MICROBLOG ANALYZER AGGREGATE ESTIMATION OVER A MICRO BLOG PLATFROM

by

SATISHKUMAR MASILAMANI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment of the Requirements for the

Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

Acknowledgements

I extend my heartfelt gratitude to Dr. Gautam Das, Advisor, for providing me an opportunity to work on this project and also for his guidance and support. It was a privilege being a part of the DBXLab which provided an excellent environment for completing my research project. I am also grateful to Dr. Ramez Elmasri and Mr. David Levine for serving on my thesis committee.

I am grateful to Mr. Saravanan Thirumuruganathan for guiding me throughout this thesis work, and I feel privileged for having worked with him for the past one year.

“முகநக நட்பது நட்பன்று நெஞ்சத், தகநக நட்பது நட்பு” by thiruvalluvar , The love that dwells merely in the smiles of the face is not friendship; (but) that which dwells deep in the smiles of the heart is true friendship. Thanks to all my friends and well wishers, a special thanks to my friend Geetha Lakshmi for her guidance and constant encouragement during the course of graduate studies. This dissertation would not have been complete without your support and assistance.

Finally and importantly I Wish to thank my Parents and Brother for their patience, support, assistance and faith in me. I am deeply indebted to my family for their persistent support and encouragement, throughout my academic carrer. My deepest appreciation is expressed to them for their love, understanding and inspiration, and I sincerely believe that without their blessings and encouragement, this would not have been as successful as it has been. I cannot thank my brother enough for showing me what it is to be a free spirit, walking with me and guiding me through out my life.

April 17, 2015

Abstract

MICROBLOG ANALYZER AGGREGATE ESTIMATION OVER

A MICRO BLOG PLATFORM

SATISHKUMAR MASILAMANI, M.S

The University of Texas at Arlington, 2015

Supervising Professor: Gautam Das

Microblogging is a new mode of communication in which users can share their current status in brief and agile way in the form of text, image, video etc. over smart phones, email or web. Recently, Micro blogs such as Twitter, Tumblr, Google+ have experience phenomenal growth and are regularly used by millions of users. The data from microblogs is very useful for researchers to analyze various facets such as user behaviors, user intentions (like daily chatter, conversations, sharing information and reporting news), microblog social network structure etc. For example, a sociologist might want to use the microblog postings to analyze the popular opinion about a particular topic. However, existing approach to facilitate such analytics has certain limitations due to the various restrictions imposed by microblogs. Restrictions include API rate limits(that restricts the amount of queries issued in a day) or other limits (Twitter search API only provides results for last week and so on).

In this thesis, we build an efficient microblog analytics platform MICROBLOG-ANALYZER to enable the approximate estimation of aggregate queries over an online microblogging service. MICROBLOG-ANALYZER works by leveraging user timeline access offered by online microblogs. It dynamically constructs a level-by-level sub-graph of the microblog and performs sampling by a novel topology aware random walk. MICROBLOG-ANALYZER can handle a number of online microblogs such as Twitter, Google+, Weibo, Tumblr, Instagram etc.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

Online micro blogging has made headway towards significant popularity in recent years as they extend obvious and easy ways for online interaction over social networking websites and allow millions of users to post and blog their contents online. Besides providing engaging channels for one-to-one interaction, it also favors micro blog data analytics. The micro blogs are a vast repository of user-generated content of world events.

Micro blogging differs from the regular blogging by encouraging shorter posts, which requires minimal investment of time and thought for content generation. Microblog data analytics helps in studying the common topic of interests amongst the mass, their opinions and conversations which in comparison with the conventional survey method is substantial

The Service providers provide free public access but limited to data through restricted API's which aids the social scientists pursue research and analyze the publicly available micro blog data such as postings and conversations to arrive at various conclusions varying from the public response over a political issue to popularity of an establishment. All this and more can be possible by answering the aggregate queries over micro blog data, available to access publicly and also serves as the focus of this paper. For instance, we can consider the number of Twitter users who used the keyword '#IceBucketChallenge' in 2014, the aggregate query can be SUM, Count, Average and can be over various attributes and selection conditions. E.g.: keywords or hash tag as attribute and time as selection condition. The technique we apply yields approximate response while on contrary, exact answers are often not workable since they require access to the complete data.

Chapter 2

Technical Background

In recent years, the growth and popularity of Online Social Networks (OSN) has experienced a volatile increase which triggered the graph sampling to play a critical role in measurement and characterization studies of such OSNs. Lot of attention has been put on how to obtain a representative or unbiased dataset from a large social graph using graph-sampling techniques. An online social network allows its users to publish contents and form connections with other users. To retrieve information from a social network, one generally needs to issue an individual-user query through the social network's web interface by specifying a user of interest, and the web interface returns the contents published by the user as well as a list of other users connected with the user1. In this chapter we will focus on the terminologies and techniques used in analyzing data in micro blogging platform.

