De-anonymizing Social Network Neighborhoods Using Auxiliary and Semantic Information

---

A Master's Thesis

Presented to

Department of Computer and Information Science

State University of New York Polytechnic Institute

At Utica/Rome

In Partial Fulfillment

of the Requirements for the

Master of Science Degree

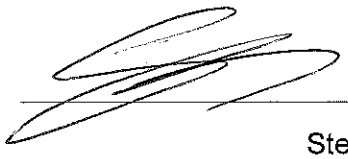
By

Steven Michael Morgan

December 2015

De-anonymizing Social Network Neighborhoods Using Auxiliary and Semantic Information

Except where reference is made to the work of others, the work described here is my own or was done in collaboration with my advisor and/or the members of the advisory committee. Further, the content of this work is truthful in regards to my own work and the portrayal of other's work. This work, I claim, does not include proprietary or classified information.



12/11/15

Steven Morgan Date

SUNY POLYTECHNIC INSTITUTE
AT UTICA/ROME

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

CERTIFICATE OF APPROVAL

Approved and recommended for acceptance as
a thesis in partial fulfillment of the requirements
for the degree of Master of Science in
Computer and Information Science.

December 11, 2015

Date

Jorge Novillo
Advisor

Bruno Andriamanalimanana

Bruno Andriamanalimanana

Michael Reale

Contents

Table of figures

Acknowledgements I would like to thank the State University of New York Polytechnic Institute for the opportunities that have been presented to me thru out my degree program. The classes I have taken provided me with a broad background in the field of Computer and Information Sciences that has already allowed me to explore new paths in my career. A special thanks to Dr. Jorge Novillo, who as an adviser directed me while moving through my coursework and guided me towards an interesting research topic within the field.

I would also like to thank my friends and family for their support during the project. Many of whom were generous enough to give their time to help me gather data. All the support I have received, whether indirect or direct, has allowed me to develop a successful thesis.

Abstract - **The increasing popularity of social networks and their progressively more robust uses provides an interesting intersection of data. Social graphs have been rigorously studied for de-anonymization. Users of social networks will provide feedback to pages of interest and will create a vibrant profile. In addition to user interests, textual analysis provides another feature set for users. The user profile can be viewed as a classical relational dataset in conjunction with graph data. This paper uses semantic information to improve the accuracy of de-anonymizing social network data.**

Introduction

Marketers and stewards of open data continually push to have more information released to improve the utility of using data collections available online. With increased availability of data related to individuals, there is an increased concern in the protection of privacy, both for individuals and organizations responsible for creating the data. In order to protect information, data sources are perturbed or obfuscated to improve their privacy [7]. However in the case of graphs, there are specific challenges related to online social networks, how data collection can be performed, and protecting the integrity of the graph itself. Even in cases where data has been perturbed in an attempt to protect privacy, data has been de-anonymized [8, 9, 10, 13].

Relational data released for the Netflix Challenge was successfully combined with data from IMDB causing private information, including political views to be exposed [13]. This was the first well known linkage attack between two datasets to prove that seemingly unrelated data could be used to expose potentially sensitive data about individual users. In the 2011 Kaggle.com challenge, anonymized Flickr data was successfully combined with a validated set of the Flickr graph based on a seeding step, which the adversary was able to positively identify nodes in the graph based on known auxiliary information [9]. Narayanan and Shmatikov argue that even though the data is anonymized, meaning that personally identifiable information has been removed, they're able to de-anonymize the data on a large scale. In turn, they successfully de-anonymized Twitter and Flickr social networks with a 12% error rate [10]. The described attack using Twitter and Flickr social networks was based on auxiliary information from the graph. Auxiliary information in a graph relates as cited in [10] to measures such as the diameter, edge weights, centricity, etc. related to a specified graph.

Measures have been taken to develop techniques that allow proprietors of social networks to perturb data. Zhou, Bin, and Pei developed an algorithm to perturb a graph data for a particular neighborhood and still maintain the integrity of the graph [8]. In this paper I will demonstrate the feasibility and difficulties of using semantic data, combined with social network graph data, to de-anonymize a social network neighborhood by combining Facebook and Reddit user information. In the feasibility experiment run, it is demonstrated that 100% of users can be identified by traversing thru a neighborhood network of a graph

and successively identifying users. The problem explored in this paper provides increased awareness of privacy concerns related to social networks, and how easily information about individuals may be obtained.

Problem Statement

Users of social networks actively create a broader profile of characteristics with their usage patterns. A user on Facebook can easily see their own usage statistics, which may be tracked by any app that the user subscribes to thru Facebook. This is far from limited to technical measures though. Post frequency, events a user joins publicly, political views shared, friend relationships, and interests all combined with a user's writing style create a feature set in the context of machine learning. Given a high enough dimensionality, all users of a system will present themselves as unique [3]. Understanding that with a high enough dimensionality that any user is unique in one system, it is theoretically feasible that a user on one social network is identifiable on another.

User information is not always publicly viewable in Facebook, as there are available settings within Facebook that allow for a user tailor the information that is viewable to the public and to their “Friends”. A friend on Facebook is simply another user that has mutually acknowledged that they are willing to share their profile information. User profile information can also be obtained by Facebook apps. Facebook does offer a variety of apps, but they have a rigorous process involved with what information is allowed to be used by a specific application that must be acknowledged by the user. This will be discussed in further detail in the threat model. However, not every social media site has the same requirements for revealing information about its users.

Reddit has become arguably the most popular website for users to share news, common interest articles, and solicit feedback from a special interests community. Similar to Facebook, the user community is mostly responsible for building the site content. The online social news website has been at the center of controversy for its stance on user provided content [22]. Unlike Facebook, user information is very easily accessible in mass. User posts are easily searchable. Posts to subreddits, which are in essence communities of interest, are easily found simply by searching for a user id on Reddit. The difference between a Reddit profile and Facebook is that the Reddit profile is considered to be pseudonymous. Pseudo-anonymity in this paper is characterized by any user having only a username. No other personally identifiable information is explicitly required to create a profile. Personally identifiable information is any data that would make a user uniquely identifiable, such as an address or telephone number.

The usage of social networks continue to grow. According to Pew Research, 90% of Americans between ages 18-29 actively use social networks, and other age groups continue to adopt social networks for communication and sharing news. Figure 1 demonstrates the growth by each age group in the United States [21].

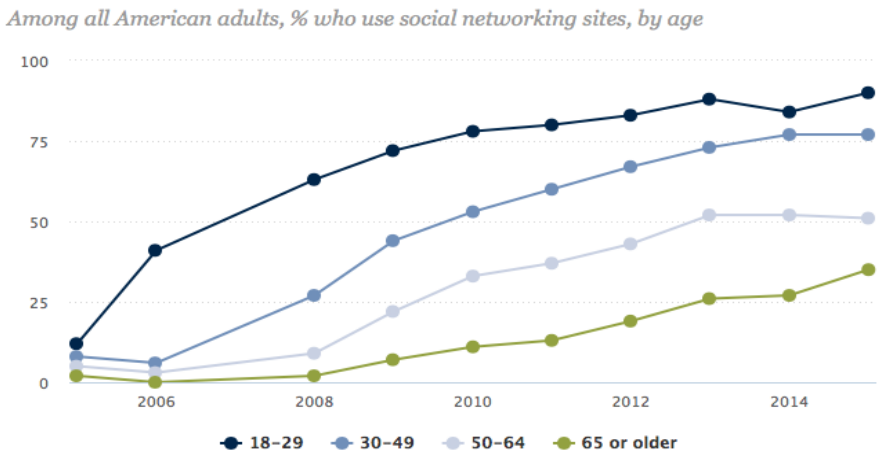


Figure 1 - Percentage of American using social networks by age group [21]

With the increased usage of social networks and the features that are created by users in each community, it is theoretically feasible to combine the datasets of two separate networks in order to de-anonymize users. With knowledge that there are communities online that engage in ethically questionable and occasionally even illegal activity, some users may have a lot at stake in terms of reputation or even legal penalties if de-anonymized. This type of attack was first popularized by Narayanan, Arvind, and Shmatikov [13] in their seminal work to combine 500,000 records of Netflix data that was released as part of the Netflix prize, with records from the International Movie Database (IMDb). The authors of [13] claim that they were able to accurately expose the political views of reviewers and other potentially sensitive information.

