Investigating Privacy Concerns

Christopher Castillo
Loyola Marymount University, Los Angeles, CA

Jennifer Felder
North Carolina State University, Raleigh, NC

Rafael J. Negrón
University of Puerto Rico, Mayagüez, PR

Anand Sonkar
Arizona State University, Tempe, AZ

Faculty Mentor: Prof. Deirdre Mulligan
School of Information, UC Berkeley

Graduate Mentor: Jen King
School of Information, UC Berkeley

Graduate Mentor: Nick Doty
School of Information, UC Berkeley

Team for Research in Ubiquitous Secure Technology (TRUST)
University of California, Berkeley
30 July, 2010

This work was supported by the TRUST Center (NSF award number CCF-0424422)

As technology advances, the way in which privacy is both protected and violated has changed with it. By analyzing empirical data--in our case, real world complaints about information privacy issues on the internet, we can seek to understand the public's current privacy concerns. The goal of this project was to create a command line executed tool written in python to query Yahoo! Answers to obtain relevant privacy complaint data for analysis using data visualization tools. From this research, we found that certain keywords were more efficient in collecting data, and that there are consistent relationships between individual terms and phrases in Answers data. Both of these results are essential in further refining not only the terms used to obtain data but also in producing a taxonomy of privacy terms.

Overview

Individuals' privacy concerns emanate from both online and offline sources: information sharing on social networking sites, new location-aware services, and offline practices such as video surveillance. The privacy concerns and objections expressed by individuals often fall outside existing legal protections, and while multiple theories of privacy—and its relationship to technology—have been proposed, little empirical effort has been undertaken to document and understand how they relate to the concerns expressed by individuals today. It is unclear how existing theories and legal protections that reflect them relate to individuals' conceptualizations of privacy. What problems do individuals perceive as "privacy" problems? How do they frame and articulate their objections? What animates their concerns? What does this tell us about the sufficiency of existing theory, policy and technical approaches to privacy protection?

To explore these questions, Professor Deirdre Mulligan and PhD candidate Jennifer King are conducting a multi-year empirical investigation of several datasets that contain information about the privacy objections individuals raise in the web 2.0 environment. The project aims to fill an important gap, using available data sets, surveys, and interviews to understand existing conceptions of privacy and the privacy management strategies employed by individuals. This research will form the basis of several papers that will: review and report our findings; use the findings to explore the utility of competing theoretical models of privacy and perhaps propose our own; examine how existing legal protections respond to the privacy concerns identified through our investigation; consider options for addressing privacy concerns through technology design, company policies, norm development, and public policy. Preparatory work will develop taxonomy of theories of privacy, as well as a more expansive taxonomy of privacy itself, to ground the initial assessment of data sets and examine the utility of existing theory.

Problem Domain

As part of this project, this team began by examining publicly available data sets to see if reliable sources of public data were available that captured and articulated the public's privacy concerns. Two of these sources were Facebook and Yahoo! Answers. More specifically, the investigation of the data available on Facebook included user comments on the Frequently Asked Questions page, group pages, comments on company blogs, and surveys and polls available on the site. To ensure that all data found on Facebook was publicly available, the search was

This work was supported by the TRUST Center (NSF award number CCF-0424422)

completed without logging into the site. After searching Facebook, a preliminary examination of the content available on Yahoo! Answers was conducted.

This examination began with finding questions that were both relevant and irrelevant to privacy complaints. From this initial data characteristics of relevant and irrelevant questions were compiled, and these characteristics were used to narrow the terms used to search for questions in later research. Along with these general characteristics, a list of the categories that contained the most relevant questions with respect to privacy was created and used to further refine searching.

After investigating both of these sources of public data, Yahoo! Answers was chosen as the final source of privacy complaints for several reasons. First, it provided a free API that contained a large amount of relevant data. Second, its API was easily implemented in a python script, which was the basis for automating the data collection stage of the larger project. Finally, because it included publicly available data, having it as a source eliminated any concerns about privacy violations arising from this research.

Yahoo! Answers

A. Overview & Evaluation

Yahoo! Answers, formerly known as Yahoo! Q & A, is a community-driven question-and-answer site launched by Yahoo! on July 5, 2005 that allows users to both submit questions to be answered and to answer questions asked by other users. Yahoo! Answers was created on the three basic pillars of ask, answer and discover. These pillars can be described as follows: ask a question on any topic that matters to you, so that other people can give you answers; answer someone's question and make their day and, if you have to, discover by browsing the answered questions; tap into the wealth of ideas and experiences that people have shared.[5] Yahoo Answers! consists of numerous categories and subcategories in which users can ask or answer others' responses.