Sampling

Data sampling is a statistical analysis technique used to select, manipulate and analyze an illustrative subset of data points in order to distinguish and identify patterns in the larger data set under examination. The objective of sampling is to randomly select Elements (e.g., nodes/users or edges/relationships) from the online social network according to a pre-determined probability distribution, and then to generate aggregate estimations based on the retrieved samples.

One of the main challenges presented by the Internet surveys is the sampling procedure, as it must be reconsidered to avoid risk of bias and a lack of scientific accountability. Thus the construction of an accurate sampling is critical in analyzing the data set and using random sampling protects against bias being introduced in the sampling process, and hence, it helps in obtaining a representative sample. The two key objectives for sampling are: minimizing bias - such that the retrieved samples can be used to accurately estimate aggregate query answers. In general, sampling bias is the distance between the target distribution of samples and the actual sampling distribution - i.e., the probability for each tuple to be retrieved as a sample. Reducing the number of queries required for sampling (query cost) - given the stringent requirement often put in place by real world social networks on the number of queries one can issue per day. Consider the number of unique queries one has to issue for the sampling process, as any duplicate query can be answered from local cache without consuming the query limit enforced by the social network provider.

Random walk

A random walk is a mathematical formalization of a path that consists of a succession of random steps where at each step the location jumps to another site according to some probability distribution. The growths of OSNs have motivated a large amount of studies from research community to measure and analyze the characteristic of social graphs. However, previous studies have only focused on the unbiased sampling of undirected social graphs and they are biased towards high-degree nodes. In literature, there are two popular random walk schemes:

Simple random walk

Simple random walk (SRW) starts from an arbitrary user/node, repeatedly hops from one user/node to another by choosing uniformly at random from the former user's neighborhood, and stops after a number of steps to retrieve the last user as a sample. When the simple random walk is sufficiently long, the probability for each user to be sampled tends to reach a stationary (probability) distribution proportional to each user's degree(i.e., the number of users connected with the user). Thus, based on the retrieved samples and knowledge of such a stationary distribution, one can generate unbiased estimations of AVG aggregates (with or without selection conditions) over all users in the social network.

Metropolis Hastings random walks.

Metropolis Hastings random walk (MHRW) is a random walk achieving any distribution (typically uniform distribution) constructed by the famous MH algorithm. As an extension of MHRW, based on the knowledge of all the ids of a graph, we can conduct random jump (RJ), which jumps to any random vertex in the graph with a fixed probability In each step when it carries on the MHRW. Although MHRW can yield asymptotically uniform samples, which requires no additional processing for subsequent analysis, it is slower than SRW almost for all practical measurements of convergence, such as degree distribution distance, KS distance and mean degree error. The random walk-based sampling Metropolis-Hasting Random Walk (MHRW) is proposed to obtain samples from an undirected graph, such as Facebook. This algorithm can guarantee the un-biasedness of the sampling procedure, thus can keep all the statistical properties of undirected social graphs. However, unlike Facebook, micro blogging networks such as Twitter do not require reciprocation in the relationship: one can follow anyone without being followed.

The data that these studies use are either complete datasets from network operators, which are commonly not publicly accessible, or self-crawled datasets, which are normally incomplete or biased to high-degree nodes. Popular micro blog sites like Twitter, Tumblr, Instagram and some other social networking sites like Google+ and Facebook offer search API calls, which allow retrieving posts containing query keywords. The result of these API calls are often limited to a few thousands except for Twitter’s streaming API which allows retrieving large number of posts with a given keyword and other search conditions. The streaming API interface yet has a limitation of fetching the data in the future.

Aggregate Estimation

Online micro blog service providers not only offer a platform for users to share information with their acquaintance, but also enables a third party to perform a wide variety of analytical applications over the social network - e.g., the analysis of rumor/news propagation, the mining of sentiment/opinion on certain subjects, and social media based market research.

While some third parties, e.g., advertisers, may be able to negotiate contracts with the network owners to get access to the full underlying database, many third parties lack the resources to do so. To enable these third-party analytical applications, one must be able to accurately estimate big picture aggregates (e.g., the average age of users, the COUNT of user posts that contain a given word) over an online social network by issuing a small number of individual-user queries through the social network’s web interface.

An important challenge facing third-party aggregate estimation is the lack of cooperation from online social network providers. In particular, the information returned by each individual-user query is extremely limited - only containing information about the neighborhood of one user. Furthermore, almost all large-scale online social networks enforce limits on the number of web requests one can issue (e.g.180 requests per 15minutes for Twitter). As a result, it is practically impossible to crawl or download most or all data from an online social network before generating aggregate estimations.

The existing research on the aggregate estimation functions over social networks generally use random walk-based sampling on the social graph, or adaptations of it like Metropolis- Hastings. However, they are inefficient for the type of aggregate queries that we study as they only consider broad aggregates, i.e. Aggregates on the whole social network, and not constrained by keywords. Most of these techniques enable aggregate estimation by drawing a random sample of all micro blog users, and extrapolating from the sample. A critical problem of existing sampling techniques is the large number of individual-user queries (i.e., web requests) they require for retrieving each sample. Consider the simple random walk as an example where in order to reach the stationary distribution (and thereby an accurate aggregate estimation), one may have to issue a large number of queries as a "burn-in" period of the random walk.

