Towards the Design and Implementation of an OSN Crawler: A Case of Turkish Facebook Users

Önder Çoban†, Ali İnan*, Selma Ayşe Özel**

†,**Çukurova University, Computer Engineering Department, Adana, Turkey.
*Adana Alparslan Türkeş Science and Technology University, Computer Engineering Department, Adana, Turkey.
e-mail: †ocoban@cu.edu.tr, *ainan@atu.edu.tr, **saozel@cu.edu.tr

ORCID ID: 0000-0001-9404-2583, 0000-0002-3149-1565, 0000-0001-9201-6349
Research Paper Received: 27.02.2020 Revised: 04.04.2020 Accepted: 16.04.2020

Abstract—Online Social Networks (OSNs) are extremely popular services that allow users to interact with each other and share content. Due to the large amounts of data shared by users, OSNs are also rich data sources for research in social network analysis. Studying the usage of OSNs helps to understand users' content-sharing behavior and privacy concerns. In order to do so, collecting data is a necessary first step. However, Application Programming Interfaces (APIs) provided by OSN providers have several limitations which make it difficult to access secured information. In this paper, we present the design and implementation of an OSN crawler, discuss the challenges of this task and our workarounds towards accessing public OSN data. Moreover, we perform analyses of the collected data to indicate users' sharing behavior and give a detailed discussion of these analyses from the perspective of individual privacy protection over OSNs. Our crawler overcomes most of the restrictions of OSN APIs and collects all forms of OSN user interactions as well as every bit of public data posted on an OSN. Most of the existing studies collect OSN data using focused crawlers and therefore are capable of collecting only the desired type of data. Our crawler, on the other hand, provides a holistic view. On the popular Facebook OSN, our crawler captures user relationships like kinship, friendship and attributes like profile items, events, posts, comments, replies, meta-data of activities (i.e., posting time, location, tagged users etc.). To the best of our knowledge, ours is the most comprehensive OSN data collection effort and also the first study focused on the behavior of OSN users in Turkey.

Keywords—Online social networks, Facebook, crawler, data collection, content-sharing analysis.

1. Introduction

Online Social Networks (OSNs) have become popular platforms for users to connect with each other, express themselves, and share other information such as activities, photographs, favorites, and posts [1], [2]. Popular OSNs like Facebook, MySpace, and Twitter serve hundreds of millions of users that are connected with each other through more than a billion links. These OSNs are even said

to be able to reflect the real life [3]. Over the fourth quarter of 2019, Facebook had over 2.50 billion monthly active users (indicating an impressive 8% yearly increase [4]. It should not be surprising that OSNs are among the most visited websites [5] rivalling many popular search engines.

As a result of this popularity of OSNs, the data collocated with OSNs have attracted extensive attention from both commercial, governmental, and academic organizations. Governments, for example, are considering OSNs as a means for informing, engaging with and serving their citizens [6]. In the academy, researchers of various fields, ranging from sociology [7] to computer science [3], [8] have shown interest in OSN data. Within computer science, OSN research has focused on various problems such as extracting useful information [9], analyzing user behavior and social activity [10], finding similar users [11], identifying fake profiles [12], opinion trend analysis [6], content-sharing behavior [13], OSN usage [14] to list a few examples.

Despite such abundance of work on OSN research, publicly available OSN data is very scarce. To the best of our knowledge, data sets provided by the Stanford University under the Stanford Network Analysis Project (SNAP) [15] is the only example. These data sets are created by crawling different OSNs including Google+, Facebook, and Twitter. Compared to our crawled snapshot, majority of these data sets contain more nodes and edges, but they often focus social circles and some basic attributes of users.

The reasons behind the scarcity of OSN data are many-fold. We believe that the primary reasons are privacy concerns, complexity and volume of the OSN data, and the value of OSN data. These are briefly discussed next.

Privacy concerns: OSN service providers are obliged by laws and regulations such as the General Data Protection Regulation (GDPR) in Europe [16] and the *"Kişisel Verileri Koruma Kanunu" (KVKK)* [17] in Turkey [18] to protect the individual privacy of their respective users. As a result, direct access to user data is limited only to *public* shares by the users, or permission-based Application Programming Interface (API) calls. Privacy concerns also withhold researchers from sharing OSN data for academic purposes - SNAP data sets [15] are overly anonymized as a precaution against possible privacy violations.

High complexity and large volume of unstructured data contained in OSNs: by its nature, OSN data needs to be modelled as a graph. Considering various possible forms of interactions between users (e.g., friendship, kinship, wall posts, likes, etc.) and the variety of the involved data types (e.g., text, emoticons, categorical, etc.), building a comprehensive yet efficient data model for the storage and analysis of OSN data is an uneasy and time consuming task [3]. Predicted size of the network traffic needed to disclose the entire Facebook OSN is roughly 750 Gigabytes [14].

Monetary value of the OSN data: the value of many OSN service provider companies are rooted in their customer base. Facebook has always been a well-known example of this fact [19]. The rich content within OSN interactions of users create the value and the service providers prevent data collection efforts partially due to this value.

Existing studies obtain OSN data in one of the following two rather different ways: (i) using API calls provided by the OSN services providers (e.g., [11], [20]), or (ii) using web crawlers that systematically browse OSN profile pages to imitate a human user's HTTP interaction with the OSN servers (e.g., [1], [11]). Each solution has its own strong and weak sides. With the API approach, the data collector acts as an OSN application developer and has to abide

by the provider's strict access restrictions (e.g., limited view, query limits etc.) and limitations [20] and unauthorized developers are unable to obtain data [8], [14]. On the Facebook OSN, it is not possible to obtain the data of more than 25 likes or comments [11], [21] for a post. Other restrictions involve posing maximum 600 queries every 10 minutes [20], obtaining at most 400 friends [22] etc. In addition, Facebook frequently updates its complex privacy policy and strengthen its API restrictions [3], [8].

Crawlers can alleviate some of these restrictions and collect more information for scientific purposes [8]. However, with the crawler approach, the data collector has to impersonate a real user. Sending too many HTTP requests within a time frame raises suspicion by the service provider and the corresponding OSN account gets blocked temporarily or permanently due to malicious activity. Additional difficulties can be listed easily: masking data requests behind click streams, extracting OSN data from HTML responses, and dealing with web browsers instead of a programming interface.

In this study, we focus on the problem of efficient collection of large-scale, content-rich and accurate OSN data while respecting the individual privacy of the respective OSN users. We present an algorithmic data collection methodology that is applicable to any OSN. Our proposed solution in its simplest form is a web crawler that operates in the application layer, using HTTP.

Our proposed system has various advantages over existing OSN data collection efforts. To summarize the key differences, our solution

- supports multi-threaded data collection with consistency guarantees,
- is resilient against access restrictions imposed by the OSN service provider (i.e., limits on the number of visited profiles, list of friends, post or shares as well as blocking of the Facebook account that is utilized),
- collects each and every bit of publicly shared data (e.g., profile items, friendship links, emoticons, wall shares, posts),
- respects bread-first traversal order from the seed account,
- is independent of the frequent API updates pushed by the OSN service provider, and
- respects individual privacy by limiting collected data to publicly visible shares.

