IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 4, Ver. III (Jul – Aug. 2014), PP 39-49
www.iosrjournals.org

Informational Retrieval Using Crawler & Protecting Social Networking Data from Information leakage

S. S. Wangikar, S. N. Deshmukh
Dept. of CS & IT Dr. B.A.M. University, Aurangabad (M.S), India

Abstract: *Online social networks, such as Facebook, Twitter, Yahoo!, Google+ are utilized by many people. These networks allow users to publish details about themselves and to connect to their friends. Some of the information revealed inside these networks is meant to be private or public. Yet it is possible to use learning algorithms on released data to predict private or public information and use the classification algorithm on the collected data. In this paper, we explore how to get social networking data to predict information. We then devise possible classification techniques that could be used in various situations. Then, we explore the effectiveness of these techniques and attempt to use. We collect the different information from the users groups. On which we concluded the classification of that data. By using the various algorithms we can predict information of users. Crawler programs for current profile work. We constructed a spider that crawls & indexes FACBOOK. In this paper we focus on crawler programs that proved to be an effective tool of data base.*
In this paper we elaborate the use of data mining technique to help retailers to identify user profile for a retail store and improved. The aim is to judge the accuracy of different data mining algorithms on various data sets. The performance analysis depends on many factors encompassing test mode, different nature of data sets, and size of data set.
Keyword: *Social network analysis, data mining, social network data,WEKA.*

I. Introduction

Social networks are online applications that allow their users to connect by means of various link types. As part of their offerings, these networks allow people to list details about themselves that are relevant to the nature of the network. For instance, Facebook is a general-use social network, so individual users list their favourite activities, books, and movies. Facebook is one of the foremost social networking websites, with over million users spanning in college campuses [2]. Conversely, LinkedIn is a professional network; because of this, users specify details which are related to their professional life (i.e., reference letters, previous employment, and so on.)Because these sites gather extensive personal information, social network application providers have a rare opportunity: direct use of this information could be useful to advertisers for direct marketing. In this paper, we explore how to get social networking data to predict information. We constructed a spider that crawls & indexes FACBOOK. The general research challenge is to build up a well-structured database that suits well to the given research question and that is cost efficient at the same time. Firstly we explain how crawler works.

In this paper we focus on crawler programs that proved to be an effective tool of data base building in very different problem settings our goal was to protect that data. We constructed a spider that "crawls" and indexes Facebook. It's really easy with the Facebook Graph API. Sign in to Facebook and navigate to the Graph API Explorer app. Click the "Get Access Token" button. For production use, we need to create a Facebook Application and use it to obtain OAuth tokens for our users. Take a look at the FB Authentication documentation for more details. We constructed a threat model that attempted to address all possible categories of privacy failures. In reality, this web crawler is a program, which automatically traverses the web by downloading documents and following links from page to page. They are mainly used by search engine to gather data for indexing. Other possible applications include page validation, structural analysis and visualization, update notification, mirroring and personal web assistants/ agents etc. After crawler implementation apply the various data mining algorithms on that datasets.

Data mining techniques that extract information from huge amount of data have become popular in commercial and scientific domains, including marketing, customer relationship management. During the evaluation, the input datasets and the number of classifiers used are varied to measure the performance of Data Mining algorithm. The results are based on characteristics such as scalability, accuracy to identify their characteristics in a world famous Data Mining tool-WEKA.

1.1 Web Crawler Methodology

Web crawler programs are as old as the world wide web [3].They are short software codes sometimes also called as bots, ants or worms written with the objective to download web pages, extract hyperlinks, create a list of its URLs and add them to a local database[4]. Their most widely spread applications are search engines in which form they are familiar to all internet users [5]. One of the first publications of how effective search crawling and indexing can be on the web was published by Google owners in 1998 as a widely referred academic paper on crawler programming [6].Building a web crawler, first of all requires the knowledge of how exactly the users browse a website, and what happens during this process from an information processing point of view. Based on this, the simple idea behind programming a crawler is to (i) imitate the actions of a user visiting a webpage, (ii) extract the information we need, and (iii) repeat these steps. Using this idea we identified three conceptually different type of crawling method each of which is suitable to collect data for different type of analysis and related problem statement.

The first and, we might say, the simplest is when the data we are interested can be found on one site at a well defined place or technically record and its value depends on time. For instance this is the logic of storing and presenting market data on some aggregate websites, or showing price information about particular products in a web store or personal information about employees. In these cases, the data structure is static so the crawler has always visited the same page and same record and periodically downloads its actual value into a database.
The second type of crawler is more complex in a sense that it imitates a user who is visiting a list of websites one after the other and downloads data, usually different records from each of these sites which after the completion of the visits is grouped, sorted, and analyzed according to the problem or research design.

The third, and most complex, type of search when the browsing sequence is not predetermined neither by its sequence of websites nor the actually numbers of them. This happens typically when we explore a domain of a topic, for instance collect information about graduate programs at universities, looking for a particular service, or try to find out the size and connection structure of a community on the net. Since this most complicated case entails both previous once technically and in this present paper we use an example for crawler illustration from this type we describe it in more details.

II. Literature Survey

2.1 Facebook Platform

Facebook (http://www.facebook.com) has grown phenomenally over the past several years from an Ivy League social web application to the second largest social web site on the Internet. The creators of Facebook have done an impressive job focusing their social software on the college demographic. In a natural progression of the social network, Facebook recently extended its network by developing a platform for developers to create new applications to allow Facebook users to interact in new and exciting ways.