In our case study, nevertheless, aggregate queries have keyword selection conditions that match only an extremely small fraction of these users - e.g., the number of Twitter users who have used the keyword privacy in their postings is only 0.4% of all active users. A straightforward solution would be to only consider users who satisfy the selection condition during the sampling random walk. However, we found that this leads to a social sub graph with tightly connected communities that significantly increase its convergence time.

Aggregate queries like sum, count and average are useful to analyze and draw results over the publicly available micro blog data. Random walk is an efficient approach for approximating certain aggregate queries on social networking applications. When a simple random walk is sufficiently long, the probability for each user to be sampled tends to reach a stationary distribution proportional to each user's degree (i.e., the number of users connected with the user). Thus, based on the retrieved samples and knowledge of such a stationary distribution, one can generate unbiased estimations of AVG aggregates (with or without selection conditions) over all the users in the social network. If the total number of users in the social network is available, then COUNT and SUM aggregates can be answered without bias as well.

Chapter 3

Data Access Model

The access to the data over micro blogs network is becoming increasingly difficult and the micro blog APIs officially provided doesn't support large amount of data mining due to commercial interests as well as security considerations. The data-access model abstracts the API interfaces provided by most popular micro blogs and helps in leveraging the wealth of micro blogs for analytics. The micro blogging platform offers mainly three functionalities.

1. Communicate and share brief updates in textual, image and video formats
2. Build social network
3. Search and subscribe to posts

These platforms provide mainly three types of queries that represent the functionalities listed above.

SEARCH: This type of query allows searching based on a keyword. For instance: given a keyword (or keywords) *w*, return *recent* posts that contain *w*. There are limitations imposed by most micro blog sites in terms of time period and result count which is mainly due to the user's interest in more recent data and also for commercial purpose where service providers consider selling the historic data. Most micro blogs return posts in recent weeks – e.g., the last week's posts in Twitter API. Other micro blogs restrict search to top-*k* results where *k* could be in the low thousands.

USER CONNECTIONS: This type of query allows searching for the network of a user under examination unless the user sets the privacy. For e.g.: Given a user u, the query fetches all other users connected with u. The user u can be connected to the other users either as a friend, follower, or being followed etc.

USER TIMELINE: This type of query fetches all the information published on the user’s timeline. For instance: Given a user u, the query return all posts published by u. There are exceptions where the provider publishes the most recent posts published by a user. For e.g.: Twitter publishes the most recent 3200 tweets published by a user. However, it is observed that only a very small percentage of users - 5% - have posted more that 3,200 tweets and even for these users only very old tweets are missing. Since our study is on aggregate estimations, this small number of incomplete user timelines can be considered negligible while calculating the aggregate estimations.

The interfaces described above can also be implemented using web crawler while the due to their unpredictable search and ranking criteria, they are not preferred over APIs for aggregate estimation. Twitter, posts may be missing from the web search but not from the search API results also, there is a restriction on the number of queries that can be issued in a given time frame. For e.g.: Twitter’s search API allows only 180 queries over a 15 minute window, and Reddit API allows no more than one request every two seconds.

Chapter 4

Methodology

Online social networks feature restrictive web interfaces which only allow the query of a user's local neighborhood through the interface. To enable analytics over such an online social network through its restrictive web interface, many recent efforts reuse the existing Markov Chain Monte Carlo methods such as random walks to sample the social network and support analytics based on the samples. The problem with such an approach, however, is the large amount of queries often required (i.e., a long "mixing time") for a random walk to reach a desired sampling distribution.

In this document, we describe the MICROBLOG-ANALYZER, an efficient platform to enable the accurate estimation of aggregate queries over an online microblogging service. The problem of aggregate estimations over microblogs can be addressed by issuing queries through the web search micro blog interface. Specifically, the aggregate queries of the form -

SELECT AGGR (f(u)) FROM U WHERE CONDITION

Here U is the set of all users, f(u) is a function that returns numeric measure for each user u (e.g., age or #connections), AGGR is an aggregate function such as COUNT, SUM or AVG, and CONDITION determines whether a user should be considered for the aggregate. This approach is feasible for both aggregations over the users and also the user posts. For example, the COUNT of posts containing keyword privacy can be specified as follows: CONDITION returns TRUE if a user has privacy appearing in its timeline, and FALSE otherwise; f (u) returns the number of posts containing privacy in the user's timeline; and AGGR is SUM. The CONDITION in this case can hold any phrase. Here we list two such phrases - keyword and time window. The keyword phrase is used to fetch the user whose timeline contains the pre-determined keyword e.g.: privacy in this case. The time window is used to fetch users who mentioned the keyword privacy between Jun2014 and Dec2014. Most of the research by social scientists focuses on the study that involves one or more keyword oriented queries with or without a time window. Hence in this document, we consider aggregate queries with option time window on user profile attributes like gender, age etc. to get accurate results.