We utilize the Facebook OSN to exemplify the advantages of our solution in collecting OSN data. Over a period of roughly 2 months, we completely profiled 20K accounts, disclosing over 2.35M users, 2.7M posts and comments. As summarized in Table 1 of Section 2, these figures are better than almost all prominent work on OSN data collection despite the increased efforts of OSN service providers towards banning/blocking crawler accounts over the years.

The primary contributions of this study can be summarized as follows.

- We provide an algorithmic and systemic solution to the OSN data collection problem that respects individual privacy of the OSN users.
- We discuss the particular challenges imposed by the service providers and our solutions to overcome these challenges.
- We present possible alternatives in the system design and justify the choices we have made in our solution.
- We analyze the crawled data in order to summarize the sharing behavior of Turkish Facebook users.
- Our analyses contain indicators on the amount of accounts to be visited towards the collection of a particular number of posts, links, etc.
- After proper privacy precautions are taken (e.g.,

anonymization), we plan to share the collected data set with researchers.

In the rest of the paper, Section 2 reviews existing work on OSN data collection. Section 3 presents our proposed methodology towards modelling and collection of public OSN data over the Facebook OSN case. Section 4 outlines the OSN data analysis results. Finally, Section 5 provides our conclusions and possible directions for future work.

2. Literature Review

Obtaining high-quality OSN data is a challenging task that has been faced by many researchers before. Due to its high world-wide popularity, Facebook has been one of the most heavily studied OSNs [14]. In this section, we present our review of literature with a focus on studies collecting data by using API approach and HTTP approach. We would like to note that these studies were compiled through careful analysis of prominent journals with the keywords *"online social networks"*, *"crawler"*, and *"data collection"*.

2.1. Studies which use the API Approach

This is the most common way to collect content from OSNs [1] in which researchers have to tackle with restrictions of related OSN API. Previous studies which are using this approach are summarized as follows: Abdesslem et al. introduced a methodology to collect more reliable data on sharing behavior of OSN users [1]. Mfenyana et al. implemented a focused Facebook crawler using RestFB (i.e., Facebook Graph API client written in Java) and Jackson libraries that are written in Java. They proposed a system prototype to perform opinion monitoring and trend analysis on Facebook [6]. Motamedi et al. implemented three independent crawlers for most popular OSNs (i.e., Facebook, Twitter, and Google+) to analyze and compare users' popularity and activity which can provide insight to their users in order to decide which OSN to use [20].

Mittal and Sahu, collected 70K tweets of their pick of famous accounts from Twitter. They performed language detection experiments on the collected data [23]. Chen et al. collected posts and interactions of 859 users with the help of Facebook Graph API [24]. Passaro et al. crawled Facebook pages of the most popular newspapers in Italy to create a corpus that can be used for sentiment analysis and emoticon detection [25]. Terrana et al. used a Facebook crawler to analyze user relationships based on sentiment classification of user generated contents such as posts, comments, and likes [10]. Kastrati et al. used Facebook Graph API to crawl only posts, feeds, and comments of a user. The purpose was to discover social criminal activity [21]. Kridalukmana crawled Facebook with the Graph API and transformed data into attributed graph to illustrate the object (i.e., user, page or photograph) interrelations [26]. Siwag et al. proposed a crawler architecture for Facebook and clustered crawled data by several criteria such as gender, location, and hometown [9]. Terrana et al., crawled different OSNs for the purpose of the comparison of user profiles based on textual data (e.g., posts, comments) posted by users [11]. Their crawler has a separate module for each distinct OSN where the Facebook crawler module uses Facebook Graph API.

2.2. Studies which use the HTTP Approach

This approach requires implementing own crawler that does not have permission requirements and be able to access far more accounts. Crawlers are also do not have to tackle with the challenges imposed by the APIs developed by OSN providers. Previous studies which are using this approach are

TABLE 1: Summary of research works crawling OSN(s) for different purposes.

Research Work	Crawled OSN(s)	Collected Information	Traversal Method	Crawling Based on	Year
Works that use API approach					
Abdesslem et al. [1]	Facebook	Locations and ESM answers of 866 candidates	N/A	Facebook Graph API & LocShare App.	2012
Motamedi et al. [20]	Facebook, Twitter, Google+	Wall activities (posts, comments etc.)	N/A	Facebook Graph API for Facebook	2013
Siwag et al. [9]	Facebook	Profile details & friend lists	BFS	Facebook Graph API	2014
Mfenyana et al. [6]	Facebook	Last status updates and associated comments	BFS	Facebook Graph API	2014
Terrana et al. [10]	Facebook	Posts, comments, and likes of 4 accounts	N/A	Facebook Graph API	2014
Kastrati et al. [21]	Facebook	198 posts of 20 users	N/A	Facebook Graph API	2015
Terrana et al. [11]	Facebook, Twitter	Posts, comments, and likes of 4 accounts	N/A	Facebook Graph API for Facebook	2015
Kridalukmana [26]	Facebook	Public profile information	N/A	Facebook Graph API	2015
Passaro et al. [25]	Facebook	Timeline activities of 8 newspapers	N/A	Facebook Graph API	2016
Chen et al. [24]	Facebook	Posts and interactions of 859 users	N/A	Facebook Graph API	2016
Mittal and Sahu [23]	Twitter	70K tweets of selected accounts	N/A	Twitter API	2017
Works that use HTTP approach					
Viswanath et al. [2]	Facebook	Friendship links & timeline activities of 90,269 users	BFS	Web crawler (uses downloaded HTML documents)	2009
Catanese et al. [22]	Facebook	Friend lists of 63.4K users	BFS & Uniform sampling	Web crawler (uses HTTP requests)	2011
Xiao et al. [8]	Facebook	Friendship links of 262,526 users	BFS	Web crawler (uses interaction simulation)	2012
Flores et al. [27]	Facebook	Posts and comments of 1K users	N/A	Web crawler (uses Selenium API)	2013
Wong et al. [14]	Facebook	Friendship links of 156,297 users	BFS	Web crawler (uses headless browser)	2014
Wani et al. [3]	Facebook	Profile features and wall activities of 10K users	BFS	Web crawler (uses iMacros browser extension)	2018
Abid et al. [28]	Facebook	Friends, liked pages etc. of 21,562 accounts at total	Random Walk	Web crawler (uses Selenium API)	2018
Ours	Facebook	Profile features and all text-based public data of 20K users	BFS	Web crawler (uses Selenium API)	2020

summarized as follows: Catanese et al. implemented a Facebook crawler using two different sampling methods, namely, BFS sampling and uniform sampling [22]. This work was the first to point to the OSN provider limitation that allows obtaining at most 400 friends of a user. Xiao et al. later designed and implemented a Facebook crawler based on interaction simulation that overcame this difficulty and was able to retrieve the complete friends list of a user [8]. Xiao et al. collected a total of 262,526 users. Viswanath et al. crawled 90,269 Facebook users' information to evaluate the activity between users from the same regional network [2]. Their two-step crawler starts from a single user and visits

all friends of the user according to Breadth First Search (BFS) traversal order. Wong et al. designed a Facebook crawler with a headless browser, modeled the collected data as a graph, visualized the graph and analyzed the graph [14]. Flores et al. implemented a Facebook crawler with the Selenium API and crawled public data [27]. This crawler was seeded with the accounts of two politicians and profiled 500 friends of these two accounts and then their second level friends. The collected data was analyzed with text mining techniques to investigate how information and influence are spread through an OSN for election purposes. Abid et al. implemented a Facebook crawler with the help of the Selenium API to explore the social network around a user according to a user-user distance measure. They proposed a tool, namely, SONSAI that predicts a user's sensitive attributes [28]. Wani et al. designed a Facebook crawler, namely, IMcrawler and obtained the data of around 10K user profiles from four metropolitan cities of India [3]. In their work, they performed behavioral analysis of users based on both profile features and wall activities.