Yahoo! Answers allows all Yahoo! users to ask and answer questions. The original questioner then chooses the best answer. Points are awarded for questions being asked and answered, as well as for having one's answer selected as the top one. To encourage participation and reward great answers, Yahoo! Answers had created a system of points and levels.[1] The number of points awarded depends on the specific action taken. While you cannot use those points to redeem or buy anything in reality, they do recognize active users who answer questions.

"Levels are another way to keep track of how active you and others have been. The more points you accumulate, the higher your level. Yahoo! Answers recognizes your ranking with our special brand of thank you's!"[5] As you attain higher levels, you will also be able to contribute more to Yahoo! Answers, asking, answering, voting on and rating questions more frequently.

The Yahoo! Answers community has set guidelines for their users. Any question is allowed on Yahoo! Answers, except ones that violate the Yahoo! Answers community guidelines. To encourage good answers, helpful participants are occasionally featured on the Yahoo! Answers Blog. Though the Yahoo! Answers service itself is free, the content of answers is owned by the respective users, although Yahoo! maintains a non-exclusive, royalty-free, worldwide right to publish the information. According to the Yahoo! Answers Community guidelines, using the same username for the chat function as in Yahoo! Answers is explicitly forbidden, accepting the fact that some categories are mostly opinion based. Community guidelines also request users to be courteous, to be clear while asking or answering the question

This work was supported by the TRUST Center (NSF award number CCF-0424422)

and to avoid the use of mean or obscene language; furthermore, exploiting the community is highly forbidden by Yahoo!.

After conducting some preliminary research on privacy and its concerns, Yahoo! Answers was chosen as the best resource available to find day to day user issues with privacy. The Yahoo! Answers database has thousands of privacy issues for which our team was searching. Querying Yahoo! using the term 'privacy' resulted in thousands of results being returned, out of which a significant portion of them were irrelevant. In order to get more relevant results for the research, our team decided to generalize the few categories that gave results focusing more on individuals' privacy concerns. Some of these categories included subcategories that yielded an even greater portion of results pertaining to privacy. During this preliminary analysis our group also looked for the terms and phrases that either meant or were closely related in meaning to the term 'privacy'. The list compiled from this analysis included terms like 'security', 'intrusion', 'spying', 'hacking', all of which relate closely to privacy.

Analyzing the Yahoo! Answers website helped our team derive the specific words, terms, and phrases that mean privacy. Using some of the promising categories like 'Computers & Internet' and 'Politics & Government' helped us narrow our search to collect more relevant data. Doing so was especially necessary, as the emphasis of the "Yahoo! Answers site being more on social networking than providing accurate information"[3], as criticized by the Wall Street Journal. Dividing our pre-research analysis to look for relevant and irrelevant results helped our team review, refine, and document the results.

B. Requirements

After the preliminary research, several requirements were determined to be necessary for the command line tool that would be later constructed. These requirements were developed in stages. The first requirement set for this tool was to be able to query Yahoo! Answers for a single term and receive questions based on that keyword search. In addition to the keyword, several other parameters were also specified, including the number of results per query returned and a Yahoo! app ID. After that stage was completed, the second requirement set for the tool was to not only query Yahoo! Answers for multiple terms, but also store the results of those queries in a database, specifically a MySQL database in this case.

Once a functional script that could successfully query Yahoo! Answers for various parameters and store the results in a database was created, establishing a foundation for the tool, the script was further refined and other requirements became necessary. The tool needed to be continuous, executing the previously described tasks either until stopped manually or by an error. To further automate the script, the tool created also needed to be able to be run on the School of Information server automatically at a specified time or interval of time, eliminating the need for an individual to manually execute the script.

C. Technical Design

The initial process was to connect and query Yahoo! Answers for specific keyword and store the result into the MySQL database. The flowchart in figure 1 summarizes the process of how the script runs when executed, with the purpose of gathering results to be used in future analysis.