User Timeline Analytics/Retrieval

Most Microblogs provide free access to their data, which offers great opportunity for researchers like social scientist. A social scientist or a non-commercial application would be interested in studying and analyzing the publicly accessible microblog conversations to determine certain factors like public response on a socio-political issue or survey on popularity of a gadget etc. These applications offer search API calls which allows retrieving posts based on query keywords but also place limitations in terms of maximum number of search results. There are also limitations on estimating aggregate functions on social network which uses random walk-based sampling since they only consider broad aggregates on the whole social network and not limited to keywords. The Microblog-Analyzer that we developed in this thesis is designed based on a central and novel idea where the user-timeline interface is leveraged to bypass the previously described limitations on the search API. We developed a novel solution exploiting the user-timeline information that is publicly available in most microblogging platforms. Theoretical analysis and extensive real-world experiments over Twitter, Google+ and Tumblr confirm the effectiveness of our proposed techniques.

In this section we explain the approach of leveraging the user timeline to answer the aggregate queries. Most of the users in a microblogging service are associated in a connected graph through social relationships revealed by the service - e.g., follower/followee in Twitter, Circles in Google+, blog followers in Tumblr, comments on same post in Reddit, etc. For the purpose of this thesis, we consider such a *social graph* to be undirected. The directed relationships such as follower/followee on Twitter can easily be converted to undirected edges by considering two users to be connected if either follows the other.

Consider a social graph where we start with a user and recursively follow edges to reach and move over the timeline of other users thus answering the aggregate based on the crawled data. This method demonstrates the While this method demonstrates the means of fetching adequate information (for aggregate estimation) through user-timeline queries, it requires a considerably high query. Also, in addition, most crawled data would not be useful for aggregate estimation - e.g., even for a broad query like the count of users who have tweeted privacy in 2014, majority of user timelines would be irrelevant because only a very small percentage (0:4% of its active users) of all Twitter users satisfies the selection condition - leading to a significant waste of resources. On the contrary, the MICROBLOG-ANALYZER samples only the *users* who satisfy the keyword predicate specified in the aggregate query, and then produce aggregate estimations according to the collected sample.

There are two design issues that are critical for enabling the sampling-based method –

(i) Sub graph generation and removal of redundant edges - An unambiguous method to sample user timelines is to carry out a random walk over a given social graph - e.g., a user recursively crawls over the timelines of other users in neighborhood in random such that the time lines of sample users can be used in aggregate estimations. However, the topology of the social graph is not favorable for sampling and requires a high query cost and contains many “redundant” edges which may “trap” a random walk inside a tightly connected component - i.e., preventing the walk from efficiently sampling all nodes in the graph. Hence this issue has to be addressed by removing the redundant edges ‘on-the-fly’ and generating a subgraph that satisfies two conditions - (i) high recall which ensures the closeness of the estimations generated from the subgraph and (ii) sample-friendliness which ensures an efficient random walk process.

(ii) Sampling design: Microblogs have been using the random walks for aggregate estimation over large graphs while there is a significant query cost associated with the SUM & COUNT queries. The knowledge of the total number of nodes in a graph is necessary for generating estimations of queries using random walks for the SUM and COUNT; in absence of which, a significantly more expensive mark-and-capture based technique needs to be used. However, in this method, $\Omega\sqrt{n}$ samples are needed to produce just one collision over an *n-node* graph an extremely high query cost even for a perfectly built sub-graph containing only users satisfying the selection condition. For example, to estimate the COUNT of all users who tweeted privacy in 2013 (about 894,000), this means at least thousands of samples must be collected, incurring a very high query cost. Hence, this issue has to be addressed by designing a sampling algorithm to efficiently traverse the graph to estimate aggregates like AVG, SUM and COUNT.

Level by Level SubGraph

A level-by-level subgraph is a subgraph that only contains edges between different levels. In practice, the random walk in this case, needs to follow a simple rule: transit from a user to its neighbor if and only if they did not first tweet privacy in the same day. In this thesis, we focus on constructing a subgraph that satisfies two major conditions.

(i) high recall which ensures the closenesss of the estimations generated from the subgraph.

(ii) sample-friendliness which ensures an efficient random walk process.

A straightforward subgraph construction that serves as a baseline for our thesis is ‘term induced’ subgraph. The term-induced subgraph differs from the original graph in terms of the number of users timelines. Unlike the original graph where all the user timelines are included, the term-induced subgraph consists of only users who satisfy the keyword selection condition of the aggregate query. Let us consider an example, the estimation of the following aggregate query over our Twitter prototype: AVG (number of followers) of users who tweeted the keyword privacy in 2014. Application of ‘term-induced’ method in this example, leads to a subgraph consisting of all users who have tweeted privacy before. This implies, during the random walk process, we always start with a user who has privacy in his/her timeline and only transit to users who satisfy the same criteria.