2.3. Comparison of the API and HTTP Approaches

Even though different studies employ various different techniques (i.e., collecting manually, making offline or online surveys, developing third-party applications etc.) to collect OSN data, researchers often prefer to use OSN API(s) or design their own crawlers to collect data depending on the amount and their purpose of use as summarized in Table 1. As seen from the table, studies using API approach often have permission requirements and limited access to OSN data.

Studies using HTTP approach, on the other hand, are capable of crawling far more accounts and do not have to overcome challenges of OSN APIs.

Fig. 1: A sample Facebook post with relative comments and folded replies.

However, these studies often use a crawler [2], [3], [27], [28] that focuses on specific location (e.g., same regional network), user (e.g., a politician) or group to collect friendship links together with desired other data. Even though majority of them crawled far more nodes and edges [2], [8], [14], [22], we crawled more content-rich data that includes profile attributes, friendship link, private relationships, kinships, wall activities (i.e., posts, comments, replies), likes, meta-data information of activities, and so on.

Notice that, in this paper, we preferred to implement our own crawler using the HTTP approach to reach our goals.

3. Data Modelling and Collection

We present our data model in Section 3.1, data collection algorithms in Section 3.2, and implementation details in Section 3.3.

3.1. Data model

As of writing this paper, a Facebook user has five sections in his/her account including *Timeline (i.e., wall)*, *About*, *Friends*, *Photos*, and *More*. In this

paper, we collect friendship and all text-based public data from *Timeline*, *About*, and *Friends* pages. *Timeline* contains all posts of both account owner and shared posts of his/her friends and other sources such as liked pages. A Facebook post has a recursive structure in which comments (i.e., a message posted under a post) and replies (i.e., a message posted under a comment) are placed at bottom of each other like a stair.

An example Facebook post with unfolded comments and replies is depicted in Figure 1. *About* page includes basic attributes (i.e., gender, date of birth etc.) and some other information about the user such as important events, family members, private relationships and so on. *Friends* page, on the other hand, provides a detailed list of friends of the account owner.

As seen from the Figure 1, Facebook posts have a complex structure, especially if there are many unfolded comments or replies under the related post. This makes it much more challenging to extract information from *Timeline* page compared to the *About* and *Friends* pages. A detailed description of the challenges faced during the data collection is given in Section 3.3.

Considering the information on these three pages, we created our data model as a star schema with the user relation placed at the very center. The schema contains 20 tables in total. We consider each data object on a Facebook account as an entity in this data model, but we use some extra (i.e., helper) relations to store the data in a meaningful and lossless way.

We decomposed our tables to eliminate data redundancy and achieve data integrity as much as possible. There is still some redundancy contained in our model. For example, users may write their education and work information with typos and in different formats which makes it too hard to group same information. For instance, users who are studying in the department of *Computer Engineering* (*"Bilgisayar Mühendisliği"* in Turkish) disclose this information in different ways, such as by typing *"bil. muh"*, *"bilgisayar müh."*, *"comp. eng."*, and so on. We left these cases of redundancy untouched in order to reduce the number of table joins during the analysis phase and improve the performance. This is a common practice in relational database design and is called, de-normalization.

Facebook assigns a unique *username* to each account holder. We utilize this value to identify references (i.e., foreign keys) to a user or a user's account. Friendship is a directionless relationship in the Facebook OSN. That is, if user U_A is friends with user U_B, then U_B is also friends with U_A. Yet, our data model maintains direction of such links in case that a future analysis might require explaining how a user account has been discovered. After all, the link (U_A, U_B) implies the existence of the link (U_B, U_A) and such missing links can always be added afterwards.

Notice that our data model is very comprehensive and the only bits of data that are missing are multimedia files. We would like to emphasize that multimedia files are indeed within access of our crawler and these files were excluded only to improve the Database Management System (DBMS) performance.

We use MySQL Workbench DBMS to design and implement our data model. The corresponding star schema of the designed relational database is given in Appendix A.

3.2. Data collection

The general architecture of our 3-tiered system design for the data collection phase is depicted in Figure 2. The DBMS tier implements the data model

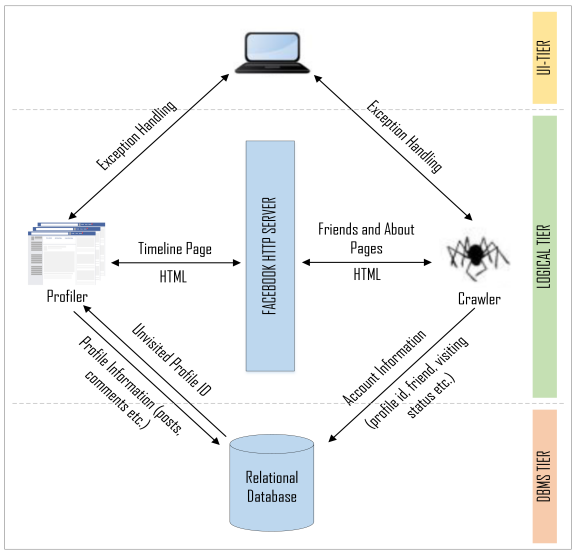

Fig. 2: Three-tiered system design.

that was presented in Section 3.1. The logical tier will be detailed below. The user interface (i.e., UI) tier allows us to follow the progress of the logical tier and handle any exceptional cases manually.

The logical tier contains two important components: the *crawler* and the *profiler*. The *crawler* is responsible for discovering accounts that are publicly reachable from the seed account s. The *profiler* visits profile pages of these discovered accounts and collects all relevant, publicly shared data. We prefer a breadth-first traversal (BFS) [22] of the Facebook OSN to ensure that every account is visited only once, and give priority to discovered accounts that are closer to s. That is, the *crawler* first inserts the root account s in the database, then continues with the direct friends of s, friends of friends and so on. At each account u, the *crawler* checks whether the friends list of u is public. If so, all discovered-but-not-yet-visited friends of u are inserted into a first-in, first- out (FIFO) queue. This BFS implementation stops only when all publicly reachable in/direct friends of s are marked as visited. The BFS nature of the *crawler* generate a visit-order for the *profiler*. When visiting the next account u, the *profiler*'s aim is to record all publicly shared content in the corresponding OSN account.