FACEBOOK consists of an HTML-based markup language called Facebook Markup Language (FBML), an application programming interface (API) for making representational state transfer (REST) calls to Facebook, a SQL-styled query language for interacting with Facebook called Facebook Query Language (FQL), a scripting language called Facebook JavaScript for enriching the user experience, and a set of client programming libraries. Generically, the tools that make up the Facebook platform are loosely called the Facebook API.

2.1.1 The Elements of the Facebook Platform

As stated previously, the Facebook platform consists of five components: a markup language derived from HTML (Facebook Markup Language), a REST API for handling communication between Facebook and application, a SQL-style language for interacting with Facebook data (Facebook Query Language), a scripting language (Facebook JavaScript), and a set of client libraries for different programming languages.

2.2 Related Work for Web Crawler

Web crawlers are almost as old as the web itself. Web crawlers are almost as old as the web itself. In the spring of 1993, just months after the release of NCSA Mosaic, Matthew Gray [8] wrote the first web crawler, the World Wide Web Wanderer, which was used from 1993 to 1996 to compile statistics about the growth of the web. The Wanderer was written in Perl and ran on a single machine. It was used until 1996 to collect statistics about the evolution of the web. Moreover, the pages crawled by the Wanderer were compiled into an index (the "Wandex"), thus giving rise to the first search engine on the web.

A year later, David Eichmann [9] wrote the first research paper containing a short description of a web crawler, the RBSE spider.

Burner provided the first detailed description of the architecture of a web crawler, namely the original Internet Archive crawler [10]. The IA design made it very easy to throttle requests to a given host, thereby addressing politeness concerns, and DNS and robot exclusion lookups for a given web site were amortized over all the site's URLs crawled in a single round. However, it is not clear whether the batch 182 Crawler Architecture process of integrating off-site links into the per-site queues would scale to substantially larger web crawls. To avoid adding multiple instances of the same URL to the queue, the IA crawler maintained an in-memory Bloom filter [11] of all the site's URLs discovered so far.

Brin and Page's seminal paper on the (early) architecture of the Google search engine contained a brief description of the Google crawler, which used a distributed system of page-fetching processes and a central database for coordinating the crawl. The original Google crawling system consisted of a single URLserver process that maintained the state of the crawl, and around four crawling processes that downloaded pages. Heydon and Najork described Mercator [11,12], a distributed and extensible web crawler that was to become the blueprint for a number of other crawlers. Mercator was written in Java, highly scalable, and easily extensible. The first version [12] was non-distributed; a later distributed version [13] partitioned the URL space over the crawlers according to host name, and avoided the potential bottleneck of a centralized URL server. The second Mercator paper gave statistics of a 17-day, four-machine crawl that covered 891 million pages. Mercator was used in a number of web mining papers [14, 15, 16, 17, 18], and in 2001 replaced the first-generation AltaVista crawler.

Other distributed crawling systems described in the literature include PolyBot [19], UbiCrawler [20], C-proc [21] and Dominos [22]. In some web crawler designs (e.g., the original Google crawler [6] and PolyBot [19]), the page downloading processes are distributed, while the major data structures the set of discovered URLs and the set of URLs that have to be downloaded are maintained by a single machine. This design is conceptually simple, but it does not scale indefinitely; eventually the central data structures become a bottleneck.

Shkapenyuk and Suel's Polybot web crawler [19] represents another "blueprint design." Polybot is a distributed system, consisting of a crawl manager process, multiple downloader processes, and a DNS resolver process. The paper describes scalable data structures for the URL frontier and the "seen-URL" set used to avoid crawling the same URL multiple times; it also discusses techniques for ensuring politeness without slowing down the crawl. Polybot was able to download 120 million pages over 18 days using four machines. The IBM WebFountain crawler [23] represented another industrial strength design. The WebFountain crawler was fully distributed. The three major components were multi-threaded crawling processes ("Ants"), duplicate detection processes responsible for identifying downloaded pages with near-duplicate content, and a central controller process responsible for assigning work to the Ants and for monitoring the overall state of the system.

The WebFountain crawler was written in C++ and used MPI (the Message Passing Interface) to facilitate communication between the various processes. It was reportedly deployed on a cluster of 48 crawling machines [24].

UbiCrawler [20] is another scalable distributed web crawler. It uses consistent hashing to partition URLs according to their host component across crawling machines, leading to graceful performance degradation in the event of the failure of a crawling machine. UbiCrawler was able to download about 10 million pages per day using five crawling machines. UbiCrawler has been used for studies of properties of the African web [25] and to compile several reference collections of web pages [26].

Heritrix [27, 28] is the crawler used by the Internet Archive. It is written in Java and highly componentized, and its design is quite similar to that of Mercator. Heritrix is multithreaded, but not distributed, and as such suitable for conducting moderately sized crawls. The Nutch crawler [29, 30] is written in Java as well. It supports distributed operation and should therefore be suitable for very large crawls; but as of the writing of [30] it has not been scaled
beyond 100 million pages.