This work was supported by the TRUST Center (NSF award number CCF-0424422)

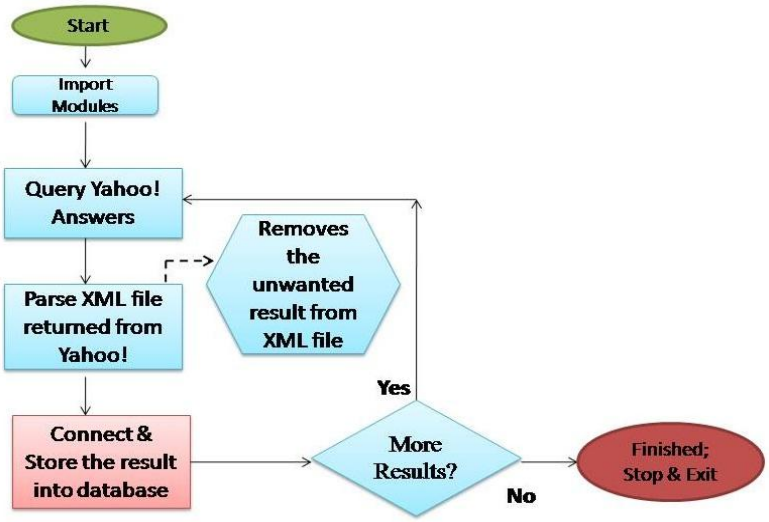


Figure 1. Flowchart Illustrating the Method to collect data.

Results

I. Technical Results

The work was divided into discrete components: connecting to Yahoo! and parsing the data, database configuration, error handling, logging, and script refinement.

A. Connecting to Yahoo! Answers and Parsing the Data

In order to gather some results to later analyze, our team first needed to connect to the content of Yahoo! Answers to obtain some results. This was accomplished using the Application Programming Interface (API) of Yahoo! Answers, which offers to developers the ability to use their content and services to build their own application. This API also allows developers to choose from four different web services: questionSearch, getByCategory, getQuestion, and getByUser. Each of these is available independently through a unique URL and offers different, useful ways of allowing developers to use the Yahoo! Answers content. Specifically for this project, we chose to use the questionSearch API, as its function is to read a query and return questions based on the specified term.

The API also has certain parameters that can be sent to Yahoo! Answers that are either required or refine the search. The parameters that we used were as follow: query (search term), search_in (the ability to only search in "all", "question", or "best answer"), sort (enables sorting results by relevance, date descending, or date ascending), appid (developer's application id provided by Yahoo! upon registration), start (desired beginning question at which to begin the search), and results (number of questions to be returned). The API has several different types of files that can be chosen from for the format of the returned results. Out of the options available from Yahoo!, the XML type proved to be of greatest use in this project, as it is an easy to use, well structured format that is used primarily to store data. It also allows users to create their own tags, which was later taken advantage of when creating the files used to execute the script. The first file created was 'config.xml', which contains the parameters necessary to connect to

This work was supported by the TRUST Center (NSF award number CCF-0424422)

individual MySQL databases. The second file used was ‘categories.xml ‘, which consists of a list of categories and sub-categories that were determined in our preliminary research to be relevant to our research. Using these key components of the Yahoo! Answers’ API, we were then able to connect to Yahoo! Answers.

Connecting to Yahoo! Answers was a fairly simple task. The code internally contained ‘urllib’, a predefined set of functions in the Python database. The main two functions used from this library were ‘urlopen()’ and ‘urlencode()’. The ‘urlopen()’ command enabled us to open for reading a network object as denoted by the Request URL (questionSearch). Since the desired return file type was XML, results based on the specified parameters were in this format. By using ‘urlencode()’, we were able to convert the parameters to a URL-encoded string that is capable of being passed to ‘urlopen().’

One problem encountered was that the results gathered contained all the information contained within the API. As most of this information was irrelevant to our research, we parsed the data, storing only the information necessary for our research. The elements relevant to our project were as follows: Question id, Subject, Content, Timestamp, Link, Category, and Chosen Answer. Limiting our data to this information made it possible to analyze our data and arrive at a substantial conclusion.

To parse the data, a new function, minidom, was added to the script. Because XML format consists of parent and children nodes, minidom was used to look for a specific node and obtain information from the desired element inside that node. This process is more efficient than searching through the entire file for one specific element, because for a large amount of queries, it greatly decreases the time necessary to extract data. Parsing the data also worked well for manually reading the data. When the data was in XML format, it contained indentations and excessive labels, making it difficult to read. In contrast, after parsing, the data was in an easy to read, plain-text format. After converting the data from XML format, the results were then sent to be stored in a group database.