Considering the large volume of data compiled by Facebook and Reddit, this paper will demonstrate that there is feasibility within a subset of Facebook and Reddit users to expose their actual identities as given thru Facebook. The subset of users will be referred to as a network neighborhood as in [8]. Graph information will be used, in conjunction with user communities of interests ("likes" on Facebook, subreddits on Reddit), and additionally will include a stylometric analysis component for each user. The specific contributions will be in validating current threat models based on the actual usage scenarios of two active social networks. Analysis will include using semantic data related to user nodes to identify users, but not inclusive of any personally identifiable information. Utilizing active social networks will differ from many previous works were the data was obtained from a social network website for a specific time frame. This will be done in conjunction with analyzing users writing styles on social networks in order to provide a more robust feature set for de-anonymization. Performing stylometric analysis of social network data has been performed previous [25], but in this work it will be used to enhance the feature set of users. Using the described feature set will additional prove that de-anonymization is feasible within the current privacy constraints related to releasing data from Facebook and Reddit.

Literary Review

Background

Privacy concerns related to data mining has been studied extensively. The internet has allowed data to be captured and subsequently released, both illegally and legally. Recent hacks of the government as well as the online cheating website Ashley Madison has greatly increased the public's awareness of the potential for a loss in privacy thru data leaks [27]. Langheinrich [2] provides a review of the issues surrounding ubiquitous computing systems and the privacy issues implied by them. The major legal changes in privacy in relation to information privacy came with the passing of the US Privacy Act of 1974 and secondly when the EU Directive 95/46/EC of 1995. These acts of legislation provided principals to follow when considering information privacy and what the best practices should follow. The seven principles are: Openness and transparency, individual participation, collection limitation, data quality, use limitation, reasonable security, and accountability. The European Union directive requires that the organization requesting the data to clearly provide intent and obtain consent in order to collect data from an individual. Highlighting these principles will give a basis to consider throughout when considering the implications of the linkage attacks performed and the means by which the data was obtained.

Although the work of Langheinrich is now dated, it does provide an ominous warning in summary stating that "ubiquitous computing is easy to imagine: the frightening vision of an Orwellian nightmare-come-true…" [2]. There is some irony to the statement, as many popular technologies now adopted can provide virtually constant surveillance, namely cell phones. Although, it is entirely up to the end user to decide how they wish to use the technology available to them, public awareness of their own monitoring main not entirely be based on actual capabilities.

Defining and protecting privacy

Differential privacy provides a definition of the probability that an individual's privacy could be compromised when two datasets differ by at most one element.

Definition 1. A randomized function K gives ε-differential privacy if for all data sets D1 and D2 differing on at most one element, and all $S \subseteq \text{Range}(K)$,

$$\Pr[K(D1) \in S] \leq \exp(\varepsilon) \times \Pr[K(D2) \in S]$$ [6]

Dwork's summarization of different methods employed to ensure that privacy would be less likely to be compromised have provided a basis for much of the work in the area of information privacy and has been motivation for countless papers in the field.

Auxiliary information in the context of Dwork's work [1, 6] refers to any information provided that is not directly related to a database. An adversary may be aware of auxiliary information related to a database then is able to cause a data breach. There is no debate whether or not value can be found from being able to query aggregate information from a

dataset, but no individual information should be compromised in the process. Dwork provides generalized theoretic strategies obfuscation and perturbation, that an employer of data would be able to utilize to prevent a loss of privacy. An adversary with auxiliary knowledge and bounded by reasonable computational power has a probability to discover private information. Differential privacy gives a means to measure if the release mechanism used on datasets can be protected with negligible probability. Dwork also proves that L1-sensitivity provides a formula for giving the amount of random noise required to protect the privacy of two individuals in a database [6]. Measuring sensitivity gives a means to determine how much integrity of the data may be lost in order to protect privacy.

There is continued work to provide a basis for a statistical bounds on how much perturbation is required on a dataset, and how much privacy is gained by an individual within these bounds. This is accomplished by setting infimum and supremum values on a dataset. There is a tradeoff described in the work by Duchi, Jordan and Wainwright [3]. The more privacy that is achieved by perturbing a dataset, the less utility is gained. This can colloquially described as an individual not wishing to share information about past addiction, but if the individual provides the information, it could be the information required to make a connection between the patient, the ailment they are describing, and the most effective treatment. The probabilistic framework developed gives models that can accurately describe the amount of perturbation that is required to maintain privacy. The models developed are based on information theory. This in contrast to other works completed that mostly focus on machine learning techniques used to discover the information gained by combining datasets.

Chaudhuri and Monteleoni consider the balance between what may be learned from a dataset using a logistic regression machine learning algorithm and the privacy that can be preserved [4]. Differential privacy is considered by Chaudhuri and Monteleoni as the formal definition of privacy and is used in the proofs for the theorems they are using to support their privacy preserving logistic regression algorithms. Claims made in the paper authored by Chaudhuri and Monteleoni are based on the assumption that an adversary is able to make queries to a dataset, and result are returned thru a sanitization mechanism, a common framework used in proving differential privacy. Two different algorithms using logistic regression are compared. One algorithm adds noise to the classifier and requires a regularization parameter, while the other, which uses a regularization constant, provides better guarantees on performance as well. Machine learning algorithms are normally sensitive to perturbation, but using a regularization constant rather than a parameter allows for sensitivity in relation to perturbation to be minimized. The results of the comparison of the two algorithms do demonstrate a 20% performance advantage for the algorithm based on a regularization constant for machine learning problems that are applicable to logistic regression matching [4].

Many models have been studied with consideration of how individual privacy can be protected. Decision tree classification was tested by Agrawal and Srikant to see how accurately it could identify records based on the amount of data that was perturbed. The authors administered a survey with the intent of using it for classification and information privacy. They allowed people to say if they would allow for certain values related to themselves to be used to in the dataset to be studied by their algorithm. The dataset studied

was used to classify if individuals are high or low risk for a loan. Agrawal and Srikant developed two different methods for distorting the data (uniform and Gaussian distortion), and three different algorithms to reconstruct the dataset; global, byclass, and local. These classes are simply different levels of granularity within the datasets to form a partition. As expected, the classification algorithm performs best on undisturbed data generally, but two of the functions perform within 5% of the accuracy of the original dataset, even with 100% accuracy. Interestingly as well, these are also probably the simplest classifiers developed [5].

Zhou and Pei developed a model for preserving privacy against network neighborhood attacks on an online social site. Much of the motivation for their paper comes from social sites releasing data in mass. Releasing data in mass for study is performed more carefully now than in recent years, probably in part thanks to works described here. Many sites do offer APIs that allow user data to be pulled from the site, which is a relevant issue today. The idea of preserving privacy in a social neighborhood is quite relevant. The type of attack described by Zhou and Pei occurs when there are neighbors of a node that only knows of the existence of other neighboring nodes. Based on the knowledge of the existing nodes, an adversary is able to obtain private information about the neighbors, without explicit knowledge of their identity. The definition of privacy used by Zhou and Pei is k-anonymity.

Definition 2. K-anonymity is a measure of privacy given as 1/k, for a user specified value k. For a dataset to be adequately protect security, an adversary cannot identify an individual with a probability of 1/k [8, 12].