2.3 Previous Paper Work

In their model, the network consists of only nodes and edges. Detail values are not included. The goal of the attacker is simply to identify people. Hay [33] and Liu and Terzi [34] consider several ways of anonymizing social networks. However, our work focuses on inferring details from nodes in the network, not individually identifying individuals. In [35], Hay considers ways to infer private information via friendship links by creating a Bayesian network from the links inside a social network. Also, compared to [36], we provide techniques that can help with choosing the most effective details or links that need to be removed for protecting privacy. Finally, we explore the effect of collective inference techniques in possible inference attacks.

In [37], Gross examine specific usage instances at Carnegie Mellon. They also note potential attacks, such as node reidentification or stalking, that easily accessible data on Facebook could assist with. We extend on their work by experimentally examining the accuracy of some types of the demographic reidentification that they propose before and after sanitization. This finding coincides very well with the amount of data that we were able to crawl using a very simple crawler on a Facebook network. In [38], Talukder propose a method of

measuring the amount of information that a user reveals to the outside world and which automatically determines which information (on a per-user basis) should be removed to increase the privacy of an individual.
Finally, in [39], we do preliminary work on the effectiveness of our Details, Links, and Average classifiers and examine their effectiveness after removing some details from the graph. Here, we expand further by evaluating their effectiveness after removing details and links.

III. Proposed System

3.1 Module Description

In this paper the following are the modules. The first module is Facebook crawler. There is different Naïve Bayes classification algorithms used. These algorithms are applied on the collecting datasets from the current Facebook account. After applying the algorithms on the data sets we can classify the data sets on the basis of the user's different links. In this paper the main task completed was the crawler for the Facebook account. In the crawler we access the all the information of the users account and the users different groups .On which we can perform different data mining algorithms.

3.1.1 Module1: Data Gathering (Facebook crawler)

This model is described in details in the next chapter. The crawler was limited to only crawling profiles inside the database. Also some people having enabled the privacy restrictions on their profile which prevent from the crawlers. The total time for crawler was depends upon the access tokens times. One access tokens time having some times may be one and half hours or only 25 minutes.
Despite the numerous applications for Web crawlers, at the core they are all fundamentally the same. Following is the process by which fundamental Web crawlers work:
1. Download the Web page.
2. Parse through the downloaded page and retrieve all the links.
3. For each link retrieved, repeat the process.
Now let's look at each step of the process in more detail.

In the first step, a Web crawler takes a URL and downloads the page from the Internet at the given URL. Oftentimes the downloaded page is saved to a file on disk or put in a database.

Saving the page allows the crawler [40] or other software to go back later and manipulate the page, be it for indexing words (as in the case with a search engine) or for archiving the page for use by an automated archiver.

In the second step, a Web crawler parses through the downloaded page and retrieves the links to other pages. Each link in the page is defined with an HTML anchor tag similar to the one shown here : <A HREF="http://www.host.com/directory/file.html">Link</A> after the crawler has retrieved the links from the page, each link is added to a list of links to be crawled.

The third step of Web crawling repeats the process. All crawlers work in a recursive or loop fashion, but there are two different ways to handle it. Links can be crawled in a depth-first or breadth-first manner. Depth-first crawling follows each possible path to its conclusion before another path is tried. It works by finding the first link on the [41] first page. It then crawls the page associated with that link, finding the first link on the new page, and so on, until the end of the path has been reached.

For example, favorite books of "Bible" and "The Bible" should be considered the same detail. Further, there are often spelling mistakes or variations on the same noun. The normalization method we use is based upon a Porter stemmer presented in [42]. To normalize a detail, it was broken into words and each word was stemmed with a Porter stemmer then recombined. Two details that normalized to the same value were considered the same for the purposes of the learning algorithm.

3.1.2 Module2: Formulation of Social Network

First, the modification of Naive Bayes classification is suitable for classifying large amount of social network data. Our modified Naive Bayes algorithm predicts information using both node and link structure. Compare the accuracy of our learning method based on link structure against the accuracy of our learning method based on node. Please see next chapter for further details of our modified Naive Bayes classifier there are a limited number of "groups" that are highly indicative of an individual user.

1. Naive Bayes Classification for prediction of user's information.
2. Naive Bayes on users groups.

3.1.3 Module3: Manipulating group Information

In this module first manipulating the group information then calculating the likehoods for each users and the groups likehoods. The manipulation of links is done by adding some details to nodes, modifying the existing node, removing node. Adding and modifying information can both be considered methods perturbation-that is introducing the “noise” into data to decrease classification accuracies. Removing a node, however can be considered an anoymization method.

IV. Implementation

4.1 Data Collection

Our collection of data directly from Facebook served two principles. It served as a proof of concept, to demonstrate that it is possible for an individual to automatically gather large amounts of data from Facebook. The collection of data was not entirely trivial, but we were able to produce the scripts necessary to do so within 48 hours. Also, the collection of data from Facebook will provide us with a large, nearly exhaustive and statistically significant data set, from which we can draw valuable conclusions on usage trends.