B. Database Management and Configuration

A database consists of an organized collection of data for one or more users. Our team decided to use MySQL for the storage and logging of user data. MySQL is a relational database management system (RDBMS) that runs as a server, providing multi-user access to a number of databases.[1] MySQL database has become the world’s most popular open source database because of its consistent, fast performance, high reliability and ease of use. In fact, several high-traffic web sites, including Yahoo! Answers, use MySQL for storing data.

Because it was open source, MySQL was used to store the results obtained from executing the script. The database is located on the server at the School of Information at UC Berkeley. Based on the categories from Yahoo! Answers that were described earlier, a table was created to store the results in a form similar to that used by Yahoo!. The architecture of the table created to store the results is shown in figure 2.

This work was supported by the TRUST Center (NSF award number CCF-0424422)

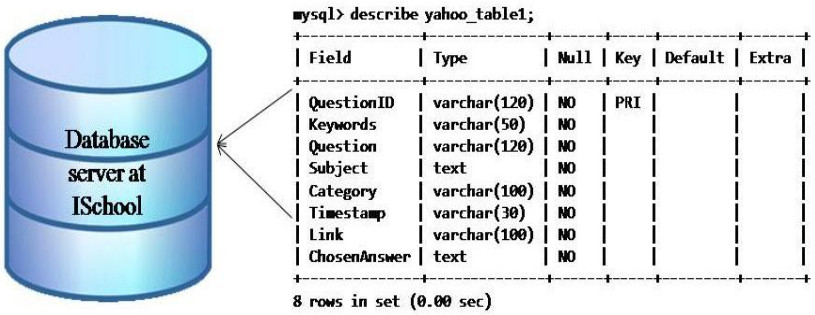
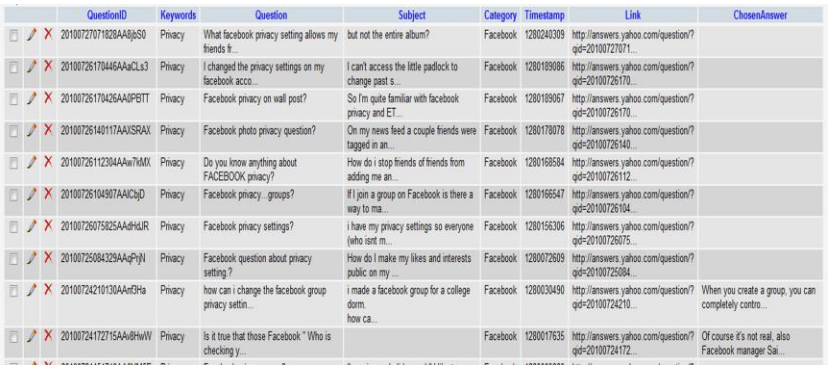


Figure 2. Table architecture for storing the result into Database.

Once the table was properly created and structured, the script was executed several times to collect data. As previously discussed, the basic concept of the Python script was to connect to the Yahoo! Answers server, parse the results of different categories, and store into the database shown in figure 2. Currently, there are over 7,000 results stored in the database; these were collected after executing the script few times. Figure 3 shows a small portion of the results that are currently stored in the database server located at UC Berkeley.



	QuestionID	Keywords	Question	Subject	Category	Timestamp	Link	ChosenAnswer
	20100727071828AABjbS0	Privacy	What facebook privacy setting allows my friends fr...	but not the entire album?	Facebook	1280240309	http://answers.yahoo.com/question/?qid=20100727071...	
	20100726170446AAaCLs3	Privacy	I changed the privacy settings on my facebook acco...	I can't access the little padlock to change past s...	Facebook	1280189086	http://answers.yahoo.com/question/?qid=20100726170...	
	20100726170426AA0PBTT	Privacy	Facebook privacy on wall post?	So I'm quite familiar with facebook privacy and ET...	Facebook	1280189067	http://answers.yahoo.com/question/?qid=20100726170...	
	20100726140117AAISRAX	Privacy	Facebook photo privacy question?	On my news feed a couple friends were tagged in an...	Facebook	1280178078	http://answers.yahoo.com/question/?qid=20100726140...	
	20100726112304AAwiTxMX	Privacy	Do you know anything about FACEBOOK privacy?	How do i stop friends of friends from adding me an...	Facebook	1280168584	http://answers.yahoo.com/question/?qid=20100726112...	
	20100726104907AAICbjD	Privacy	Facebook privacy...groups?	If I join a group on Facebook is there a way to ma...	Facebook	1280166547	http://answers.yahoo.com/question/?qid=20100726104...	
	20100726075825AAdHdJR	Privacy	Facebook privacy settings?	i have my privacy settings so everyone (who isnt m...	Facebook	1280156306	http://answers.yahoo.com/question/?qid=20100726075...	
	20100725084329AAgPrjN	Privacy	Facebook question about privacy setting?	How do I make my likes and interests public on my ...	Facebook	1280072609	http://answers.yahoo.com/question/?qid=20100725084...	
	20100724210130AAnf3Ha	Privacy	how can i change the facebook group privacy settin...	i made a facebook group for a college dorm. how ca...	Facebook	1280030490	http://answers.yahoo.com/question/?qid=20100724210...	When you create a group, you can completely contro...
	20100724172715AAu8HwW	Privacy	Is it true that those Facebook " Who is checking y...		Facebook	1280017635	http://answers.yahoo.com/question/?qid=20100724172...	Of course it's not real, also Facebook manager Sai...