The principle for this approach is that the nodes in the term-induced subgraph form a superset of those covered by the aggregate and hence the subgraph has a high recall as long as it remains connected. Also, due to the reduced graph size by using keyword predicates, the sampling efficiency is likely to be improved. Although the design of the subgraph vastly reduces the subgraph size while keeping it connected, the other condition may not be satisfied where for a time-interval condition which, when excessively short, can result in a low recall.

Table 4-1 Statistics: Term Induced and Level-by-Level Sub graphs

Keyword	Recall	Avg#common neighbors	% of intra & cross-level
FiscalCliff	97%	16, 2	27%, 1%
New York	91%	49, 3	32%, 2%
Super Bowl	93%	34, 1	29%, 2%
Obamacare	96%	21, 5	22%, 1%
Tunisia	86%	11, 4	28%, 1%
Simvastatin	81%	19, 2	24%, 2%
Oprah Winfrey	91%	22, 4	29%, 3%

We conducted experiments on Twitter with keywords and hash tags for most popular ones like Fiscalcliff, New York, Superbowl to more obscure ones such as Tunisia, Simvastatin) which helped us confirm the validity for the high recall assumption. The largest connected component of the subgraph contains almost all (on average 94%) nodes in the subgraph demonstrating the high-recall of a term-induced subgraph as shown in the table above.

We start with the idea of studying a level-by-level subgraph by introducing a classification of edges in the term-induced subgraph and how each type of edges can affect the efficiency of random walks. Let us consider an elementary organization of nodes (user in this case) into multiple levels according to the time when a user first qualified for the keyword predicate (e.g.: tweeted privacy). Consider an arbitrary time interval i.e. 1 day. We partition all users in the term-induced subgraph into multiple segments according to the interval (e.g., users published privacy between 01Jan14 and 31Oct14 will be partitioned into 303 segments).

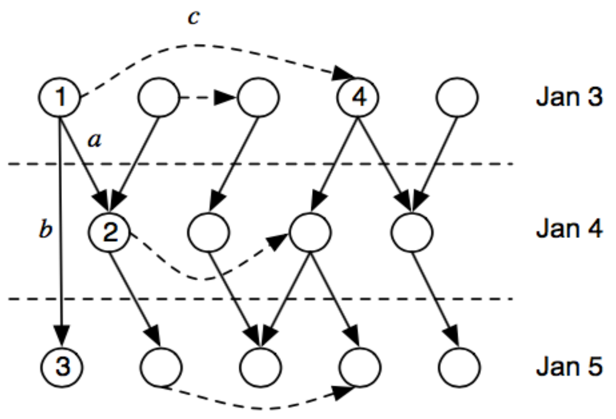

Figure 4-1 Level By Level – Term Induced Subgraph

If we draw each segment as a “virtual level” as in Figure 4.1, and place these levels from top to bottom in chronological order, then we can classify all edges in the subgraph into three categories:

(i) Adjacent-level edges connect two users in adjacent levels - e.g., Edge ‘a’ in Figure 4.1 connects User 1 who first tweeted privacy on Jan 3 and User 2 who did so on Jan 4. These types of edges are beneficial in the random walks

(ii) Cross-level Edges connects two users in unequal and non-adjacent levels - e.g., Edge ‘b’ in Figure 4.1. These types of edges increase the efficiency of the random walks but are rare in practice (<1% for privacy keyword).

(iii) Intra-level Edges connect two users in the same level - e.g., Edge ‘c’ in Figure 4.1 This type of edge decreases the efficiency of the random walks.

The different type of edges listed above have varying effects on the efficiency of sampling. The intra-level edges usually exist between users in a tightly connected component, while adjacent and cross-level edges are most often not. We can observe that, on average, one in four edges in the term-induced subgraph is an intra-level edge.

Further, the users connected by intra level edges have significantly more common neighbors. Therefore, to “burn-in” to a stationary distribution, subgraph should have cross-adjacent and/or cross-level edges and not many intra-level edges. However, significant percentage of edges in a real-world term induced graph is intra-level ones (e.g., even for a short interval of 1 hour, more than 28% of edges for keyword privacy are intra level ones). Hence, the key idea for our Level-by-level subgraph design is to remove all intra-level edges from the term-induced graph and to deal with the time interval used in defining the intra level edges.

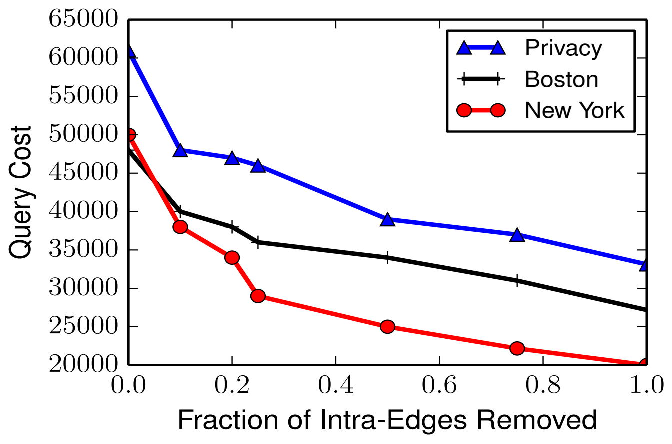

Figure 4-2 Intra Edges and Query Cost Comparison

In figure 4-2, it is evident that the removal of intra-level edges significantly Increases the graph conductance and thereby make the random walk process more efficient. Our experiments on the Twitter prototype verified this finding. The figure above shows, for various keywords, how the removal of 10% to 100% randomly chosen intra-level edges affect the query cost of simple random walks to achieve a relative simple random walks to achieve a relative error of <= 5% on estimating the average number of followers for all users who tweeted the keyword in 2014. One can observe from the figure 4.2 that as the query cost decreases dramatically when intra level edges are removed.