4.1.1 Working of General Crawler

When a user wants to open e.g. Facebook.com the followings takes place on the server and on the client side:
1. Generally the crawler has to have a connection function to reach a webpage.
2. The connection is parameterized with the request data (who we are, what we want, how long the program should wait for the answer, what is the maximum number of redirections)
3. Has to be able to receive the response, understand the status code, turn the source code to textual information, and with setting the proper character encoding, understand it (store in the memory and decode)
Using the above techniques, the crawler is ready to download and process a web page let us do the same and repeat it. Naturally, the repetition requires some further work, but it’s depending on the final purpose as well. In the followings, we present a situation where one domain (Facebook.com) is being inspected through thousands of subpages.
The following steps are describes in detail working of crawler:
1. First of all we need to login to Facebook.com with the crawler program as most of the data are available only to authenticated users. When a client signs in, the previous process is repeated except the second step where some additional parameters are set. These parameters are derived from the login form (mainly the username and the password) and from cookies, provided by the page.

2. After successful login, we select a certain person to analyze his profile. So after all, it is possible to extract the information we need (using connection stream analysis) and store them.

3. Not only one profile, but its connections are also important. At this step, the crawler saves all the friends, belonging to the previous person. The database stores only the user ID-s and Group IDs, the connections (in [parent, child] form), as the profile URLs can be recovered from them.

4. For a given set of connections the crawler uses systematic breadth-first-search procedure to continue the process from the second step. It means we analyze the profiles first and not the connections. As long as there are unanalyzed parents, there is no child process happens.

4.2 Data Acquisition from Current Users Profile through Facebook Crawler

We are not the first to download user profiles from Facebook in large numbers. In the past, others have utilized Facebook's use of predictable, easy to understand URLs to automatically request information and save user information for further analysis. The algorithm we used to gather this data is very straightforward. Our approach used the user profile identifier to download information in large amount of datasets. The algorithm we used to gather this data is very straightforward:

The followings steps are described for getting the access token from current user’s profile which is used for the crawler. Online social networking sites are usually run by individual corporations, and are accessible via the Web. Some social networking sites allow for browsing the public data such as (e. g birthday, hometown, gender, groups etc.) which is added to the user’s profile links. The social networking is composed of user accounts and links between users. A users groups links along with profile, are visible to those who are in that groups. Thus, users and users group are able to explorer the social network by following the user-to-groups links, browsing the information about users and their group’s links. Most sites enable users to create and join special interest groups. Users can post messages to groups and upload shared content to the group. Certain groups are moderated; admission to such a group and postings to a group are controlled by a user designated as the group’s moderator. Other groups are unrestricted, allowing any member to join and post messages or content.

Once we have an account and have set up a server environment, the next step is to add Facebook’s Developer application. Go to http://www.facebook.com/developers, and install the application. The users are presented with when installing a new application. The check boxes allow certain functionality to be performed by the application, and they give users the ability to turn certain aspects of the application on and off. Once we have read the platform application's terms of use, just click the Add Developer button to install the Developer application.

Step1: Log in to Facebook and Create Application.

A Facebook developer gives the different types application. We can also create by the graph apps. Now, with all that out of the way, let's set up an application. At this point, need to tell Facebook about our application. In the Facebook Developer application (http://www.facebook.com/developers), click the Set Up New Application button.

Step2: Crate New Application ID

step 2 we create the new application ID. By following the step 1 we get the one new application ID for our new App Name.

Step3: Save this Application ID

The only required fields are the application name and the one confirming we have read the terms of service. However, to really do anything with an application, we will need to fill out the optional fields. If we already set up an application and didn't fill out these fields, we can change them by clicking the My Apps link in the Developer application.
In Step 3 we save the New App ID in our Facebook account.

Step4: Access New Application ID through our Sever (Apache Server)

In Step 4 we can access this App ID through our Apache Server. First we completed install Apache Server in our system then by our crawling programming we can access the our FACBOOK account. Then save this App ID in our crawler then save that access token ID.

Step5: Collection of Network Data

For this step require the user to be logged on to Facebook in order to execute the code. This example uses the require_login() method and stores the user's identification number (UID) in the variable user. The next call actually queries the Facebook API method friends get by wrapping it in the PHP method friends_get. Since the facebook object holds the user's identification number (their primary key or UID), there's no need to pass the UID to the API request. From this short sample code, we can see that code is actually setting up a REST client to call the facebook.friends.get API method and returns an array of user identifiers:

In step5 through our crawler program we collect the friends name data. In this step we get the friends name from our Facebook account. To implement our algorithm, we used Apache Tomcat Server for the network downloader. In addition to implementing the above algorithm, we made pretend to be another web browser by changing its user agent We took advantage of the fact that logins and passwords are not encrypted. This step is only use for the getting the our access token number from the current profile and get the only friends name. It's really easy with the Facebook Graph API. Sign in to Facebook and navigate to the Graph API Explorer app. Click the "Get Access Token" button. For production use, we need to create a Facebook Application and use it to obtain OAuth tokens for our users. Take a look at the FB Authentication documentation for more details.

4.3 Database Design

When developing a web application, one of the more complex tasks is developing a database back end. For example, using MySQL since it's quite typical for developers to have this as their back end. If using another RDBMS such as PostgreSQL, Oracle, Derby, or Microsoft SQL Server, there won't be much of difference because won't be getting into any of the more advanced features of the DBMS engine.

4.3.1 Working with SQL

Create a new SQL file with a connection to the database to insert the data. Database include some default data, just for testing purposes.
To add the new SQL file, switch to the Database Development view and select New > SQL File, or select New > Other > SQL Development > SQL File. Give file a name (such as facebook_app), and set connection information that set up earlier .Then click Finish. Next, simply type the earlier code (or copy and paste it) into the file, and save it. To add the tables, add Go between the SQL statements, right-click, and select Execute All. Check with favorite tool to see whether the tables were created properly. It's time for some dummy data so we can have something in the database just paste these lines into the SQL file.