Based on a reduction from k-dimensional perfect matching, k-anonymity is said to be NP-hard.

In order to simplify an attack against a neighborhood, Zhou and Pei define a neighborhood thru a lexical encoding that represents the edges of a graph. This encoding allows them to generate a minimum spanning depth first search tree to compare if neighborhoods are identical. Two neighborhoods are said to be equal if their minimum spanning trees are lexically equal, which is then equivalent to the graphs being isomorphic.

A method based on seeding from the unique nodes with high degree is used to determine where dummy edges may be placed, by placing dummy edges on the most unique nodes, it then makes seeding for an adversary more difficult to match the nodes. This method of perturbation allows for the overall integrity of the graph to maintained, while still protecting the privacy of the individuals [8].

De-anonymizing data

Narayan and Shmatikov claim only a 12% error rate when identifying anonymized Twitter and Flickr data despite the fact that there only exists a 15% overlap of nodes [10]. Their work is unique in that they avoid using auxiliary information and only identify nodes based on the structure of the graph. They also address other methods that have been used that do not scale as well as their methods, most notably, a reasonability argument for why

sybil nodes are not practical at scale. Sybil nodes are intended to create a unique node in a network that can later be identified by an adversary, but in order for this to be effective, an adversary would likely need thousands of nodes to identify considering that popular social networks have millions of nodes. Many social media sites do have requirements that make it difficult for an individual to have multiple accounts.

The basis for their linkage attack does consider that correct identification of a node provides additional information that can then be used to correctly identify other nodes. This correct identification could then be used to obtain additional auxiliary information. Seed identification is crucial to the algorithm used by Narayan and Shmatikov in [10]. The authors compare it to spreading a health epidemic, if too few seeds are identified, then the epidemic never gains traction. Identifying seeds with high node degree allows the algorithm then to identify unique cliques within the graph. Further demonstrating the success of their algorithm, the success of correct identification only needs to meet a specified threshold, in this case it was 10,000 seeds.

Narayan, Shi and Rubinstein won the 2011 Kaggle.com competition with the context winning area under curve. They used a simulated annealing technique based on seeding from the cosine similarity of inward edges. Seeds were identified by finding cliques originating from the highest connected nodes. From the seeds the algorithm propagated thru the network. Initially the contestants did not know the network which they were attempting to de-anonymize, but thru the course of the contest they were told the actual network, which gave them a way of validating their findings. Narayan, Shi and Rubinstein contend that de-anonymizing a graph is a global optimization problem, over a graph matching problem, which is believed to be NP-complete, thru a generalization to graph isomorphism.

A voting system was employed for link prediction to determine who would be the likely neighbors. This was done by reviewing possible edges and voting if they were sufficiently similar to the valid dataset. This gave the criteria for developing a local optima for the simulated annealing algorithm. Other methods tested were random walks, and random forests, but the simulated annealing algorithm proved to be the most accurate [9].

Brennan, Afroz and Greenstadt have done comprehensive work on stylometric analysis.

Stylometry has been used in legal cases and has been an area of open study related to authorship attribution. Brennan, Afroz and Greenstadt test to see if machine learning techniques can be applied to correctly identify authors, even when the author is attempting to obfuscate their writing style, or trying to imitate someone else. They developed a corpus of at least 5000 words and tested three different machine learning algorithms, each with its own respective feature set. The accuracy of the machine learning approaches does degrade with increased authors, but always performs better than random, even with up to 40 authors. The work by Brennan, Afroz and Greenstadt was a primary motivation for the analysis performed in this paper [14].

Threat Model

As discussed, it is feasible to de-anonymize graph networks when there is an overlap between two networks by finding common nodes, or in the case of the Kaggle.com contest, by validating the anonymous network against a de-anonymized network [9, 10]. Greenstadt et al. have also demonstrated that using stylometric analysis provides a means to accurately identify authors, even given the case that the author may not be deliberately using their natural writing style [14]. It is also feasible to use a minimum spanning tree to compare if two networks are the same by lexically encoding them for comparison [8]. In this paper, an adversary will be considered that has knowledge of a neighborhood and has semantic information regarding the users of a mostly public neighborhood, in order to de-anonymize the pseudonymous users of another neighborhood on a separate social networking site.

The adversary will have the ability to create their own seed node. This node can then be used to gather information about the individuals within the neighborhood. The adversary will also have partial knowledge of the pseudonymous users on the second social network. Partial knowledge being that the user is able to see the activity of any pseudonymous user, but not obtain any personally identifiable information about the user based on their profile. The goal of the adversary is to be able to de-anonymize the users on the pseudonymous social network, in order to gain knowledge of information that otherwise may be private.

In this particular simulation, information has been gathered from Facebook by users that were willing to volunteer their information for the study. Many of the same users also volunteered their Reddit usernames. Facebook provides the social network neighborhood data that will be considered to be public. Reddit user information will be the forum considered for pseudonymous users.

Methods Used

Data Collection

The ability to pull information in mass from social websites using APIs provided by social networking websites is common practice. Facebook has a large catalog of applications that are built on top of the social network that are intended to enhance the overall user experience. The apps that are developed for Facebook have the ability to pull user information, but only with explicit consent from users. There are exceptions to fields that an application can pull without user consent, but these fields are generally information that can be viewed publically regardless of individual settings, IE: Facebook display name.

A secure token exchange must take place between the Facebook app and the user in order for the application to pull user information. The token exchange is valid once a Facebook user validates themselves using their Facebook credentials. For the duration of the user session, the application is able to query information about the user. All fields that an application is able to pull beyond the default settings given by Facebook, must pass their internal review process. In other words, Facebook applications must prove that pulling specific information is providing value to the user experience, and is not strictly for the purpose of data mining [23].

Due to the amount of rigor required to get a Facebook application it was not feasible to create an app and have it accepted by the Facebook application store, particularly for academic purposes. There is the ability to create test users for an app, but it would have required more development and involvement from individuals who volunteered to participate. At this point, it was clear the most appropriate approach would be to manually pull the information that would be relevant for this study. A page scraper was also considered, but given the relatively small dataset, it was not deemed to be necessary. Most Facebook users do not type long comments, at least the subjects in this experiment. Only a few hundred words per user was actually obtained.

Pulling data from Reddit is much simpler than from Facebook. The Reddit API has been named PRAW (Python Reddit API Wrapper) [26]. It allows for user information to be pulled by a Reddit username. Virtually any information can be pulled, in the particular case, using only a few lines of code (See figure 2), data was successfully pulled data for all Reddit usernames obtained. Some Reddit users had thousands of words, while others would only post short statements, comparable to the manner Facebook users seemingly most often phrase Facebook posts. In the code sample below, 'redditor' refers to a Reddit username.

```
>>> Import praw
>>> r = praw.Reddit('Comment getter 1.0 by /u/morgansm')
>>> user = r.get_redditor('redditor')
>>> for comment in user.get_comments(limit=None)
…       print comment.body
```

Figure 2: PRAW code for pulling user comments

Validating Feasibility

Including additional semantic information beyond textual analysis did prove to be beneficial, particularly for the graph data. Graph measures such as high node degree can make an individual node much more identifiable, if not entirely unique and has proven to be useful in prior works [9, 10]. Additionally including interests the feature set makes the overall feature set more robust and increased the likelihood that an individual may be identified.

Definition 2: A graph is set of nodes and vertices defined as a set of ordered pairs. $G = (V, E)$ where V represents a vertices and E represents an edge between two vertices.

Definition 3: A user of a social network *S* consists of a graph $G = (V, E)$ where V represents the set of users as a nodes ($u = \{u_1, u_2, \ldots, u_n \mid u_x \epsilon V\}$), and E represents an edge ($e = \{e_1, e_2, \ldots e_n \mid e_x \epsilon E\}$), indicative of a friend relationship. Edges in the social network graph G are undirected.

