPENDIX

A. TABLES

User	Total Generated	Actual Total	Coverage (%)
1	288	336	85.71
2	671	712	94.24
3	195	214	91.12
4	266	365	72.88
5	450	481	93.56
6	71	80	88.75
7	350	363	96.42
8	812	905	89.72
9	423	574	73.69
10	391	412	94.90
11	129	134	96.27
12	202	223	90.58
13	395	440	89.77
14	247	264	93.56
15	188	207	90.82
16	444	490	90.61

Table 1: Table of coverage of friends found.

User	Friend Requests	Family Requests	Nodes	Requests/Node (%)
1	285	286	1226	46.57
2	668	669	1774	75.37
3	195	196	634	61.67
4	265	266	752	70.61
5	447	448	1334	67.09
6	67	68	251	53.78
7	339	340	1199	56.63
8	807	808	2612	61.83
9	423	424	1371	61.78
10	391	392	1397	56.05
11	130	131	437	59.73
12	201	202	737	54.68
13	391	392	1171	66.87
14	246	247	864	57.06
15	188	189	788	47.84
16	441	442	1328	66.49

Table 2: Table of efficiency of requests per nodes found.

User	Total Friends	Family Found	F/Positives	F/Negatives	Ratio	Alt Ratio	Parent (Y/N)
1	336	5	0	0	1.0000	1.00	Y
2	712	9	0	0	1.0000	1.00	Y
3	214	1	0	0	1.0000	1.00	N
4	365	7	0	0	1.0000	1.00	Y
5	481	4	1	0	0.9979	0.75	Y
6	80	5	0	0	1.0000	1.00	Y
7	363	15	0	0	1.0000	1.00	Y
8	905	7	0	0	1.0000	1.00	Y
9	574	3	1	1	0.9965	0.50	Y
10	412	4	1	0	0.9976	0.75	N
11	134	2	0	0	1.0000	1.00	N
12	223	4	0	0	1.0000	1.00	Y
13	440	10	5	0	0.9886	0.50	Y
14	264	4	0	0	1.0000	1.00	N
15	207	3	0	0	1.0000	1.00	Y
16	490	10	1	0	0.9980	0.90	Y

Table 3: Table of accuracy ratio.