Query="INSERT INTO DKT.GROUPUSER (UID, GID,GROUPNAME) VALUES ("+Uid+","+ gid+",'"+Group+"')"Now we have some data into database . If query the data
qry = "select COUNT(*) as rowcount from DKT.GROUPS where GID= " + GID; It will give the Gid of the users profiles.

4.3.2 Data Gathering Method

In the following diagrams database design was described in details. We wrote a program to crawl the Facebook network to gather data for our experiments. Written in Java 1.7, the crawler loaded a profile, parsed the details out of the HTML, and stored the details inside a oracle database. Then, the crawler loaded all friends of the current profile and stored the friends inside the database both as friendship links as well as groups information possible profiles to later crawl. Because the data inside a Facebook profile is free form text, it is critical that the input be normalized.

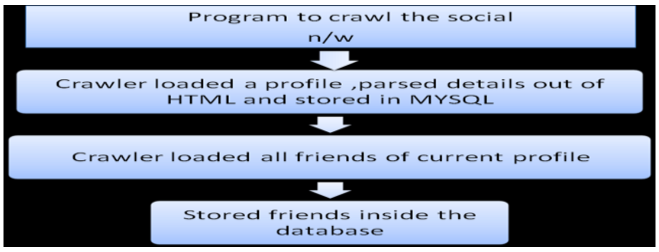
Program to crawl the social n/w
Crawler loaded a profile ,parsed details out of HTML and stored in MYSQL
Crawler loaded all friends of current profile
Stored friends inside the database

Figure 1: Data Gathering Method

In the next step we want to collect the groups from the particular profile users. For this we are using the fql query to fetch the data from the our profile. Now ,the crawler having the more information from the current user profile. The crawler is Proceed if the group is not present the our dataset. It will fetch the all the members of that groups. In the Groups tables we are having the information about the Group ID and the ISPROCSSED or NOT. In the Group User tables we are having the information about the Group ID ,USER ID and GROUP NAME.
Our total crawl resulted in over 11 groups of current profile. In table 1,We provide some general statistics of our Facebook data set.

4.3.3 Data Set

Dataset	Values
Number of Groups Proceed in Dataset	22
Number of Groups Not Proceed in dataset	1
Number of rows in dataset	26863
Number of User Details	39778

Table 1 : Facebook Dataset

4.3.4 Dataset Saved in the oracle :

Name of the dataset	Parameters stored in dataset
Friendship Link	UID, FRIEND ID
Groups	GID,ISPROCESSED
GroupUser	UID,GID,GROUP NAME
User Details	FNAME,LNAME,SEX,BDAY,EMAIL,EDUCATION,RSTAT,WORKEX,ID

Table 2 : Dataset in Oracle

The following are the some 22 groups collected from current user facebook crawler where we apply the different algorithms on that datasets.
A= Ambajogai Online
B= Arya group of lovers
C= BAAAP grp
D= BAMU Online
E= COEA ALUMNI ERA
F= Computer_Science_Books_Free_Download
G= Dept of CS & IT, Dr. B.A.M.U. Aurangabad
H=Freshers 2014 batch offcampus News
I=GATE 2012 CSE
J=GATE 2014 Zara HatK3 For Computer Science & Information Technology
K=Gate 2014 CS/IT

L=General Knowledge
M=General Knowledge & IQ
N=General knowledge and IQ
O=Genral Knowledge Share & Learn
P=IT JOBS
Q=Job Fair 2011
R=M.B.E.S.College Of Engineering,Ambajogai.
S=Mtech Rocksssssssssssss.........
T=Satyamev Jayate
U=Techanical information

V. Implementation of Algorithms Using WEKA

5.1 Analysis of Data Mining algorithms:

5.1.1 Classification Algorithm

A classification algorithm is to use a training data set to build a model such that the model can be used to assign unclassified records in to one of the defined classes. A test set is used to determine the accuracy of the model. Usually ,the given dataset is divided in to training and test sets, with training set used to build the model and test set used to validate it.

There are various classifiers are an efficient and scalable variation of decision tree classification. Bayesian classifiers are statistical based on Bayes‟ theorem, they predict the probability that a record belongs to a particular class. A simple Bayesian classifier, called Naïve Bayesian classifier is comparable in performance to decision tree and exhibits high accuracy and speed when applied to large databases.