Definition 4: Semantic information related to a vertex (or node) that is not critical to the structure of the graph, but relevant to the social network dataset (e.g. User A likes

something is true). The dataset is defined to be $S = \{s_1, s_2, ..., s_n \mid s \in S\}$ such that for every user u there exists a set S.

Definition 5: Auxiliary Information measures such as the diameter, edge weights, centricity, etc. related to a specified graph. This information may be stored in the same manner as semantic information, for every user.

In order to validate that there was feasibility to identify users, a test was run with Naïve Bayes analysis. The overall dataset could be validated because the users are known in the test set. A feature set for each node was developed in both graphs. The feature set consists of semantic and auxiliary information. Being able to develop a feature set for the social network data required coding the graph in such a way that it could be handled like a relational dataset. Once the data was converted to a format that allowed it to be usable by the Naïve Bayes classification algorithm, continued testing was simplified.

Definition 6: The user feature set, f contains edges $e_{1...n}$ of social network graph G, and semantic information set S, related to a node u_x. $F_x = S_x || E_x$ where || denotes concatenation of the sets for a given user. For a given user u_x, that belongs to a social network, user u_x has multiple features f, making up a feature set defined as $F = \{f_1, f_2, \ldots, f_n \mid f_x \in F\}$. A feature set for a user will be stored as a vector v, which will be referred to as the user feature set.
$v_x = u_x || F_x$

Definition 7: A social network dataset D consists of user vectors $v_{1...n}$ which contains a user identity u_x, and a feature set F_x.
$D = \{ v_1, v_2, ...v_n \mid v_x \in D \}$

The feature set F_x consisted of all known friends in the feature set, all textual analysis, and the interests in each social network. De-anonymization took place by comparing the overall training set of features from Facebook and compared with all features obtained from Reddit data. Algorithm 1 found the most likely features that could identify an individual. This was done by providing a likelihood based on each feature identifying a particular user. In the sense of this test, there was no true de-anonymization, strictly a validation that users could potentially be identified. Algorithm 1 is the summation of definition 10 for the correctly identified users divided by the total user set.

Definition 8: For a given user $v_x = \{u_x, F_x\}$ and datasets D *and* D' *where* $D \neq D'$, there exists a function where a user can be identified. This function will be referred to as $\mathbf{de-anon}\ (v_x, D')$ = 1. If the function $\mathbf{de-anon}$ evaluates to 1 then it has correctly identified a common user between each dataset. The function $\mathbf{de-anon}$ is defined as follows:

max(v, v'):
$f_{1...n} \in F,$
$u \in v$

$u' \epsilon v'$
$\forall f'_{1...n} \epsilon F'$
choose max u' *for* Pr[u|f'] = Pr[v] / Pr[f']

Definition 9: The definition for maximum probability is from [19], as described for Naïve Bayes classification. The highest likely probability that a feature set determines a label is defined as the label, feature set combination divided by the probability of the feature set.

$\mathbf{de-anon}(\boldsymbol{v_x}, \boldsymbol{D'}) = 1$ *iff:*
$F_x \epsilon \boldsymbol{v_x}$
$u_x \epsilon \boldsymbol{v_x}$
$\forall \boldsymbol{v_n}' \epsilon D'$
$u_x = max(\boldsymbol{v_x} = \boldsymbol{v_n}')$
and 0 otherwise

Algorithm 1: An adversary is successful if it is able to successfully identify a subset with an acceptable user defined probability, *p*. The definition of k-anonymity, this value could be defined as 1/k, where k is the number of nodes in a graph. A function ***A*** to define the adversary is as follows:

A(D, D'): $\forall v_n \epsilon D$
$\sum_o^n \mathbf{de-anon}(\boldsymbol{v_n}, \boldsymbol{D'}) = 1 / \sum_o^x \mathbf{de-anon}(\boldsymbol{v_x}, \boldsymbol{D'}) = 0 + \sum_o^n \mathbf{de-anon}(\boldsymbol{v_n}, \boldsymbol{D'}) = 1$

Definition 10: The adversary is said to be successful if ***A***(D, D') > p.

Several tests were run, revealing that there was potential to successfully de-anonymize users based on the features acquired. Experimentation was performed with varying values of users in the test and training sets. Interestingly, the percentage of users correctly identified is more effective generally when the training set is smaller. It is expected that this is in part due to there being less available labels for each feature set.

Test Set Size	Train Set Size	Accuracy
24	6	50%
23	7	57%
22	8	50%
21	9	44%
20	10	10%
17	13	40%

15	15	47%
14	8	50%
13	9	56%
12	10	70%
11	11	72%

Figure 3: Validation results

The training set size also shows that there was some ability to over train the machine learning model. A training set of size six and eight yielded the same percentage of correctly identified users. Both tests with training sets of size six and eight users performed comparably well to when the overall dataset is split equally between training and test data. The most effective test yielded 72% when the training set and test sets were of equal size, but smaller than the overall data set. Knowing that the users contain features detailed enough to yield results that would define a breach in privacy by k-anonymity [8, 12]. The results of these tests lead to the intuition that there are users in the data set that are easy to identify no matter the size of the training data. This proved to be very valuable in de-anonymizing the overall dataset.

Identifying users by Textual Analysis

The initial attempt to identify users was based on K-Means clustering. A feature set was developed strictly from the text that users provided in an attempt to identify pseudonymous Reddit users. The feature set used was made up initially by creating a bag of words using the available libraries in the natural language toolkit available in Python [17, 20, 14]. Despite there being intuitive identifiers between certain users that would use both Facebook and Reddit to discuss certain interests, the bag of words technique did not prove to have any effectiveness using k-mean clustering. These results are likely due to the fact that the bag of words approach created a very high dimensionality feature set for each user, but even the reoccurrence of certain terms was not enough to conclusively identify an individual. Bag of words analysis was completed using the NLTK library [20].

The second feature set developed using textual analysis was made up of lexical and semantic evaluations of writing style. For each user the following feature set was developed:

Average Length of Words
Average Sentence Length
Frequency of commas
Frequency of semi-colons
Frequency of colons
Frequency of parentheses
Frequency of periods
Positive Sentiment
Negative Sentiment

Sentiment analysis was performed using the Blob library [24]. Sentiment analysis uses a Naïve Bayes classification to decide if the sentiment of a sentence is positive or negative, the overall sentiment of a user was given an aggregate score for both negative and positive sentiment. Despite some known users having very similar sentiments analysis scores, it also did not prove to be valuable in identifying pseudonymous users.

Frequency counts were used for lexical features as an indication of writing style. Colons and parentheses were included because they were used as substitutes for emoticons that are commonly used on social media. NLTK did not effectively handle UTF-8 encoded characters, and thus had to be reduced to ASCII characters in order for it to be effective. Although the textual analysis alone did not provide any meaningful results, it did provide some interesting metrics on the writing styles between the users on each social media site. These values could be helpful in understanding why textual analysis was not valuable in this experiment, and how it could be altered in the future to improve the results.

Averaged values	**Facebook**	**Reddit**
Words per sentence	2.29797	16.81192
Word length	10.96455	3.25127
Positive Sentiment	2.903	0.95
Punctuation per sentence	0.9859	0.285

Figure 4: Stylometric measurements on Facebook and Reddit text

Sentences were tokenized using the tokenizer provided with the NLTK library. Not all phrases are in the form of complete sentences, and some may not have any punctuation at all. The NLTK tokenizer offered a tested means to quickly break the mass text into tokens. Considering that not all sentences are equal could also account for some of the disparity in the metrics between the social media platform.

The differences in these text metrics can be partially explained by a typical post on Facebook being short. A post typically is of the form: “Thanks everyone!” or “I can’t wait for the wedding festivities!”. Reddit comments are generally complete sentences replying to a specific subject. This also explains the overall sentiment being difficult to correlate, common Facebook comments are short and frequently congratulatory or celebratory. Reddit comments, at least in the observed data set, tend to be more thoughtful and complete opinions.