Level by Level Randomwalk

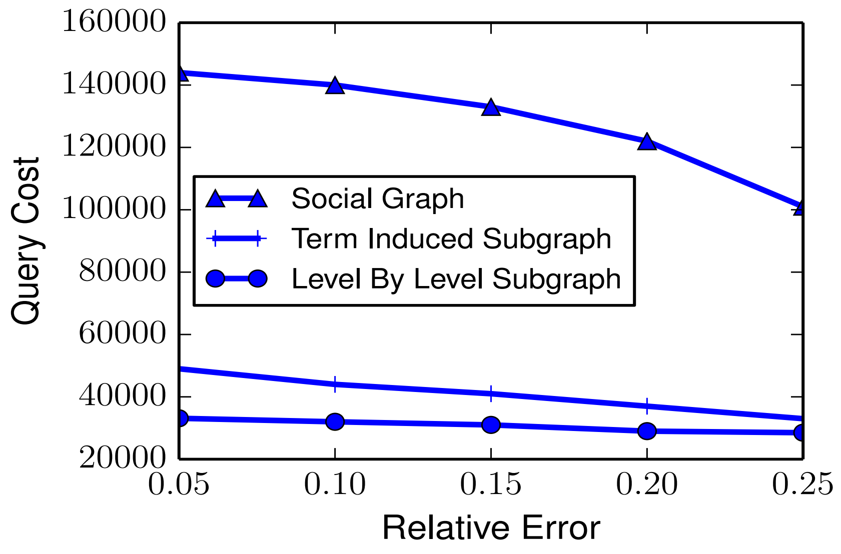

Figure 4-3 Average Users tweeted privacy

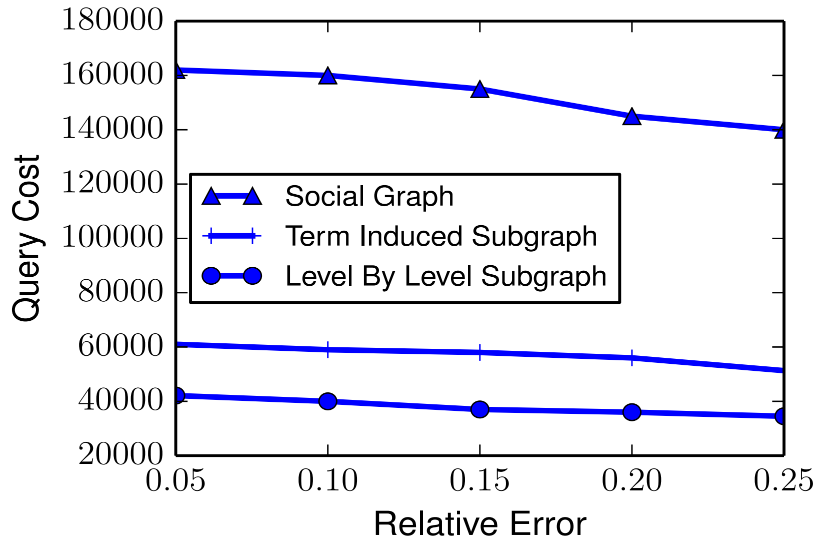

Figure 4-4 count (user) tweeted privacy

In the figure 4.3 above, we observe that though the simple random walk is performed on the level by level sub graph as in the case of social graph and term-induced subgraph, the query cost reduces considerably. The level by level sub-graph has a special property - directed acyclic graph (DAG) due to which we adopt a new algorithm MA-TARW [1] which will further reduce the query cost. While the query cost is indeed much lower than the original social graph, it is still very expensive. In the figure 4.3 above (average number of followers for users who tweeted privacy), this subgraph required close to 49,000 queries to obtain an estimate with less than 5% relative error. While this value is significantly less than the 144,000 queries required for the original graph, it is still high considering Twitter's rate limit.

The figure 4.4 above shows the term-induced subgraph performs on estimating the Average and COUNT for users who tweeted privacy. To understand why the efficiency problem remains with the term-induced subgraph, we note that even though users who tweeted privacy only represent a small percentage of all Twitter users, the number of *edges* connecting them in the term induced graph is still very large (e.g close to 1 million edges connecting approximately 142 thousand nodes for the running example).