We next present our algorithms towards the implementation of the *crawler* and *profiler* components. We employ a single threaded solution to the *crawler*, and a multi-threaded solution to the *profiler*. With these two design choices, our logical tier solves an instance of the well-known "single producer, multiple consumers" problem [29]. The concurrency problems caused by the race condition between these threads are resolved based on the transaction management system of the data tier (i.e., the underlying relational DBMS).

Crawler module: The *crawler* relies on a helper relation which is named as $Crwlr$. This relation contains 4 simple fields: (1) the Facebook ID of an account $username$ (or fid) - a primary key, (2) a $boolean$ flag $all\_visited$ indicating whether this account's friends list has been inserted into the table, (3) a $boolean$ flag $tl\_visited$ indicating whether this account's profile page has been processed, and (4) a field $depth$ indicating how far the current account is from the root user.

Our *crawler*'s pseudo-code is given in Alg. 1. The *crawler* creates the $Crwlr$ relation, inserts the fid representing the seed account and visits its friends list in the OSN (line 3). Then, the *crawler* begins an infinite loop. At each iteration, the fid of an unvisitied user from the current level/depth is retrieved. This user is identified by vid. At line 8, all friends of vid are retrieved from the OSN and these friends are inserted with an incremented $depth$ into $Crwlr$ at line 11. We assume that $vid.frnds$ will be $null$ if the friends list is not public. At line 13, vid will have been visited by the *crawler*, therefore the table is updated accordingly.

Notice that the entire operation of visiting a user's friends list is enclosed packed into a database transaction (lines 6, 14). This precaution ensures

consistency under possible failure scenarios. The *crawler* can be stopped and restarted arbitrarily.

Algorithm 1 Crawler module

Require: The seed user's fid (i.e., $username$)
Ensure: Traversal of the seed's in/direct friends in BFS order
1: **if** Relation $Crwlr$ does not exist **then**
2: Create an empty relation with schema $Crwlr(fid, all\_visited, tl\_visited, depth)$
3: Insert $(fid, false, false, 0)$ into $Crwlr$
4: **end if**
5: **loop**
6: Begin transaction
7: $vid \leftarrow fid$ of an unvisited user with $min(depth)$
8: $vid.frnds \leftarrow getFriends(vid)$
9: $depth = depth + 1$
10: **for all** $w \in vid.frnds$ **do**
11: Insert $(w, false, false, depth)$ into $Crwlr$
12: **end for**
13: Set $all\_visited$ for vid
14: End transaction
15: **end loop**

Profiler module: The *profiler* module supports multi-threading and the threads are controlled with a master-slave organization. The pseudo-code of the *profiler*'s master thread is given in Alg. 2. The master thread first creates the data model described in Section 3.1 that will store the public profile information. Then it invokes as many workers as necessary within an infinite loop (lines 4-6).

Alg. 3 provides the pseudo-code for a worker thread of the *profiler*. Upon invocation, basic attributes (e.g., gender, phone, email) and wall activities of an unprofiled user are extracted. Then these extracted information is stored in the schema. The final step marks the related account as profiled (line 6). Notice that we assume that profile information

Algorithm 2 Profiler-MasterThread

Require: Crawler populating the $Crwlr$ relation
Ensure: Traversal of discovered accounts in BFS order
1: **if** Data model is not implemented **then**
2: Create the schema for storing the public data of discovered profiles
3: **end if**
4: **loop**
5: $vid \leftarrow fid$ of a user whose $tl\_visited$ is $false$
6: Fork into $Profiler - WorkerThread(vid)$ {*a thread pool controls all live workers*}
7: **end loop**

of vid is null if its wall is not public and entire operation of a worker is enclosed within a database transaction (lines 1, 7) to ensure resilience against failures.

Algorithm 3 Profiler-WorkerThread

Require: $\exists vid \leftarrow fid$ of a non-profiled user
Ensure: Inserts all public data within the profile page of vid
1: Begin transaction
2: Extract basic info: gender, phone, interest etc.
3: Extract places, work, education
4: Extract family members, relationships
5: Extract wall activities (posts, comments etc.)
6: Set $tl\_visited$ for vid
7: End transaction

3.3. Implementation details

In this section, we present the details of our implementation, the challenges we faced, and the solutions developed to overcome these challenges.

Selenium API: The crawler imitates a real-world OSN user with the help of the Selenium API [30],

which is a browser automation library and is most often used for testing web applications. It may be used for any task that requires automating interaction with the browser and can help to replicate and store human actions on a web page [27].

Information extraction: The crawler uses Document Object Mapping (DOM) tree to identify the data to be crawled. DOM provides the structured representation of a web page where nodes denote the HTML tags and the tree hierarchy represents the organization of the nested elements [3]. To extract information from accounts, we locate elements in the DOM tree of each account's HTML source. We perform this task using the XPath query language which is used for locating nodes in an XML document. As HTML can be an implementation of XML (XHTML), we use XPath selectors (e.g., //title[@lang='en'] selects all the title elements that have a "lang" attribute with a value of "en") to locate elements in HTML source of any page returned by Facebook's HTTP Server.

Dynamic content loading & browser interaction: Facebook uses dynamic content loading to show most of the data including friends list, comments and replies under a post, posts in previous and current year(s). For instance, in *Friends* page, it initially loads 20 and shows at most 100 friends at a time as the user scrolls the page down. As an another example, if there are 3 or more messages under a post or comment, Facebook folds them and does not show all of them at once as depicted in Figure 1. This is also true for content of *Timeline* page. In such a case, someone who wants to see the data has to click on the *"show more"* link added automatically by the Facebook. Notice that the user/crawler may have to do this one or more times to see all relevant data.

This complicates the data collection task as the information is not available as a whole in the HTML source code of a web page [3], [14]. As stated above, in Facebook, dynamic content on the web page needs to be triggered by user interactions with the page. Therefore, our crawler has a mechanism to automate such interactions to load the dynamic content into the parent HTML content.

We overcome the dynamic content loading by executing several JavaScript functions (e.g., scrolling, link clicking etc.) through Selenium API to obtain data from updated HTML content. We also use some variables to control the completion of these tasks. For instance, we use a `boolean` variable to stop scrolling on friends page of a user. Initially, its value is false, but if an image element (that is hidden in HTML content, but only visible while loading the rest of the list, if exists) in friends page is unvisible, it takes `true` value and we stop the page scrolling. This means that all of the friends of the related user are loaded in that page.

Connection pooling: Connection pooling is a mechanism to create and maintain a collection of JDBC (i.e., Java Database Connectivity) connection objects. We implemented a connection pool to ensure concurrent access of multiple threads that simultaneously access our database. In this way, we provide each thread the ability to borrow a separate connection from the pool, use it, then return it to the pool by closing it. This mechanism also improved the performance of our crawler, as it allows reusability of the pool of connection object.