De-anonymization of the Reddit data set

The work of [9] demonstrated that seeding could be used as a step in de-anonymizing a graph. The seeding technique used identified unique nodes, and once they were identified the graph could continue to be traversed. In this paper, the technique used was to effectively use two sybil nodes that could be used for seeding in the initial step. In addition to using the sybil nodes for initial step, nodes were sorted by their amount of outgoing edges. In this experiment, this is the auxiliary information obtained from the graph. Additional metrics did not prove to be useful in identifying users, particularly because the

graph is small, consisting only of 14 nodes. One node was discarded because it lacked a connection to the neighborhood.

The graph was successfully traversed by first selecting nodes with one edge, then the nodes with two edges, and so on. This technique proved to be highly effective. If not all nodes with a specified edge count were successfully identified, then the test would be run again for a given edge count, removing the nodes that were already successfully identified. The edge counts for a node was used to combine nodes between the two graphs, but not all equivalent nodes shared the same amount edges. This was not a particular concern because enough nodes shared equivalent edge counts that a large percentage was identified by this comparison. In the case that individuals had separate node counts, these nodes were handled after all other edge counts had been tested.

As demonstrated in the feasibility test, the smaller the dataset the Naïve Bayes classifier had to choose from, the better likelihood that it would be able to accurately identify a user. This proves true by the step required to trim the data set after effectively identify some subset of users with a specified edge count. The results were not as effective when attempting to classify a larger set of users.

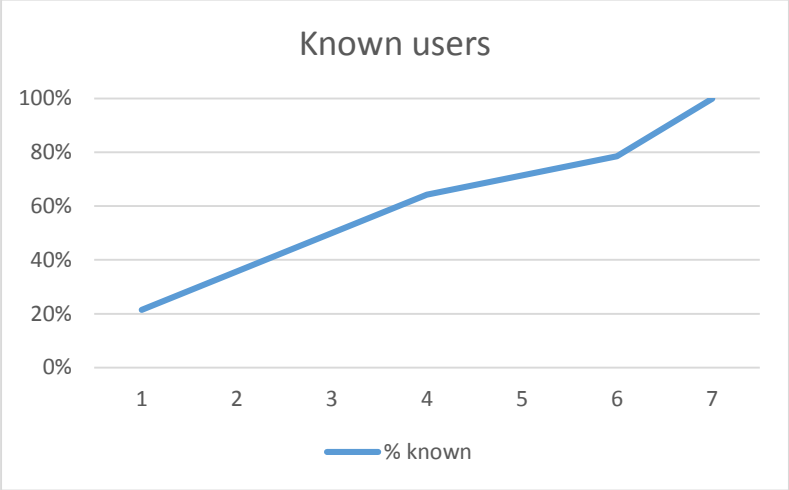


Figure 5: Percentage of Users Identified by Run

Algorithm 1 was run, the same as in the validation set, but this time the user information was kept pseudonymous for Reddit users. In each iteration a subset with the same edge count would be processed to limit the amount of users attempting to be processed. Users with the same edge count in both Facebook and Reddit datasets would be run for comparison. Not all users had the same edge count in both networks. This did not prove to be an issue for the classification algorithm.

After the sixth iteration, nodes were no longer able to be identified with the just textual analysis and user interests. Textual analysis in this step was carried out by find the twenty most common words for a specified user, and then each word could be used as a feature. IE: “John’s text contains bike = True”. Reducing the training set and the test set made the overall feature set smaller. The algorithm when classifying users used the top twenty words for every user, when the feature set was reduced by reducing the amount of

users being evaluated by the classifier, it was more effective. When the training set contained all users, it would results in an apparently unwieldly feature set for the classifier. This was demonstrated by the textual analysis not showing any value in the validation step of the experiment.

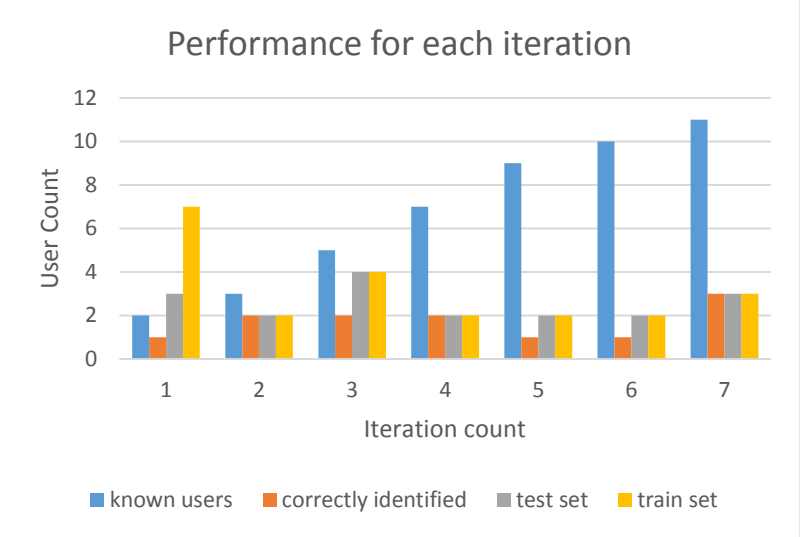


Figure 6: Test Results on Identifying Reddit Users

The final step of the algorithm proved to be completely effective in identifying the remaining users. At this point, the identity of the previously anonymous Reddit users could be added back into the training set. The classifier effectively had a third feature set that could uniquely identify users. In essence, even though the algorithm did not know the identity of a particular user, the classifier was now able to identify who a given user's friends are.

The chart for performance of each iteration demonstrates how each successive run of the de-anonymization algorithm progresses thru the anonymized users. At the first run the algorithm was only aware of the sybil nodes, but using these nodes made it possible for the algorithm to identify nodes that were only connected to the graph with one node. The chart showing the increase of the amount of users successfully identified shows that the algorithm consistently performed with each iteration, identifying at least one user. Similar to the simulated annealing technique used in [9], this method of using Naïve Bayes classification requires that if the graph is unable to further identify users that it would move on to the next set of nodes. This is comparable to a cooling phase. In this particular example, there is no digression to find the local optimum. The algorithm begins processing the next batch once no other nodes can be identified. If there are nodes that are not successfully identified, those users were stored for the final identification step.

The results successfully demonstrate at the validation step and at the de-anonymization test that it is possible to achieve a privacy breach of the provided data sets. Using definition 2, even successfully identifying two nodes in the initial step warrants a breach of privacy. Considering definition 1, differential privacy, it has been proven that information about a particular user can be gained based on the two distinct data sets.

Discussion

There are several areas that proved to be more difficult than initially estimated. In analyzing text, using measures such as words per sentence and punctuation would be more useful with a larger volume of text for every user. Users from Facebook typically frequently had only a few hundred words. In comparison to [14], a minimum of 5,000 words was required by each individual included in the corpus. This also proves to limit the effectiveness of using the bag or words technique. Despite there being words that certain users frequently used intuitively, specific counts did not warrant any features that would make them uniquely identifiable.

Collecting user interests also proved to be difficult for many users. Being limited to individuals willing to volunteer their profiles did not yield robust profiles in every scenario. This does reflect a real world issue, and gives some credibility that performing de-anonymization at scale is mostly dependent on having highly active users, but other studies have been able to de-anonymize graphs solely based on the structure of the graph [9, 10]. The technique used in this paper also provides a basis for how de-anonymizing social networks at scale could be performed. The challenge would be identifying networks large enough that could easily be taken at scale. Data from Reddit is easily available, my recommendation would be to find another social graph to combine with it that guarantees some level of overlap.