Figure 3. Result stored into the MySQL database server.

C. Errors

When executing the script to gather data, several errors were encountered. Errors appear when unexpected conditions are met while a script is running. These conditions may include non-existent data asked by a specific definition, unexpected or unrecognizable data formats, duplicate data, or syntax errors. By default, when an error occurs, the script stops running and displays in the command prompt a message explaining the details of the error. To avoid this, the 'Try-Except' method was used; this method catches the exception when it occurs and runs another piece of code that is intended to solve the error. Despite this, other errors may occur after an initial error is solved. To account for this, more than one 'Try-Except' case may be added to the script to account for other possible errors. In our script, we used this method to resolve errors that were occurring, including duplicate questions, non-existent answers, non-existent questions, and others.

The first of these errors was adding to the database a question that already existed, which threw a 'MySQLdb.IntegrityError'. Each question received as a response has a unique question id. Therefore, the question id field of the database table was set as unique, thereby accounting for the duplicate error. In using the 'Try-Except', the script ignored the entry and notified the

This work was supported by the TRUST Center (NSF award number CCF-0424422)

user in a log message, which included the question id of the duplicate. The second error encountered was caused by attempting to add a non – existent chosen answers-or 'Open Questions', which created an 'AttributeError'; to fix this, another 'Try-Except' exception was added to the 'chosenanswer' variable.

The third error accounted for was the 'IOError', which resulted from a failure to open either a URL or a file successfully. A 'Try-Except' was added to account for the case when the URL, which is used to communicate with the server, changed, thus making the one being sent unknown to the server. The same exception was also thrown when the file being utilized by the script is not located in the same directory as the script. If either of these conditions were met, an error message appeared in the log file, indicating that either the URL or the file either did not exist or was unable to be opened. Additionally, it was necessary for the log file to be in the same folder as the script; otherwise, an error message appeared on command prompt.

Yet another error dealt with was the 'xml.parsers.expat.ExpatError'. This occurs when the data sent from the Yahoo! Answers server, expected to be in XML format, is returned in an unknown format. When this occurred, as well as when the parameter 'start' was out of bounds, the node 'Error' appeared at the end of the search. When this node appeared in the results, the script stopped querying that particular word, and instead searched for the next keyword if one was available.

A fifth error, called the 'UnicodeEncodeError', appeared when special characters were included in either the subject, content, or chosen answer of a question. As a solution, we included a 'Try-Except' that attempted to encode those attributes of the question into the'utf-8' format, after which it tried to add them to the database once more and informed the user of the error. If the same error occurred a second time, after encoding the special characters, the entry was simply ignored. The last error accounted for was the result of receiving questions that had been deleted from the Yahoo! Answers server; these had a question id of a different form than existing questions. While these still appeared as part of the responses from the server, they raised the 'AttributeError' exception. Instead of adding a 'Try-Except', a condition was added inside an 'if' statement in the script that, if met, ignored the entire entry and continued with the next one. By handling these errors, our team ensured that the script was able to run completely, and that any unexpected or irrelevant data was not added to the database.

D. Log

A log file contains information explaining what is happening when a script is being executed and is an important component of software development. A log was imperative in this application, because the script was designed to be executed automatically, so a user was not required to be physically present at runtime.

Once a log file was created in the same directory as the script, it was then integrated into the script to record each log message. In doing so, three steps were necessary: 1.open the file and set it to append; 2.write the log message, including the timestamp, entry level and explanatory message; 3.close and save the file. The only information stored in the log file was the name of the definition, message and entry level; the timestamp was also automatically included. The three levels of the various messages recorded are as follow: info, debug, and warning. Implementing this log gave us an advantage for debugging, but was not without its limitations, specifically that if the file was not created in the same folder as the script or if it did

This work was supported by the TRUST Center (NSF award number CCF-0424422)

not exist at all, it caused an 'IOError'. Without a log, error messages appeared in the command prompt, making it difficult for the user to discover or understand an error.