We implemented algorithms to prediction for of each user. The first algorithm is called "Link Only." This algorithm uses [36] to prediction and ignores friendship links. In this algorithms local classification result is shown. The result was shown throughout the our current user profile groups. In this we are having the Link removal algorithm where we can manipulate the link information. The some probability formulas are used. The second algorithm is called "Details Only." This algorithm uses [37] to predict political affiliation using friendship links and does not consider the details of a person. Here the local classification is done for the each row in the datasets which are selected differently from the class

1. Naïve Bayes Classification algorithms:

Naive Bayes is a classifier that uses Bayes Theorem to classify objects.
Given a node, ni, with m details and p potential classes to choose from, $C_1, \ldots\ldots, C_p$, Naive Bayes determines which class Cx is more likely under the assumption that traits are independent. That is, argmax $[P(C_x|D^1{}_i \ldots D^m{}_{,i})]$ where argmax $1 \leq x \leq p$ represent the possible class that maximizes. By applying the Naïve Bayes algorithm

$$\underset{1 \leq x \leq p}{\text{argmax}} \left(\frac{P(C_x) \times P(D^1{}_i|C_x) \;\ldots\ldots\; P(D^m{}_i|Cx)}{P(D^1{}_i, \ldots, D^m{}_i)} \right)$$

Because $P(D^1{}_i, \ldots, D(^m{}_i))$ is a positive constant over all possible classes for any given user ni, this factor becomes irrelevant when probabilities are compared. This reduces our more difficult original problem of P (Cx|D1,i . . . Dm,i).

2. Naive Bayes on Links

Consider the problem of determining the class trait of person ni given their friendship links using a Naive Bayes model. That is, of calculating P (Cx|Fi,1, . . . , Fi,m). Because there are relatively few people in the training set that have a link to ni, the calculations for P (Cx|Fi,j) become extremely inaccurate. Instead, we decompose this relationship. Rather than having a link from person ni to nj , we instead consider the probability of having a link from ni to someone with nj is same.

5.1.2 Clustering Algorithm

Clustering is the process of discovering the groups of similar objects from a database to characterize the underlying data distribution. K-means is a partition based method and arguably the most commonly used clustering technique. K-means clustered assigns each object to its nearest cluster centre based on some similarity function. Once the assignments are completed, new centres are found by the mean of all the objects in each cluster.

Density based methods grow clusters according to some other density function. DBscan , originally proposed in astrophysics is a typical density based clustering method. After assigning an estimation of its density for each particle with its densest neighbours, the assignment process continues until the densest neighbour of a particle is itself. All particles reaching this state are clustered as a group.

S/W tool:

In all the experiments, we used Weka 3-6-9 we looked at different characteristics of the applications using classifiers to measure the accuracy in different data sets, using clustered to generate number of clusters, time taken to build models etc.

Input data sets:

Input data is an integral part of data mining applications. The data used in my experiment is either real world data obtained from Facebook social networking and widely accepted dataset available in Weka toolkit, during evaluation multiple data sizes were used, each dataset is described by the data type being used, the types of attributes, the number of instances stored within the dataset, also the table demonstrates that all the selected data sets are used for the classification and clustering task. These datasets were chosen because they have different characteristics and have addressed different areas.

Other of data Set:

We used data set for evaluation with classifier on WEKA, one of them from Data repository that are Cenus data set, in WEKA 3-6-9 .Census data set in csv file format. It contains the near about 32562 user's dataset. The parameters of this datasets are age, work class, education, marital status, occupation, gender, native country and income etc. For implementing the classification and the clustering algorithms on this dataset divides that datasets into the different classes and the clusters.

VI. Evaluation on Data Set

To evaluate the performance of various classifiers on two test modes 10 fold cross validation and percentage split with different data sets at WEKA 3-6-9, The results after evaluation is described here.

6.1.1 Evaluation of classifier on Data set

Classifier	Classifier Model	Test mode	Mean absolute Error	Root mean squared error	Relative absolute error	Root relative squared error
Naïve Bayes	Full training set	10 fold cross validation	0.173	0.3715	47.3191	86.884
BayesNet	Full training set	10 fold cross validation	0.1766	0.3438	48.2896	80.3974
NaïveBayes simple	Full training set	10 fold cross validation	0.1739	0.3732	47.5538	87.2759
NaiveBayes Updatable	Full training set	10 fold cross validation	0.173	0.3715	47.3191	86.884

Table 3: Evaluation of classifiers on Cencus data set with cross validation test mode

Classifier	Classifier Model	Test mode	Mean absolute Error	Root mean squared error	Relative absolute error	Root relative squared error
Naïve Bayes	Full training set	Percentage split	0.1687	0.3658	46.3148	86.2523
BayesNet	Full training set	Percentage split	0.1749	0.3406	48.0239	80.3221
NaïveBayes simple	Full training set	Percentage split	0.1694	0.3677	46.5222	86.7013
NaiveBayes Updatable	Full training set	Percentage split	0.1687	0.3658	46.3148	86.2523

Table 4: Evaluation of classifiers on Cencus data set with percentage split test mode

Classifier	Classifier Model	Test mode	Mean absolute Error	Root mean squared error	Relative absolute error	Root relative squared error
Naïve Bayes	Full training set	10 fold cross validation	0.0307	0.1391	35.9072	67.229
BayesNet	Full training set	10 fold cross validation	0.0316	0.1398	37.6986	68.3423
NaïveBayes simple	Full training set	10 fold cross validation	0.0319	0.1409	37.5438	68.7459
NaiveBayes Updatable	Full training set	10 fold cross validation	0.0307	0.1391	35.9072	67.229

Table 5: Evaluation of classifiers on Facebook group data set with cross validation test mode

6.1.2 Evaluation of clustered on data set

Clustering Algorithm	No. of Instances	Test mode	No. of cluster generated	Time taken to build the model	Unclustered
DB-Scan	32561	Full Training data	866	302.29	12468
EM	32561	Full Training data	1345	585.59	Null
K-Mean	32561	Full Training data	13676	1.67	Null