The graph information that was used in the paper was limited by the amount of participants as well. Without having the ability to pull information from Facebook using the available APIs, it was difficult to collect large volumes of information that could be used. Gaining the ability to create a Facebook application to demonstrate that information currently stored by applications that are active would provide much more conclusive results. This is a credit to the privacy settings and policies that Facebook has created and adheres to.

Reddit user data is much more easily obtained than Facebook data, but the overlap that was exploitable between the sites is relatively small. There was an expectation beginning the project that it would be much easier to obtain users that are active on both sites, which also proved to be less than initially imagined. At minimum, a basis has been provided for academic research that could be done using the actual access channels in industry. Basing research on data leaks, particularly from social networks no longer seems to be as relevant as it once was. There will continue to be privacy issues surrounding social networks. One could reasonably argue that for many people, Facebook has become their primary means for communications. With continuing growth of data belonging to these networks, it is even conceivable that identity theft could be taken to levels that were not conceivable in the past.

Conclusion

This paper successfully demonstrates that users between Facebook and Reddit can be accurately de-anonymized when considering a specific neighborhood. Using validating techniques it was proven that stylometric analysis can improve the ability of machine learning algorithms to identify pseudonymous users on a separate social network. This was done in conjunction with exploiting the structure of the graph of the underlying network, and using user interests.

There are several areas of further research. Most papers found when doing background research for this paper used data sets that were released by social networks. In searching for released data sets, I was unable to obtain data sets that are readily available that would allow for testing at scale. Data is considered by most companies to be an extremely valuable asset in the current business environment, and is protected by all organizations. There are still ways, as mentioned in this paper, which could further demonstrate an adversary's ability to de-anonymize data. Demonstrating that it is possible to de-anonymize data between social networks has significant financial implications on the organizations that host the data, but also reputation and privacy are at stake for the individual users of social networks.

An additional direction that would be beneficial to future research would be consider a social network with more lax policies than Facebook. There are many other platforms that are used daily around the world that would offer similar datasets. In collecting information for this paper and discussing with friends and family, it is an issue that people care about, and that would be of great interest to anyone concerned about privacy and their digital footprint.

References

[1]C. Dwork and A. Roth, *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science, 2014, pp. 28-64.

[2]M. Langheinrich, 'Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems', *In Proceedings of the 3rd international conference on Ubiquitous Computing (UbiComp '01)*, pp. 273-291, 2011.

[3]J. Duchi, M. Jordan and M. Wainwright, 'Privacy Aware Learning', *Journal of the ACM*, vol. 61, no. 6, pp. 1-57, 2014.

[4]K. Chaudhuri and C. Monteleoni, 'Privacy-preserving logistic regression' In *Advances in Neural Information Processing Systems*, pp. 289-296. 2009.

[5]R. Agrawal and R. Srikant, 'Privacy-preserving data mining', *ACM SIGMOD Record*, vol. 29, no. 2, pp. 439-450, 2000.

[6] Dwork, 'Differential privacy' *Encyclopedia of Cryptography and Security*, pp. 338-340. Springer US, 2011.

[7] Y. de Montjoye, C. Hidalgo, M. Verleysen and V. Blondel, 'Unique in the Crowd: The privacy bounds of human mobility', *Sci. Rep.*, vol. 3, 2013.

[8] Zhou, Bin, and Pei, ‘Preserving privacy in social networks against neighborhood attacks’ *Data Engineering*, 2008. ICDE 2008. IEEE 24th International Conference on, pp. 506-515. IEEE, 2008.

[9] Narayanan, Shi, Rubinstein, ‘Link prediction by de-anonymization: How we won the kaggle social network challenge’ *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 1825-1834. IEEE, 2011.

[10] Narayanan, Arvind, Shmatikov, ‘De-anonymizing social networks’ *Security and Privacy, 2009 30th IEEE Symposium on*, pp. 173-187. IEEE, 2009.

[11] S. Raskhodnikova, 'DP for Graphs and Social Networks | Simons Institute for the Theory of Computing', *Simons.berkeley.edu*, 2015. [Online]. Available: http://simons.berkeley.edu/talks/sofya-raskhodnikova-2013-12-14. [Accessed: 24- Jul- 2015].

[12] L. SWEENEY, 'k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557-570, 2002.

[13] Narayanan, Arvind, Shmatikov, ‘Robust de-anonymization of large sparse datasets’ *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 111-125. IEEE, 2008.

[14]M. Brennan, S. Afroz and R. Greenstadt, 'Adversarial stylometry', *ACM Transactions on Information and System Security*, vol. 15, no. 3, pp. 1-22, 2012.

[15] Ramyaa, Congzhou He, and Khaled Rasheed. ‘Using machine learning techniques for stylometry’ Proceedings of International Conference on Machine Learning. 2004.

[17] S. Bird, E. Klein and E. Loper, Natural language processing with Python. Beijing: O'Reilly, 2009.

[18] G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[19] Aicbt.com, 'Authorship Attribution with Python', 2015. [Online]. Available: http://www.aicbt.com/authorship-attribution/. [Accessed: 5 - Nov- 2015].

[20] Kaggle.com, 'Part 1: For Beginners - Bag of Words - Bag of Words Meets Bags of Popcorn | Kaggle', 2015. [Online]. Available: https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words. [Accessed: 7- Nov- 2015].

[21] A. Perrin, 'Social Media Usage: 2005-2015', Pew Research Center: Internet, Science & Tech, 2015. [Online]. Available: http://www.pewinternet.org/2015/10/08/social-networking-usage-2005-2015/. [Accessed: 12- Nov- 2015].

[22]BBC News, 'Reddit will not ban 'distasteful' content, chief executive says - BBC News', 2012. [Online]. Available: http://www.bbc.com/news/technology-19975375. [Accessed: 14- Nov- 2015].

[23] Facebook Developers, 'Graph API: Overview - Documentation - Facebook for Developers', 2015. [Online]. Available: https://developers.facebook.com/docs/graph-api/overview/. [Accessed: 20- Sep- 2015].

[24] Textblob.readthedocs.org, 'TextBlob: Simplified Text Processing — TextBlob 0.11.0 documentation', 2015. [Online]. Available: https://textblob.readthedocs.org/en/dev/#. [Accessed: 12- Oct- 2015].

[25] C. Peersman, W. Daelemans, and L. Van Vaerenbergh. 'Predicting age and gender in online social networks.' *Proceedings of the 3rd international workshop on Search and mining user-generated contents*. ACM, 2011.

[26] Praw.readthedocs.org, 'PRAW: The Python Reddit API Wrapper — PRAW 3.3.0 documentation', 2015. [Online]. Available: https://praw.readthedocs.org/en/stable/. [Accessed: 10- Oct- 2015].

[27] N. PERLROTH, 'Ashley Madison Chief Steps Down After Data Breach', *Nytimes.com*, 2015. [Online]. Available: http://www.nytimes.com/2015/08/29/technology/ashley-madison-ceo-steps-down-after-data-hack.html?_r=0. [Accessed: 23- Nov- 2015].