E. Refining the Script

After developing a script that successfully queried Yahoo! Answers and stored results in a MySQL database, several changes were made to improve the script. First, several command line flags were added, adding options for how the program was executed. One of these flags was a help flag: if a user entered none of the required flags or used a flag incorrectly, the script exited and displayed a message explaining the use of the flags and providing the contact information of the script developers. Two flags were also added to provide query terms to be sent to Yahoo! Answers: one if these terms were to be supplied from a file and another if a single query was to be supplied from the command line. One of these flags was required for the script to execute successfully. A flag to specify the number of results returned from Yahoo! Answers per query was also added.

There were also two command-line flags added for purposes of analysis that do not actually connect to Yahoo! Answers. First was a database flag, which connected to the MySQL database and prints in the terminal the number of results stored for each keyword specified in the keyword file. Secondly, an option for Many Eyes, an online visualization tool from IBM, was added-running this flag stripped the data in the database of punctuation and printed it in the terminal, which was necessary to use some of the visualizations on Many Eyes.

Another change made to the script was in the function that queried Yahoo! Answers. Two parameters were added to the URL sent to Yahoo! Answers. First, the 'start' parameter was added, allowing the first question searched to be specified, which was comparable to clicking the 'next page' link on the Yahoo! Answers webpage. This allowed the number of results obtained to be increased substantially. To make the results from Yahoo! Answers more consistent, the 'sort' parameter was also added to the URL; in this case results were sorted by descending date.

Furthermore, a loop was created to continuously query Yahoo! Answers until an error message was received that there were no more questions available for a particular keyword: the 'start' parameter was incremented by fifty after each search, and then Yahoo! Answers was queried again, until the error tag was received in the XML file returned by Yahoo! Answers. Finally, a cronjob was set to automatically run the script on the School of Information server. It was set to execute the script every two hours, and was run for three days to collect a set of data substantial enough for some initial evaluation.

These changes were important for three main reasons. First, the command-line flags provided the user with a level of flexibility, making the script useful in several contexts. Second, the parameters added to the URL increased the quantity of data obtained, increasing the validity of the analyses of that data. Finally, the cronjob added a level of autonomy, eliminating the need to manually run the script and allowing the script to run at times of low-traffic on the server, which may be inconvenient times of day.

II. Qualitative Results

Further analysis was done using Many Eyes, a visualization tool developed by IBM and used to discover patterns in bodies of text. The visualizations available through Many Eyes help to collectively interpret data.[2] The question and subject fields of the table containing the data collected were analyzed using four different visualization tools: Word Tree, Tag Cloud, Phrase

This work was supported by the TRUST Center (NSF award number CCF-0424422)

Net and Word Cloud Generator. The purpose of creating different visualizations was to study the results using several different approaches and, through that analysis, ultimately create and refine for future studies a list of terms to be included in the keyword file utilized by the script.

The Word Tree visualization enabled users to search for a word from a data set and see how it was used in context. The visualization began with a blank screen, allowing users to enter a word. Once a word was entered, it displayed the other words and phrases connected to the word in the text being analyzed, illustrating the relationships with branch-like structures, as shown in figure 4. This search was then further narrowed down by selecting a branch stemming from a word, which then displayed the exact context in which that word appeared. This visualization was very useful in examining how a term was used in context within a data set.

Figure 4. Word Tree Visualization

In contrast, the Tag Cloud visualization showed frequently appearing words within the text. The frequency with which a word appeared in the text being analyzed corresponded to the size of that word in the visualization. For example, if the term 'privacy' appeared two hundred times, while 'security' appeared one hundred times, the word 'privacy' appeared twice as large in the visualization. Also, when guiding the mouse over a word, the visualization constructed a small, separate box that included both the total number of occurrences and a portion of six examples in which that word appeared in context within the text being analyzed. The Tag Cloud also enabled the user to display occurrences of either one-word or two-word phrases, as well as the ability to draw a comparison between two different texts. A sample representation of one-word tag cloud can be seen in figure 5.