Challenges faced: Using a self-designed tool (i.e., the crawler) that does not require permission-based access to user data allows us to access far more accounts than an API-based solution. In addition, this approach also facilitates by-passing the limitations imposed by the OSN service providers through the APIs. Although favorable in these aspects, this approach has its own challenges.

The following is a list of the important challenges

TABLE 2: Quantitative description of crawled public Facebook data.

Attribute (Total # of)	Count
Visited users	20,000
Users whose first neighborhoods are visited	261
Friends	3,980,270
Unique friends	2,350,454
Filtered users (user location outside Turkey)	1,079
Posts	2,762,023
Comments	2,743,513
Replies	466,995
Discovered family members	7,581
Discovered private relationships	4,660
Crawling period: 20/12/18-28/02/19	

that we had to tackle during the data collection:

- To access dynamic content, we execute some JavaScript functions which enable our crawler to interact the browser. Unfortunately, the execution of JavaScript has negative effects as well, because it takes time and slows our crawler. To overcome these effects, we speed-up our crawler by using multi-threaded implementation which needs concurrency control between different threads.
- Facebook has run a dynamic HTML template in the background and the attributes (e.g., id, class, etc.) of the HTML elements (e.g., div, img, etc.) are updated regularly. Therefore, we developed a control mechanism that detects such updates and warns the data collector through UI layer. The warnings state that the `XPath` selectors should be updated accordingly.
- We use different time delays within the *crawler* and *profiler* modules to avoid certain exceptions. For example, Firefox browser throws a binding exception if more than 5 processes are executed concurrently.
- Facebook restricts the number of user profiles a single account or IP address can access in a given time interval. To give an example, Facebook thinks that a user may be a computer if the corresponding IP address frequently accesses accounts that are not in friends list of neither itself nor its friends. When the predefined threshold is exceeded, Facebook temporarily blocks the account with which the *crawler* or *profiler* logs in. After an account is blocked, the account holder should verify her identity through a phone number or a personal photograph. To avoid being blocked, we used multiple accounts to connect to Facebook in short time intervals and with fewer threads per process.
- Our crawler can not establish a consistent interaction with the browser and this either interrupts (e.g., crawler can not login until the login page is loaded completely) or slows down (e.g., page and dynamic content loading take too much time) the data collection process.
- Facebook has a rather complex HTML structure and this makes our crawler's maintenance difficult.

4. Analysis Results of Collected Data

In this section, we perform analyses of what kind of information users tend to share about themselves on Facebook and what type of content they mostly post on their timelines. To reach this goal, we collected public Facebook data by running our crawler over the Firefox browser. We then analyzed this public OSN data using aggregate statistical queries of relational and graph DBMSs. Our analysis results have been visualized with the JFreeChart [31] library developed for use with the Java programming language.

During the crawling period from December 20th to February 28th, we collected account information and timeline activities (i.e., posts, comments, and replies) of 20K users. Note that we only crawled textual data from user accounts. Multimedia files

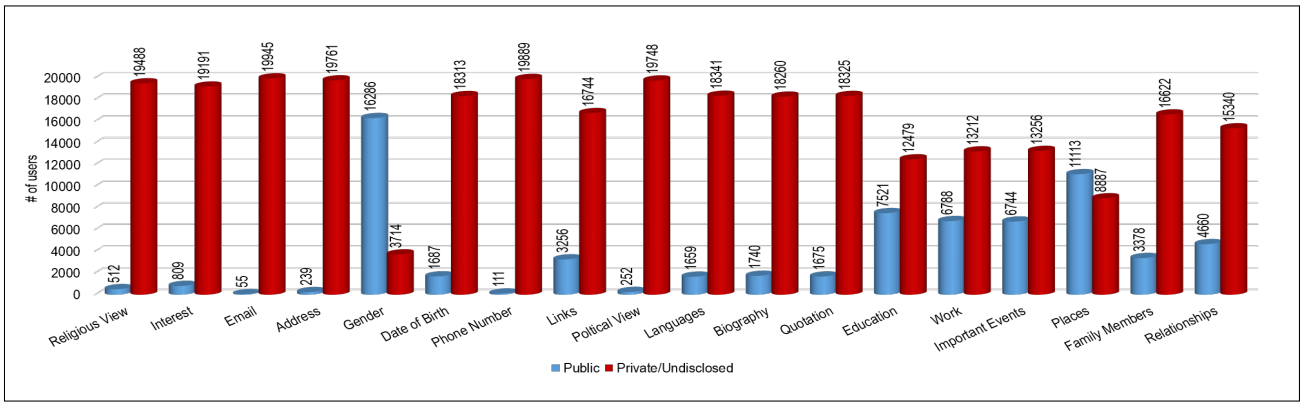

Fig. 3: Publicity ratios of revealed user attributes.

were excluded only to keep the relational database compact and not due to an inherent incapability of our data collection methodology.

We present the quantitative description of our collected data (i.e., the public sub-network) in Table 2. As seen from the table, we reached more than 2.35M unique accounts. We manually filtered 1,079 users who are not from Turkey by taking several parameters into consideration such as language (e.g., user's posts are not written in Turkish), real name, and place lived-in.

Note that among these 20K users, we only visited the first neighborhoods of 261 users. That is, we were able to discover 2.35M accounts by visiting 20K accounts, but there are still 19,739 users whose all friends have not been visited yet. Upon completion of data collection, we performed detailed statistical analyses of the data in terms of *page publicity*, *basic attributes*, *friendship and relations*, and *timeline activities*.

Following sub-sections present the most interesting subset of our analysis results, but all of the generated charts and graphs are accessible on the web page (https://ocbn.bitbucket.io/web/index.html) of our project.

TABLE 3: Distribution of users based on publicity of their *friends* and *wall* pages.

Page	Private	Public	Partially Public
Wall	1,421	18,579	NA
Friends	11,675	8,109	216

4.1. Publicity of Pages

Privacy-aware Facebook users are able to keep their *Timeline* (i.e., wall) and *Friends* pages private. Table 3 presents the numbers of users who make their pages public, private or partially public in the crawled portion of the OSN. As seen from the table, users mostly tend to hide their friend lists but, the same is not true for wall activities and only 7% (i.e., 1,421) of users hide their *timeline* pages, whereas 58% (i.e., 11,675) of them completely hide *friends* pages. Note that *partially hidden* means that one can only see friends in common with the account owner.

4.2. Basic Attributes

At sign-up, Facebook forces users to fill in certain profile information (e.g., username, e-mail), while other profile attributes are often disclosed willingly by the users to polish their profiles. Figure 3 shows

TABLE 4: Age-gender distribution of users who disclose gender and birth-date information.

Gender	Age-range						Total
	11-20	21-30	31-40	41-50	51-60	61+	
Male	13	107	216	92	38	12	478
Female	6	108	131	31	14	3	293
NA	1	30	58	10	5	4	108
Total	20	245	405	133	57	19	**879**

the summary of publicity ratios for basic attributes among users in our crawled snapshot.