Table 6: Evaluation of clustered on Cencus data set with Full training data test mode

Clustering Algorithm	No. of Instances	Test mode	No. of cluster generated	Clustered Instances	Time taken to build the model	Unclustered
DB-Scan	32561	Percentage split	866	32561	292.7	4731
EM	32561	Percentage split	3	32561	586.7	null
K-Mean	32561	Percentage split	6105	32561	1.12	4966

Table 7: Evaluation of clustered on Cencus data set with Percentage split test mode

6.2 Experimental Result

To evaluate the selected tool using the given datasets, several experiments are conducted. For evaluation purpose, two test modes are used, the k-fold cross-validation(k-fold cv) mode, & percentage split(holdout method) mode. The k-fold cv refers to a widely used experimental testing procedure where the database is randomly divided in to k disjoint blocks of objects, then the data mining algorithm is trained using k-1 blocks and the remaining block is used to test the performance of the algorithm, this process is repeated k times. At the end, the recorded measures are averaged. It is common to choose k=10 or any other size depending mainly on the size of the original dataset.

In percentage split (holdout method) ,the database is randomly split in to two disjoint datasets. The first set, which the data mining system tries to extract knowledge from called training set. The extracted knowledge may be tested against the second set which is called test set, it is common to randomly split a data set under the mining task in to 2 parts. It is common to have 66% of the objects of the original database as a training set and the rest of objects as a test set. Once the tests is carried out using the selected datasets, then using the available classification and test modes ,results are collected and an overall comparison is conducted.

6.2.1 Performance Measures

For each characteristic, we analyzed how the results vary whenever test mode is changed. Our measure of interest includes the analysis of classifiers and clusterers on different datasets, the results are described in value of correlation coefficient, mean absolute error, root mean squared error, relative absolute error, and root relative squared error after applying the cross-validation or holdout method.

For performance issues, after applying the BayesNet and Simple Naïve Bayes classifiers on Cencus dataset with 10 fold cross-validation methods, the mean absolute error are 0.1739 and 0.1766, and root mean squared error are 0.3438 and 0.3732 respectively. The BayesNet and Simple Naïve Bayes classifiers on Cencus dataset with percentage split method, the mean absolute error are 0.1749 and 0.1694, and root mean squared error are 0.3406 and 0.3677respectively. The BayesNet and Simple Naïve Bayes classifiers on Facebook group dataset with 10 fold cross-validation method, the mean absolute error are 0.0316 and 0.0319, and root mean squared error are 0.1398 and 0.1409 respectively.

VII. Conclusions

In first part we concluded, the analysis of collected datasets has been designed. We strongly advise all Facebook users to restrict access to their profiles, to not post information of illegal or policy-violating actions to their profiles, and to be cautious with the information they make available. This lasting change will only come with execution time and groups of the users .By crawler designing we can have the database of Facebook. In this database we can have the different user ID and group ID. Using this ID’s we can performed the various data mining techniques. We explain how crawler programs for current profile work. In future work we design the crawler for the other profile and related to that groups, pages, likes etc.
We show that using friendship data gives better predictability than details alone.

In second part we explored the effect of removing details in preventing information leakage. In the process, we discovered situations in which collective inferencing does not improve on using a simple local classification method to identify nodes.

Data Mining has a large family composed of different algorithms, and the scope of research is rapidly increasing to improve the accuracy of existed algorithms. In this paper, we evaluate some Data Mining algorithms classification algorithms, and clustering algorithms,. We analyzed important characteristics of the applications when executed in well known tool WEKA. The work described in this comparatively evaluates the performance of algorithms on test modes that is hold out method, percentage split, and full training.

When we combine the results from the individual results, we begin to see that details and friendship data together is the best way to reduce classifier accuracy. This is probably infeasible in maintaining the use of social networks. However, we also show that by removing only details, we greatly reduce the accuracy of local classifiers, which give us the maximum accuracy that we were able to achieve through any combination of classifiers.