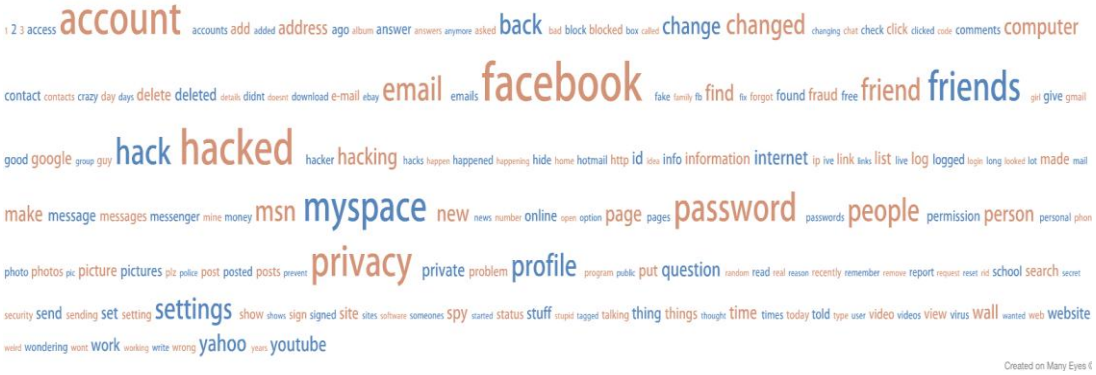


Figure 5. One-Word Tag Cloud Visualization

This work was supported by the TRUST Center (NSF award number CCF-0424422)

While the Word Cloud Generator visualization also had the ability to determine the words most frequently used in a text, it is different from the Tag Cloud. The Word Cloud Generator allowed users to create a more artistic visualization. In addition, it provided the capability to change its layouts, fonts, and color schemes, as illustrated in figure 6. Another useful aspect of the tool was the fact that users were able to maximize or minimize the number of results displayed, which was not possible in the Tag Cloud visualization. It also allows users to exclude common words in the analysis and had the ability to make the words being analyzed all of the same type case.

Figure 6. Word Cloud Generator Visualization

The Phrase Net visualization analyzed and diagrammed the relationship of different words within a text. While there were useful default relationships between words, such as "and", "[space]", and ", "the", it also allowed users to supply their own relationship for which to be searched. The visualization showed these relationships by displaying arrows that linked two words related together. Similar to the Tag Cloud and Word Cloud Generator, in which the number of matches was proportional to its size in the visualization, the size of the arrow connecting the words was proportional to the number of relationships found between the two words. Furthermore, the degree of darkness in which the word appeared was proportional to the frequency with which it was used in the primary search position. This visualization also enabled users to see how the linked words were used in context by moving the cursor over the arrow linking the words. Figure 7 is a representation of this Phrase Net Visualization.

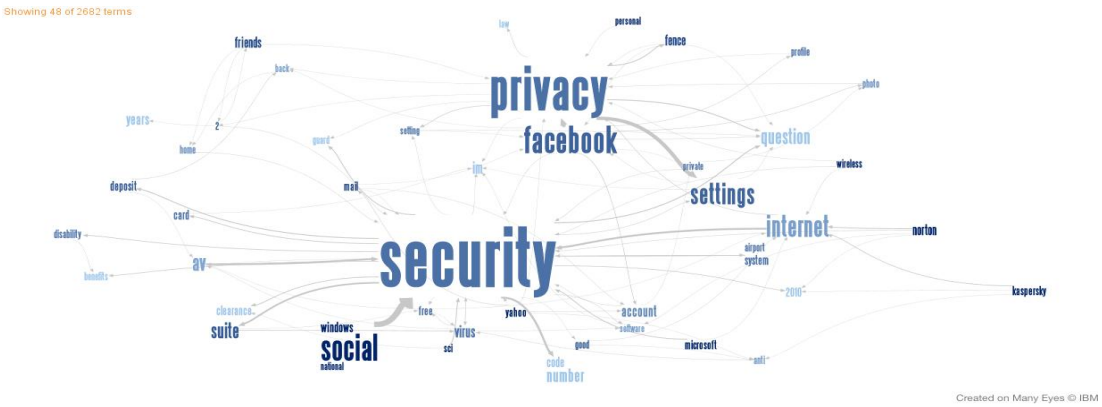


Figure 7. Phrase Net Visualization

This work was supported by the TRUST Center (NSF award number CCF-0424422)

III. Quantitative Results

The various keywords used for collecting data from Yahoo! Answers each retrieved different quantities of questions from the site, which were then stored in the group database. The contribution of the results of each keyword to the total data collected is illustrated in figure 8, which also demonstrates which keywords were most efficient in obtaining results from querying Yahoo! Answers within the specified categories.