As seen from the figure, *e-mail* and *phone number* are the least frequently shared attributes, while the most frequently shared attributes are *gender* and *places*. Only 18% (i.e., 3,714) of users do not share their *gender*, 49% of all users are *male* and 32% are *female*. 47% state that they have undergraduate education and are interested in females.

English and Turkish are mostly disclosed languages with the rates of 39% and 33% respectively. Among the users who reveal their religious view, 92% express themselves as Muslims. In addition, users mostly share their Instagram accounts on their Facebook profiles to stay connected on other OSNs and *phone* and *address* attributes are usually disclosed by accounts (e.g., a local company) which are used for business purposes. 1,687 users disclose their date of birth, but only 879 of them give this information in dd-mm-yyyy format. For those users, using birth date information, we also calculated the ages of the users.

Table 4 provides the age-gender distribution. Majority of users are at age-range of 31 to 40 years. An important point to mention here is that date of birth attribute is one of the most important personal identifier which may be used to impersonate the target user over long distance service (e.g., internet or phone) channels. In addition, our simple analysis on usernames showed that 20% of users select a username that is completely composed of numeric values (i.e., *"123456789"*), while the rest prefer to use a username with any string (i.e., *"jack.sparrow.25"*). Our analysis of users' places makes it clear that they mostly live in big cities (e.g., Istanbul, Bursa, Ankara) and those living in these cities mostly come from another city.

4.3. Friendship & Relations

Users share their family members and private relations as well as friendships. When we explore our data, we find out that users who have a private *relationship* mostly state that type of this relationship is marriage. As seen from Figure 4, on the other hand, the most shared *family membership* (or kinship) types are cousin and brother. Taking a closer look into friendship links also shows that the average number of friends for *male* users is 276, while this value is 137 for *female* users. The users who do not disclose gender information have an average of 102 friends. Most importantly, 194 users reveal their Mothers (see Figure 4) and this case carries potential risk of being targeted by an adversarial attack that aims to infer/learn mother's maiden name. Notice that if such an attack is successful it may cause to serious privacy and security risks such as identity theft, fraud, and cyber stalking.

4.4. Wall Activities

We explore wall activities of users regarding both basic attributes and meta-data (i.e., time of posting, via, place etc.) of activities. According to our basic analysis, users prefer the *like* reaction with 98% to express their feelings about a content. They often share more content in the winter months (i.e., most in January) and the most common time of account inactivity is between 02:00 hours and 06:00 hours.

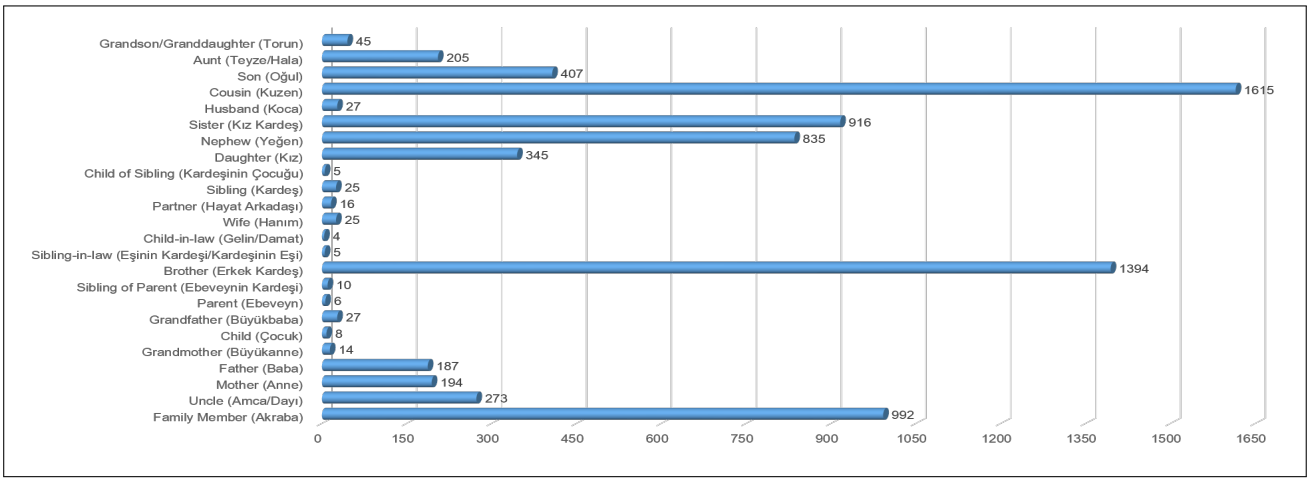

Fig. 4: Total numbers of disclosed family members with respect to their kinship types.

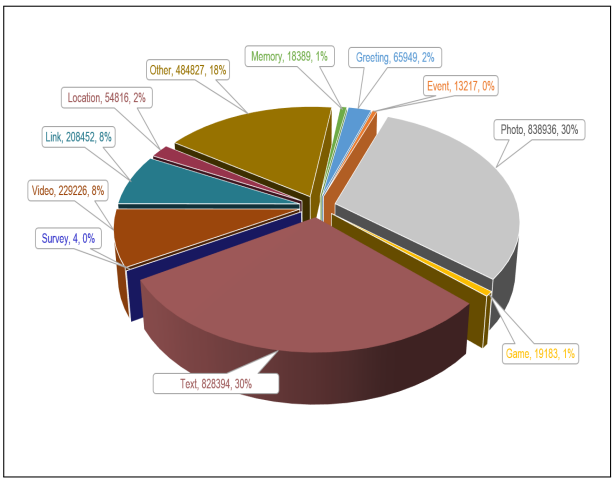

Fig. 5: Distribution of the number of *posts* in different types.

Important events are shown as a post in the wall page of a user. Considering events as posts, our analysis reveals that the most frequently shared event types are *starting a work* and *graduation* among users. We also found that *male* users have more timeline activity than *female* users. Using titles and contents of posts, on the other hand, we categorized posts as depicted in Figure 5, which shows that users mostly post contents which include raw text or any photograph.

5. Conclusion and Future Work

OSNs provide privacy settings such that a user gets to control which piece of data is shared with whom. However, many OSN users do not sufficiently leverage these settings and share public information. The arguments that they have nothing to hide or they do not understand how the privacy settings really work are quite common. Such behavior threatens the individual privacy and security of both the public sharers and those users that are connected to them on some OSN.

The main reason behind this behavior is due to the users often want to increase their digital existence by providing more information about themselves. Even though this is a risky matter, data provided by users make OSNs a huge and rich source of public data. As such, analyzing OSNs may produce interesting findings and useful knowledge. However, researchers face with different challenges to access the data even there are APIs provided by OSN service providers.