References

[1]. R.Heatherly, M.Kantarcioglu, and B.Thuraisingham,"Preventing Private Information Inference Attacks on Social Networks" IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 8, August 2013.
[2]. Terremark Worldwide, Inc."Facebook Expands Operations at Terremark's NAP West Facility" Tuesday November 1, 8:30 am ET.
[3]. Risvik,K.M. and Michelsen,R. Search Engines and Web Dynamics. Computer Networks, Vol. 39, pp. 289–302, June 2002.
[4]. Chakrabarti,S.Mining the web: Analysis of hypertext and semi structured data. New York: Morgan Kaufmann-2003.
[5]. Pant G.,Srinivasan P.Menczer F.,"Crawling the Web", in Levene M., Poulovassilis A. Web Dynamics: Adapting to Change in Content, Size, Topology and Use, Springer, pp. 153–178-2004.
[6]. Brin,S.,Page L. The anatomy of a large-scale hypertextual Web search engine, Computer networks and ISDN systems, 30 (1-7). 107–117-1998.
[7]. WEKA at http://www.cs.waikato.ac.nz/~ml/weka.
[8]. Gray M. Internet Growth and Statistics: Credits and background. http://www.mit.edu/people/mkgray/net/background.html
[9]. Eichmann D. The RBSE Spider – Balancing effective search against web load. In Proc. 3rd Int. World Wide Web Conference, 1994.
[10]. Burner M. Crawling towards eternity: building an archive of the World Wide Web. Web Tech. Mag., 2(5):37–40, 1997.
[11]. B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, pp. 422–426, 1970.
[12]. Heydon A. and Najork M. Mercator: a scalable, extensible web crawler. World Wide Web, 2(4):219–229, December 1999.
[13]. Najork M. and Heydon A. High-performance web crawling. Compaq SRC Research Report 173, September 2001.
[14]. A. Broder, M. Najork, and J. Wiener, "Efficient URL caching for World Wide Web crawling," in Proceedings of the 12th International World Wide Web Conference, 2003
[15]. D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener, "A large-scale study of the evolution of web pages," in Proceedings of the 12th International World Wide Web Conference, 2003.
[16]. M. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork, "Measuring index quality using random walks on the web,"in Proceedings of the 8th International World Wide Web Conference, 1999.
[17]. M. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork, "On nearuniform URL sampling," in Proceedings of the 9th International World Wide Web Conference, 2000.
[18]. M. Najork and J. L. Wiener, "Breadth-first search crawling yields high-quality pages," in Proceedings of the 10th International World Wide Web Conference, 2001.
[19]. Shkapenyuk V. and Suel T. Design and Implementation of a high-performance distributed web crawler. In Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 357–368.
[20]. Boldi P., Codenotti B., Santini M., and Vigna S. UbiCrawler: a scalable fully distributed web crawler. Software Pract. Exper., 34(8):711–726, 2004.
[21]. Cho J. and Garcia-Molina H. Parallel crawlers. In Proc. 11th Int. World Wide Web Conference, 2002, pp. 124–135.
[22]. Hafri Y. and Djeraba C. High performance crawling system. In Proc. 6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval, 2004, pp. 299–306.
[23]. J. Edwards, K. S. McCurley, and J. A. Tomlin, "An adaptive model for optimizing performance of an incremental web crawler," in Proceedings of the 10th International World Wide Web Conference, 2001.
[24]. D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien, "How to build a WebFountain: An architecture for very large-scale text analytics," IBM Systems Journal, vol. 43, no. 1, pp. 64–77, 2004.
[25]. P. Boldi, B. Codenotti, M. Santini, and S. Vigna, "Structural properties of the African web," in Poster Proceedings of the 11th International World Wide Web Conference, 2002.
[26]. Yahoo! Research Barcelona, "Datasets for web spam detection," http://www.yr bcn.es/webspam/datasets.
[27]. Internet Archive, "Heritrix home page," http://crawler.archive.org/.
[28]. G. Mohr, M. Stack, I. Ranitovic, D. Avery, and M. Kimpton, "An introduction to Heritrix, an open source archival quality web crawler," in Proceedings of the 4th International Web Archiving Workshop, 2004.
[29]. A. S. Foundation, "Welcome to Nutch!," http://lucene.apache.org/nutch/.
[30]. R. Khare, D. Cutting, K. Sitakar, and A. Rifkin, "Nutch: A flexible and scalable open-source web search engine," Technical Report, CommerceNet Labs, 2004
[31]. www.eecs.northwestern.ed/~yingliu/papers/pdcs.pdf
[32]. O. S. Abbas in his article "comparison between data clustering algorithms".
[33]. M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava,"Anonymizing Social Networks," Technical Report 07-19, Univ.of Massachusetts Amherst, 2007.
[34]. K. Liu and E. Terzi, "Towards Identity Anonymization onGraphs," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'08), pp. 93-106, 2008.
[35]. J. He, W. Chu, and V. Liu, "Inferring Privacy Information from Social Networks," Proc. Intelligence and Security Informatics,2006.
[36]. R. Gross, A. Acquisti, and J.H. Heinz, "Information Revelation and Privacy in Online Social Networks," Proc. ACM Workshop Privacy in the Electronic Soc. (WPES '05), pp. 71-80, http://dx.doi.org/10.1145/1102199.1102214, 2005.
[37]. N. Talukder, M. Ouzzani, A.K. Elmagarmid, H. Elmeleegy, and M. Yakout, "Privometer: Privacy Protection in Social Networks,"Proc. IEEE 26th Int'l Conf. Data Eng. Workshops (ICDE '10), pp. 266-269, 2010.
[38]. J. Lindamood, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham,"Inferring Private Information Using Social Network Data,"Proc. 18th Int'l Conf. World Wide Web (WWW), 2009.
[39]. C. van Rijsbergen, S. Robertson, and M. Porter, "New Models in Probabilistic Information Retrieval," Technical Report 5587,British Library, 1980.
[40]. B. Bamba, L. Liu, J. Caverlee, V. Padliya, M. Srivatsa, T. Bansal, M. Palekar, J. Patrao, S. Li, and A. Singh, "DSphere: A source-centric approach to crawling, indexing and searching the world wide web," in Proceedings of the 23rd International Conference on Data Engineering, 2007.
[41]. Z. Bar-Yossef and M. Gurevich, "Random sampling from a search engine's index," in Proceedings of the 15th International World Wide Web Conference, 2006.
[42]. E. Zheleva and L. Getoor, "Preserving the Privacy of Sensitive Relationships in Graph Data," Proc. First ACM SIGKDD Int'l Conf. Privacy, Security, and Trust in KDD, pp. 153-171, 2008.