Appendix A – Reddit Comment Retrieval

```
import praw
r = praw.Reddit('Comment getter 1.0 by /u/morgansm')
user = r.get_redditor('test')
for comment in user.get_comments(limit=None):
    print comment.subreddit

#Read from a file, output to another
with open("redditors.txt", 'r') as input, open("redditcommentsgrouped.tsv", 'w') as tsv_out:
    for line in input:
        user = r.get_redditor(line)
        tsv_out.write(line)
        tsv_out.write('\t')
        for comment in user.get_comments(limit=50):
            text = str(comment.body).encode('ascii', 'ignore')
            text = re.sub(r'[^\x00-\x7f]',r' ', text)
            tsv_out.write(text)
            tsv_out.write(" ")
        tsv_out.write('\n')

#Test...
with open("redditors.txt", 'r') as input:
    for line in input:
        print line
        user = r.get_redditor(line)
        for comment in user.get_comments(limit=50):
            print comment.author

#For printing original posts
import praw
r = praw.Reddit(user_agent='testing search 1.0 /u/morgansm')
results=r.search('author:test', subreddit=None, sort=None, period=None)
for x in results:
            print x
```

Appendix B – Classification Algorithm

```
#!/usr/bin/env python

import csv
import numpy as np
import nltk
import glob
import os
import re
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
from scipy.cluster.vq import whiten
from itertools import groupby
from sklearn.ensemble import RandomForestClassifier

sentence_tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
word_tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')


def readfile( input ):
        output = []
        with open(input, 'r') as tsvfile:
                for line in csv.reader(tsvfile, dialect="excel-tab"):
                        output.append(line)
        return( output )


def groupfile( input, output ):
        with open(input, 'r') as tsv_in, open(output, 'w') as tsv_out:
                reader = csv.reader(tsv_in, dialect="excel-tab")
                writer = csv.writer(tsv_out, dialect="excel-tab")
```

```
                grouped = groupby(reader, lambda x: x[0])
                for key, group in grouped:
                        rows = list(group)
                        columns = zip(*(r[1:] for r in rows))
                        use_values = [c for c in columns]
                        new_row = [key] + use_values
                        writer.writerow(new_row)

#groupfile( 'redditcomments.tsv', 'redditcommentsgrouped.tsv' )

fbgraph = []
fbgraph = readfile( 'FBGraphRedditOnly.tsv' )
fbgraphsize = len(fbgraph)

fblikes = []
fblikes = readfile( 'FacebookInterests.tsv' )
fblikessize = len(fblikes)
for line in fblikes:
        line[1] = line[1].lower()

#Get the Facebook posts
fbtext = []
with open('FacebookPostsClean.tsv', 'r') as tsvfile:
                for fbUsers, line in enumerate(csv.reader(tsvfile, dialect='excel-tab')):
                        line[1] = re.sub(r'[^\x00-\x7F]+',' ', line[1])
                        fbtext.append(line)

redditgraph = []
redditgraph = readfile( 'RedditGraphGrouped2.tsv' )
redditgraphsize = len(redditgraph)

subreddits = []
subreddits = readfile( 'subredditsGrouped.tsv' )
subredditssize = len(subreddits)
for line in subreddits:
        line[1] = line[1].lower()

#Get the Reddit comments
reddittext = []
with open('redditcommentsgrouped.tsv', 'r') as tsv_in:
        for redditUsers, line in enumerate(csv.reader(tsv_in, dialect="excel-tab")):
                        reddittext.append(line)

#Remove non-letters, get meaningful words
```

```
def review_to_words( all_text ):
        letters_only = re.sub("[^a-zA-Z]", " ", all_text)
        words = letters_only.lower().split()
        meaningful_words = [w for w in words if not w in stopwords.words("english")]
        return( ' '.join(meaningful_words) )

usertext = []
for user in reddittext:
        user[1] = review_to_words(user[1])

for user in fbtext:
        user[1] = review_to_words(user[1])

usertext = reddittext + fbtext

#Build global lists for feature selection
word_features = []
for user in usertext:
        words = user[1].lower().split()
        all_words = nltk.FreqDist(w.lower() for w in words)
        word_features = list(all_words)[:20] + word_features

all_friends = []
for user in fbtext:
        all_friends.append(user[0])

all_interests = []
for user in subreddits:
        interests = user[1].split(',')
        all_interests = interests + all_interests

for user in fblikes:
        interests = user[1].split(',')
        all_interests = interests + all_interests

#count in degree

def graphedges( graph ):
        graph_metrics = []
        for user in graph:
                friends = []
                friends = set(user[1].split(','))
                edges = len(friends)
                graph_metrics.append(edges)
```

```
        return( graph_metrics )

redditgraph.sort()
redditgraph_edges = []
redditgraph_edges = graphedges( redditgraph )
fbgraph.sort()
fbgraph_edges = []
fbgraph_edges = graphedges( fbgraph )

#define the function for feature selection
def user_features(user, graph, interests):
        user_text = set(user[1].split())
        features = {}
        for word in word_features:
                features['contains({})'.format(word)] = (word in user_text)
        friends = []
        for line in graph:
                if user[0] == line[0]:
                        friends = set(line[1].split(','))
                        break
        for friend in all_friends:
                features['friendswith({})'.format(friend)] = (friend in friends)
        interests = []
        for line in interests:
                if user[0] == line[0]:
                        interests = set(line[1].split(','))
                        break
        for interest in all_interests:
                features['likes({})'.format(interest)] = (interest in interests)
        return features

selectfbtext = []
for user in fbtext:
        for line in reddittext:
                if user[0] == line[0]:
                        selectfbtext.append(user)

for e, line in enumerate(redditgraph_edges):
        if line == 1:
                for user in fbtext:
                        if redditgraph[e][0] == user[0] and user[0] != 'Joe Burpoe':
                                selectfbtext.append(user)

selectreddittext = []
```

```
for line in selectfbtext:
        for user in reddittext:
                if line[0] == user[0]:
                        selectreddittext.append(user)

#Removing friends from the feature set
def user_features(user, graph, interests):
        user_text = set(user[1].split())
        features = {}
        for word in word_features:
                features['contains({})'.format(word)] = (word in user_text)
        interests = []
        for line in interests:
                if user[0] == line[0]:
                        interests = set(line[1].split(','))
                        break
        for interest in all_interests:
                features['likes({})'.format(interest)] = (interest in interests)
        return features

fbfeatureset = [(user_features(user, fbgraph, fblikes), user[0]) for user in selectfbtext]
redditfeatureset = [(user_features(user, redditgraph, subreddits), user[0]) for user in
selectreddittext]
featureset = fbfeatureset + redditfeatureset

train_set, test_set = featureset[2:], featureset[:2]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
classifier.show_most_informative_features(5)

classifier = nltk.DecisionTreeClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)

for e, line in enumerate(redditgraph_edges):
        if line == 1:
                for user in fbtext:
                        if redditgraph[e][0] == user[0]:
                                selectfbtext.append(user)

selectreddittext = []

for line in selectfbtext:
        for user in reddittext:
```

```
                if line[0] == user[0]:
                        selectredditext.append(user)

fbfeatureset = [(user_features(user, fbgraph, fblikes), user[0]) for user in selectfbtext]
redditfeatureset = [(user_features(user, redditgraph, subreddits), user[0]) for user in
selectredditext]
featureset = fbfeatureset + redditfeatureset

train_set, test_set = featureset[2:], featureset[:2]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
classifier.show_most_informative_features(5)

classifier = nltk.DecisionTreeClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)

#identify nodes with 4 edges
selectfbtext = []
for e, line in enumerate(redditgraph_edges):
        if line == 4:
                for user in fbtext:
                        if redditgraph[e][0] == user[0]:
                                selectfbtext.append(user)

selectredditext = []

for line in selectfbtext:
        for user in reddittext:
                if line[0] == user[0]:
                        selectredditext.append(user)

fbfeatureset = [(user_features(user, fbgraph, fblikes), user[0]) for user in selectfbtext]
redditfeatureset = [(user_features(user, redditgraph, subreddits), user[0]) for user in
selectredditext]
featureset = fbfeatureset + redditfeatureset