In this paper, we designed and implemented a crawler that uses HTTP requests and responses to reveal public data on Facebook. We analysed the collected data and investigated the content-sharing

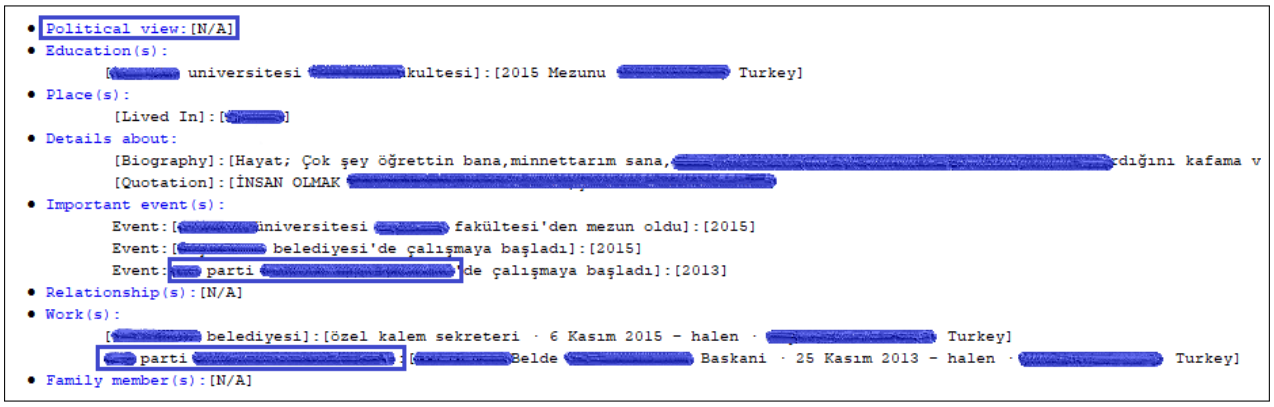

Fig. 6: Screenshot of our crawler for an arbitrary user's basic information (not all). He/she does not disclose his/her *political view*, but one can infer about this attribute by taking its *work* and *important event* attributes into consideration.

behaviors and awareness of personal privacy of users in Turkey. Based on our analysis results, we conclude that users generally do not tend to disclose their sensitive attributes either consciously or unconsciously. However, sharing rates of some attributes such as birth-date, family members, relationships, place, and work are too high to be underestimated.

OSN users share information easily for any number of reasons and differences behind these reasons are not clear. Learning and social engagement, on the other hand, are found to be encouraging motivations for users to share information on OSNs [32], [33]. Moreover, characteristics and features of the OSN could also be one of the motivating reasons in sharing information [32]. Therefore, it is challenging to detect main reasons behind why sharing rates of these attributes are higher when compared to the others. Nevertheless, we think that users are not aware of the privacy risks that may arise when they disclose such attributes. For instance, users often keep their political view private, but many of those disclose where they live. In addition, users may be motivated themselves to share information for reasons such as pleasure, sacrifice, social participation and reciprocity.

An adversary that gains access to public data due to insensible use of OSNs may infer (see Figure 6) with high probability private information about a target OSN user that he/she would prefer to keep private and violate individual privacy. Such attacks may reveal the user's religious belief, race, political opinion, residence address among many others. For instance, if friends of a user post on his/her timeline to celebrate his/her birthday, one can learn this sensitive attribute about user by taking the dates of related post into consideration. If the attacker obtains personal identifiers such as mother's maiden name or exact date of birth of an individual, he/she may use these data to impersonate the target individual over long distance service channels (e.g., internet or phone). This threat also brings together security risks. Primary cases of concern are identity theft, fraud, and (cyber) stalking.

As a future work, we plan to transfer our data into a graph database so as to perform attribute disclosure and privacy and security analysis. We will also try to extract *online social footprint* for users who disclose their other OSN accounts (e.g., Instagram) using profile matching techniques.

References

[1] F. Abdesslem, I. Parris, and T. Henderson. *Reliable online social network data collection*. London: Springer, 2012, Ch.8.

[2] B. Viswanath, A. Mislove, M. Cha, and K. Gummadi. "On the evolution of user interaction in facebook", *The 2nd ACM workshop on online social networks*, Barcelona, Spain, pp. 37-42, 16-21 August 2009.

[3] M. Wani, N. Agarwal, S. Jabin, and S. Hussai. "Design and Implementation of iMacros-based Data Crawler for Behavioral Analysis of Facebook Users", *Computer Science: Social and Information Networks*, February 2018.

[4] "Facebook Reports Fourth Quarter and Full Year 2019 Results", https://investor.fb.com/investor-news/default.aspx, accessed: 2020-01-03.

[5] "The top 500 sites on the web", https://www.alexa.com/topsites, accessed: 2020-01-03.

[6] S. Mfenyana, N. Moorosi, and M. Thinyane. "Facebook Crawler Architecture for Opinion Monitoring and Trend Analysis Purposes", *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, Port Elizabeth, Eastern Cape, South Africa, 1-3 September 2014.

[7] Y. Modi, and I. Gandhi. "Internet sociology: Impact of Facebook addiction on the lifestyle and other recreational activities of the Indian youth", *SHS Web of Conferences*, Jacarta, Indonesia, pp. 1-4, 14-16 October 2013.

[8] Z. Xiao, B. Liu, H. Hu, and T. Zhang. "Design and implementation of facebook crawler based on interaction simulation", *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Liverpool, England, pp. 1109-1112, 25-27 June 2012.

[9] T. Siwag, P. Sirohi, and N. Singhal. "Novel Architecture of a Focused Crawler For Social Websites", *International Journal of Computer Engineering and Applications*, Vol.7, No.3, pp. 132-144, September 2014.

[10] D. Terrana, A. Augello, and G. Pilato. "Facebook users relationships analysis based on sentiment classification", *IEEE International Conference on Semantic Computing (ICSC)*, Newport Beach, CA, USA, pp. 290-296, 16-18 June 2014.

[11] D. Terrana, A. Augello, and G. Pilato. "A system for analysis and comparison of social network profiles", *IEEE International Conference on Semantic Computing (ICSC)*, Anaheim, CA, USA, pp. 109-115, 7-9 February 2015.

[12] M. Conti, R. Poovendran, and M. Secchiero. "Fakebook: Detecting fake profiles in on-line social networks", *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Istanbul, Turkey, pp. 1071-1078, 26-29 August 2012.

[13] B. Jansen, K. Sobel, and G. Cook. "Classifying ecommerce information sharing behaviour by youths on social networking sites", *Journal of Information Science*, Vol.37, No.2, pp. 120-136, April 2011.

[14] C. Wong, K. Wong, K. Ng, W. Fan, and K. Yeung. "Design of a crawler for online social networks analysis", *WSEAS Transactions on Communications*, Vol.13, pp. 264–274, 2014.

[15] "SNAP Datasets: Stanford large network dataset collection", http://snap.stanford.edu/data, accessed: 2020-12-12.

[16] "EU General Data Protection Regulation (GDPR)", https://www.ingramflyhigher.com/assets/2018/gdpr/img/ingrammicro-gdpr-1pp.pdf, accessed: 2020-04-01.

[17] "Personal Data Protection Law in Turkey", https://www.kvkk.gov.tr, accessed: 2020-04-01.

[18] "Kişisel Verileri Koruma Kurumu". https://www.kvkk.gov.tr/, 2020-04-01.