Our topology-aware, level-by-level, random walk follows a bottom-top-bottom flow on the subgraph - i.e. a random walk instance starts from the bottom level and moves up one level at a time, by following the inverse direction of edges, until it reaches a node with no incoming edge. Then it reverses traversal direction and starts following the original edge directions to transit down, again one level at a time until it reaches a node with no outgoing edge. At each transition during the random walk, a branch is chosen uniformly at random. Note that all nodes we pass through during a random walk will be used to generate an aggregate estimation. In a level-by-level random walk, each instance passes through a set of nodes during the bottom-top and top-bottom phases. After each such instance terminates, we take the sets of nodes the instance passes through during the phases and for each of these nodes, we start a bottom-top, level-by-level random walk starting from u for recursively estimating p(u) and this process is repeated multiple times. One can execute multiple instances of the random walk and average out the results to produce more accurate estimations.

In a level-by-level random walk, the query cost required by each instance of the random walk is much smaller than that for traditional topology random walks. Specifically, our walk instance requires at most $2(h-1)$ transitions, fewer than simple and Metropolis-Hastings random walks. By leveraging knowledge of the level-by-level topology, our random walk process is capable of transiting between different "clusters" of nodes much faster than traditional topology-oblivious random walks and thus forms an advantage. Specifically, for a $2(h-1)$ step level-by-level random walk instance over the above described *h*-level graph, each of the first (or last) *h* - 1 steps is guaranteed to draw from mutually exclusive subsets of nodes. This makes the random walk process reach all nodes in the graph much faster than traditional random walks.

The level-by-level random walk process requires fewer queries than traditional random walks (simple or Metropolis Hastings). To understand the level-by-level random walk process, we begin by considering an example where a level-by-level subgraph constructed for a given keyword (e.g.: privacy) has *h* levels and only edges between nodes of adjacent levels. We develop a level-by-level random walk process by leveraging knowledge of the subgraph topology and estimate $p\ (u)$ in a level-by-level random walk, which in turn enables accurate aggregate estimations. As shown in Figure 4.1 above, the top level consists of users who mentioned the keyword earliest, while users at the bottom or few levels above are guaranteed to be returned by Twitter's search i.e., our random walk process starts from these bottom levels. Note that every edge in the graph is directed from top to bottom.

Chapter 5

Architecture

Microblog-Analyzer is a system for enabling analytics over a microblog by issuing queries through its limited access interface. The two main components of Microblog-Analyzer are

(i) Graph Builder: The generation of a conceptual graph that connects user timelines together.

(ii) Graph Walker: The design of an efficient sampling algorithm over such a graph.

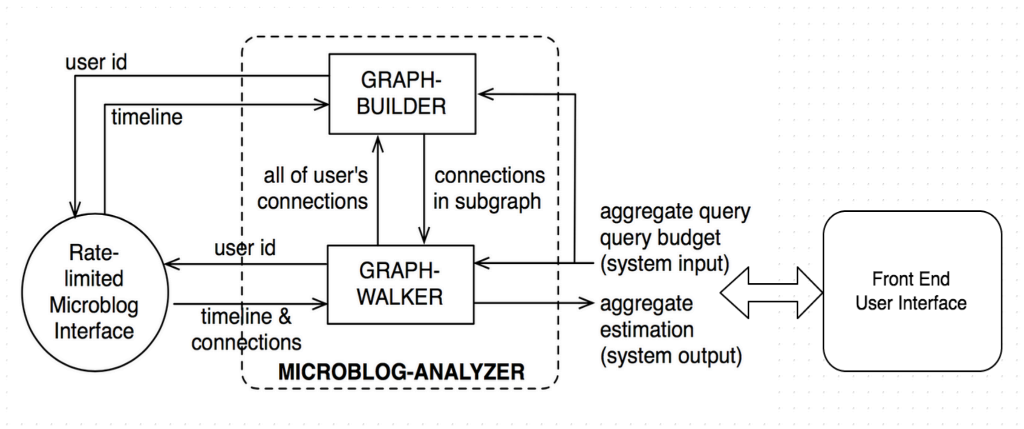

Figure 5-1 System Architecture

The architecture comprises of an aggregate query to be estimated, query budget implying the maximum number of queries and one or few ‘seed users’ who posted the blogs satisfying the selection condition of the aggregate. These are fed as an input to the system as depicted in the figure above. For a given seed user, the Microblog-Analyzer uses a Graph-Builder to determine its neighboring users. The Graph-Builder can be designed by using all the social connections of a user or can be a subset of such social connections with a well designed algorithm that considers the aggregate estimation and user timeline information. For a given set of neighbors, the MICROBLOG-ANALYZER uses a GRAPH-WALKER to determine the probability to transit-to and sample each neighbor for aggregate estimation. The GRAPH-WALKER can be designed by choosing each neighbor uniformly at random or can be a well-designed algorithm that considers certain topological characteristics of the graph produced by the GRAPH-BUILDER.

The process of building the graph connecting users and performing a random walk over such graph can be repeated multiple times until exhausting the query budget, so as to produce a more accurate aggregate estimation as the final output of MICROBLOG-ANALYZER.