train_set, test_set = featureset[2:], featureset[:2]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
classifier.show_most_informative_features(5)

classifier = nltk.DecisionTreeClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
```

```
#identify nodes with 3 edges
selectfbtext = []
for e, line in enumerate(redditgraph_edges):
        if line == 3:
                for user in fbtext:
                        if redditgraph[e][0] == user[0]:
                                selectfbtext.append(user)

selectredditext = []

for line in selectfbtext:
        for user in reddittext:
                if line[0] == user[0]:
                        selectredditext.append(user)

fbfeatureset = [(user_features(user, fbgraph, fblikes), user[0]) for user in selectfbtext]
redditfeatureset = [(user_features(user, redditgraph, subreddits), user[0]) for user in
selectredditext]
featureset = fbfeatureset + redditfeatureset

train_set, test_set = featureset[2:], featureset[:2]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
classifier.show_most_informative_features(5)

classifier = nltk.DecisionTreeClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)

#identify nodes with 2 edges
selectfbtext = []
for e, line in enumerate(redditgraph_edges):
        if line == 2:
                for user in fbtext:
                        if redditgraph[e][0] == user[0] and user[0]:
                                selectfbtext.append(user)

selectredditext = []

for line in selectfbtext:
        for user in reddittext:
                if line[0] == user[0]:
                        selectredditext.append(user)

fbfeatureset = [(user_features(user, fbgraph, fblikes), user[0]) for user in selectfbtext]
```

```
redditfeatureset = [(user_features(user, redditgraph, subreddits), user[0]) for user in
selectreddittext]
featureset = fbfeatureset + redditfeatureset

train_set, test_set = featureset[1:], featureset[:3]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
classifier.show_most_informative_features(5)

classifier = nltk.DecisionTreeClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)

#Removing text analysis from the feature set
def user_features(user, graph, interests):
        features = {}
        friends = []
        for line in graph:
                if user[0] == line[0]:
                        friends = set(line[1].split(','))
                        break
        for friend in all_friends:
                features['friendswith({})'.format(friend)] = (friend in friends)
        interests = []
        for line in interests:
                if user[0] == line[0]:
                        interests = set(line[1].split(','))
                        break
        for interest in all_interests:
                features['likes({})'.format(interest)] = (interest in interests)
        return features

#Attempt to identify remaining unidentified nodes
redditgraph = []
redditgraph = readfile( 'RedditGraphMostlySolved.tsv' )
redditgraphsize = len(redditgraph)

#define the function for feature selection
def user_features(user, graph, interests):
        user_text = set(user[1].split())
        features = {}
        for word in word_features:
                features['contains({})'.format(word)] = (word in user_text)
        friends = []
        for line in graph:
```

```
                if user[0] == line[0]:
                        friends = set(line[1].split(','))
                        break
        for friend in all_friends:
                features['friendswith({})'.format(friend)] = (friend in friends)
        interests = []
        for line in interests:
                if user[0] == line[0]:
                        interests = set(line[1].split(','))
                        break
        for interest in all_interests:
                features['likes({})'.format(interest)] = (interest in interests)
        return features

selectfbtext = []
selectredditext = []
for user in fbtext:
        if user[0] == '3' or user[0] == '2' or user[0] == '1':
                for line in reddittext:
                        if line[0] == user[0]:
                                selectfbtext.append(user)
                                selectredditext.append(user)

fbfeatureset = [(user_features(user, fbgraph, fblikes), user[0]) for user in selectfbtext]
redditfeatureset = [(user_features(user, redditgraph, subreddits), user[0]) for user in
selectredditext]
featureset = fbfeatureset + redditfeatureset

train_set, test_set = featureset[3:], featureset[:3]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
classifier.show_most_informative_features(5)

classifier = nltk.DecisionTreeClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)


#Removing friends from the feature set
def user_features(user, graph, interests):
        interests = []
        for line in interests:
                if user[0] == line[0]:
                        interests = set(line[1].split(','))
                        break
```

```
        for interest in all_interests:
                features['likes({})'.format(interest)] = (interest in interests)
        return features

#Removing interests
def user_features(user, graph, interests):
        user_text = set(user[1].split())
        features = {}
        for word in word_features:
                features['contains({})'.format(word)] = (word in user_text)
        friends = []
        for line in graph:
                if user[0] == line[0]:
                        friends = set(line[1].split(','))
                        break
        for friend in all_friends:
                features['friendswith({})'.format(friend)] = (friend in friends)
        return features


#create feature vectors for text - This section can be used for text analysis features
fvs_text = np.zeros((len(reddittext), 4),  np.float64)
for e, user in enumerate(reddittext):
        upperl = sum(1 for c in user[1] if c.isupper())
        tokens = nltk.word_tokenize(user[1].lower())
        words = word_tokenizer.tokenize(user[1].lower())
        sentences = sentence_tokenizer.tokenize(user[1])
        vocab = set(words)
        words_per_sentence = np.array([len(word_tokenizer.tokenize(s)) for s in sentences])
        fvs_text[e, 0] = words_per_sentence.mean()
        fvs_text[e, 1] = words_per_sentence.std()
        fvs_text[e, 2] = len(vocab) / float(len(words))
        fvs_text[e, 3] = upperl / len(sentences)
        fvs_text[e, 4] = tokens.count(',') / float(len(sentences))
        fvs_text[e, 5] = tokens.count(';') / float(len(sentences))
        fvs_text[e, 6] = tokens.count(':') / float(len(sentences))
        fvs_text[e, 7] = tokens.count('!') / float(len(sentences))
        fvs_text[e, 8] = tokens.count('.') / float(len(sentences))
        blob = TextBlob(user[1], analyzer=NaiveBayesAnalyzer())
        fvs_text[e, 9] = blob.sentiment.p_pos


# apply whitening to decorrelate the features
fvs_text = whiten(fvs_text)
```

```
#This section is for collecting bag of words


clean_text = []
for users in usertext:
        clean_text.append(review_to_words(users[1]))

vectorizer = CountVectorizer(analyzer = "word",   \
                        tokenizer = nltk.word_tokenize,    \
                        preprocessor = None, \
                        stop_words = None,   \
                        max_features = 500)

train_data_features = vectorizer.fit_transform(clean_text[trainset:])
train_data_features = train_data_features.toarray()
#vocab = vectorizer.get_feature_names() this is for testing

test_data_features = vectorizer.transform(clean_text[:testset])
test_data_features = test_data_features.toarray()
#End bag of words

trainset, testset = fvs_text[:fbUsers+3], fvs_text[fbUsers+3:]
km = KMeans(n_clusters=22, init='random', n_init=10)
km.fit(trainset)
results = km.predict(testset)

for index, result in enumerate(results):
        print usertext[result][0], usertext[fbUsers+3+index][0]

#Print for debug
print fvs_lexical_fb
print fvs_lexical_red
print fvs_punct_fb
print fvs_punct_red
print vocab

#get most common words
#NUM_TOP_WORDS = 10
#all_tokens = nltk.word_tokenize(all_text)
#fdist = nltk.FreqDist(all_tokens)
#vocab = fdist.keys()[:NUM_TOP_WORDS]

#vectorizer = CountVectorizer(vocabulary=vocab, tokenizer=nltk.word_tokenize)
```

```
#fvs_bow = vectorizer.fit_transform(textlist).toarray().astype(np.float64)
# normalise by dividing each row by its Euclidean norm
#fvs_bow /= np.c_[np.apply_along_axis(np.linalg.norm, 1, fvs_bow)]
#These features are not helpful
        fvs_text[e, 3] = tokens.count(',') / float(len(sentences))
        fvs_text[e, 4] = tokens.count(';') / float(len(sentences))
        fvs_text[e, 5] = tokens.count(':') / float(len(sentences))
        fvs_text[e, 6] = tokens.count('!') / float(len(sentences))
        fvs_text[e, 7] = tokens.count('\'') / float(len(sentences))
        fvs_text[e, 8] = tokens.count('.') / float(len(sentences))
        fvs_text[e, 9] = tokens.count('-') / float(len(sentences))


#Random Forests
forest = RandomForestClassifier(n_estimators = 100)
forest = forest.fit( train_data_features, fvs_punct_fb )
result = forest.predict(test_data_features)
```