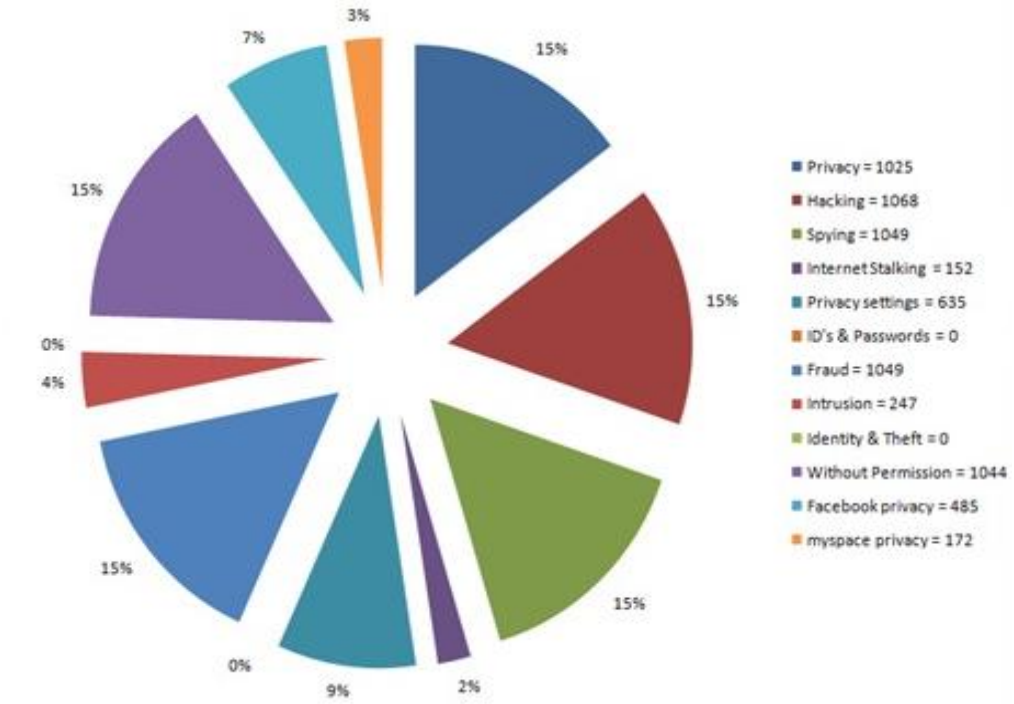


Figure 8. Pie chart shows the percentage of keywords occurrence in the result.

Conclusion

This research incorporated both quantitative and qualitative approaches of analyses, from which certain trends and characteristics were observed. The quantitative analysis, which provided a count of how many questions were added to the database for each keyword queried, revealed that certain keywords were more efficient at gathering data. In addition, the qualitative analysis demonstrated relationships between terms within the question and subject fields that may not have been evident from a strictly quantitative approach. The most useful way to take advantage of the visualizations was to begin with the tag cloud and word cloud generator to find the most prevalent terms. These terms were then searched for within the word tree visualization to find other related terms and phrases frequently used within the text.

These analyses are imperative to future work. In the larger scheme of the project, the trends will be used to further refine and add to the taxonomy of privacy terms, which will in turn be used to produce more data to then be analyzed.

This work was supported by the TRUST Center (NSF award number CCF-0424422)

Acknowledgments

We would like to thank our graduate student mentors, Nick Doty and Jen King, as well as our faculty mentor, Professor Deirdre Mulligan. We would also like to thank Dr. Kristen Gates, TRUST (The Team for Research in Ubiquitous Secure Technology), the NSF and UC Berkeley for giving us the opportunity to conduct this research.

References

[1]Database MySQL. http://www.mysql.com/why-mysql
[2]IBM Many Eyes. http://manyeyes.alphaworks.ibm.com/manyeyes/ 2010
[3]Post Gazette. http://www.post-gazette.com/pg/06235/715694-96.stm
[4]Signonsandiego. http://recall.signonsandiego.com/uniontrib/20060713/news_lz1n13yahoo.html
[5]Yahoo! Answers.
http://answers.yahoo.com/info/about;_ylt=AoFnLeph3VzSztmTXFXm6wTj1KIX

This work was supported by the TRUST Center (NSF award number CCF-0424422)