[19] A. Gamboa, and H. Gonçalves. "Customer loyalty through social networks: Lessons from Zara on Facebook", *Business Horizons*, Vol.57, No.6, pp. 709-717, November–December 2014.

[20] R. Motamedi, R. Gonzalez, R. Farahbakhsh, A. Cuevas, R. Cuevas, and R. Rejaie. What osn should i use? characterizing user engagement in major osns. *Technical report*. University of Madrid. http://www.it.uc3m.es/ rgonza1/pubs/whatOSN.pdf, 2013.

[21] Z. Kastrati, A. Imran, S. Yildirim-Yayilgan, and F. Dalipi. "Analysis of Online Social Networks Posts to Investigate Suspects Using SEMCON", *International Conference on Social Computing and Social Media*, Los Angeles, CA, USA, pp. 148-157, 2-7 August 2015.

[22] S. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. "Crawling facebook for social network analysis purposes", *International conference on web intelligence, mining and semantics*, Sogndal, Norway, pp. 52, 25-27 May 2011.

[23] S. Mittal, and G. Sahu. "Twitter Crawler with Multilingual Text Classification", *International Journal of Innovations & Advancement in Computer Science (IJIACS)*, Vol.6, No.6, pp. 77-83, June 2017.

[24] H. Chen, K. Hsu, and S. Chiu. "Event Detection in an ego Network on Facebook", *Pacific Asia Conference on Information Systems (PACIS)*, Chiayi, Taiwan, pp. 172, 27 June-1 July 2016.

[25] L. Passaro, A. Bondielli, and A. Lenci. "Fb-news15: A topic-annotated facebook corpus for emotion detection and sentiment analysis", *Third Italian Conference on Computational Linguistics (CLiC-it)*, Napoli, Italy, pp. 228-232, 5-6 December 2016.

[26] R. Kridalukmana. "Generic social network data crawler using attributed graph", *2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia, pp. 138-142, 16-18 October 2015.

[27] G. Flores, A. Lorena, C. Penteado, and C. Kamienski. "Can Social Network Influence Voters?", *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, Brasilia, Brazil, pp. 3-8, 6-10 May 2013.

[28] Y. Abid, A. Imine, and M. Rusinowitch. "Sensitive attribute prediction for social networks users", *EDBT/ICDT 2018 Joint Conference*, Vienna, Austria, pp. 28-35, 26-29 March 2018.

[29] L. Higham, and J. Kawash. "Critical sections and producer/consumer queues in weak memory systems", *International*

Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'97), Taipei, Taiwan, pp. 56-63, 20 December 1997.

[30] U. Gundecha. *Selenium Testing Tools Cookbook*. Packt Publishing Ltd./Birmingham, 2012.

[31] D. Gilbert. *The jfreechart class library. Developer Guide*. Object Refinery, 2002.

[32] S. Syn, and S. Oh. "Why do social network site users share information on Facebook and Twitter?", *Journal of Information Science*, Vol.41, No.5, pp. 553-569, May 2015.

[33] K. Hew, and N. Hara. "Knowledge sharing in online environments: A qualitative case study", *Journal of the American Society for Information Science and Technology*, Vol.58, No.14, pp. 2310-2324, December 2007.

Appendices

Appendix A. Star schema of the our relational database design is depicted in Figure 7.

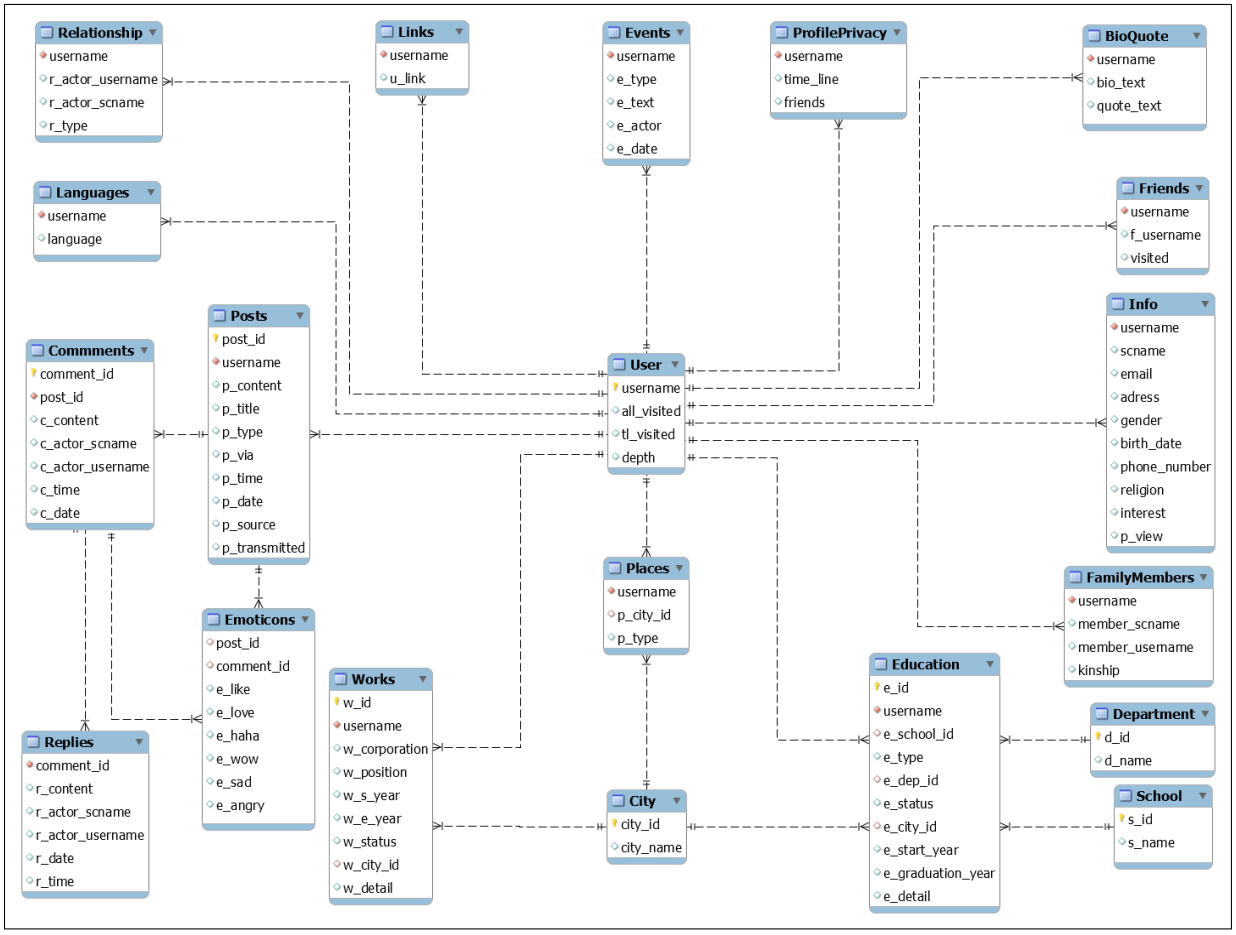

Fig. 7: Star schema of our relational database