Micro Blog Analyzer Component

Graph Builder

The GRAPH-BUILDER aims to construct a subgraph of a social graph with two properties:

(i) A high recall of (timelines of) users who satisfy the selection condition of the aggregate query to be estimated.

(ii) A topology that enables efficient sampling of such users.

The high recall of users can be achieved by constructing a term-induced subgraph, which includes only users who satisfy the keyword selection condition (privacy) of the aggregate query. Since the nodes in the term-induced subgraph form a superset of those covered by the aggregate, the subgraph has a high recall as long as it remains connected or has a large connected component. On the other hand, the sampling efficiency is likely to be improved because of the reduced graph size. The design of the subgraph balances between the two objectives by filtering nodes only with keyword predicates which vastly reduces the subgraph size while keeping it connected but not other conditions in the aggregate query - e.g., a time-interval condition which, when overly short, can result in a low recall. To understand why the efficiency problem remains with the term-induced subgraph, we note that even though users who tweeted privacy only represent a small percentage of all Twitter users, the number of edges connecting them in the term induced graph is still very large (e.g., close to 1 million edges connecting approximately 142 thousand nodes). With such a large and dense graph, the efficiency of sampling critically depends on whether the graph topology is carefully designed to enable efficient random walks. The design of the term-induced subgraph cannot adequately address the sampling-efficiency problem of the original social graph, mainly because of the long burn-in dictated by traversing between tightly connected communities. In the next subsection, we describe our proposed methods for constructing a "sampling-friendlier" subgraph topology - specifically, by exploiting *time dimension* of the term-induced subgraph - i.e., the time order with which users posted a specified term like privacy.

Consider a term-induced subgraph as in the figure 1 where the edges are classified into three categories Adjacent-level edges, Cross-level Edges and Intra-level Edges. In earlier sections, we found that for a "reasonable" time interval (>1 hour), (more) intra-level edges are *detrimental* to the efficiency of random walks, while (more) adjacent-level edges are *beneficial* to it. Cross-level edges, on the other hand, contribute to more efficient random walks but are relatively rare in practice (e.g., less than 1% for privacy).

The key idea for the subgraph design is to remove all intra-level edges from the term-induced graph. We refer to this subgraph as the level-by-level subgraph to properly design a level-by-level subgraph, we have to address two issues

i) The effect of intra-level edges on the efficiency of random walks.

ii) The setting of time interval, which affects the edge classification.

Effect of intra-level edges: Consider the change of graph conductance after removal of intra-level edges. The conductance (G) of a graph G measures how "well-knit" G is i.e. how fast a random walk can converge to its stationary distribution.

$$\varphi(G) = \min_{S\subseteq V} \frac{\sum_{v_i\in S, v_j\in \bar{S}} a_{ij}}{\min\,(a(S), a(\bar{S}))}$$

Where V is the set of vertices in G, S and S = V\S form a partition of V into two disjoint subsets, a_{ij}= 1 if there is an edge connecting v_i and v_j in G and 0 otherwise, and $a(S) = \sum_{v_i\in S\, v_j\in\bar{S}} \sum_{v_j\in V} a_{ij}$. In general, a simple random walk burns-in faster on graphs with higher conductance [2].

To understand the construction of GRAPH-BUILDER, consider an example of a level-by-level subgraph G. Let there be n nodes in the graph which are distributed evenly across h levels The adjacent-level edges in the graph are constructed such that each node at Level i ($i \in [1, h-1]$) is connected with d nodes chosen uniformly at random from those at Level $i+1$. The intra-level edges, on the other hand, connect each node at Level i with $d^!$ Other nodes chosen uniformly at random from Level i. While this simple model does not match real world graph topologies, it nevertheless this model gives us an indication of how intra-level edges affect conductance [1], Refer figure 4.2 for the impact of the intra level edges on query cost.

Time Interval in Level-by-Level Subgraph: To address the issue of how to properly set the time interval *T* which directly affects edge classification, consider the level-by-level subgraph explained in previous section. The setting of T (time interval) affects two parameters

- The number of levels h – the longer T is, the smaller h.
- d, The number of (randomly chosen) Level $i+1$ nodes a Level i node is connected with.

While a longer T will in general lead to more nodes on Level $i + 1$, it *might* actually reduce *d* if most followers of the Level i node already responded within the time interval corresponding to Level i. The equation below illustrates the relationship between h and d in order to maximize the conductance of the level-by-level subgraph.

$$d = \frac{(2h-1)(2h-2)}{h(2h-9)}$$

Hence, instead of setting the T to a fixed value, we should adjust it according to the propagation pattern of the query term or hashtag. Specifically, the average number of followers who "pick up" the hashtag after the current time interval should be close to its optimal value *d* as shown in above equation. For example, if the average degree is around $d = 14$, then there should be around $h \approx 5$ levels in the lattice structure. By combining the level-by-level subgraph with the simple random walk, we follow an Algorithm MA-SRW[1], which enables aggregate estimation.

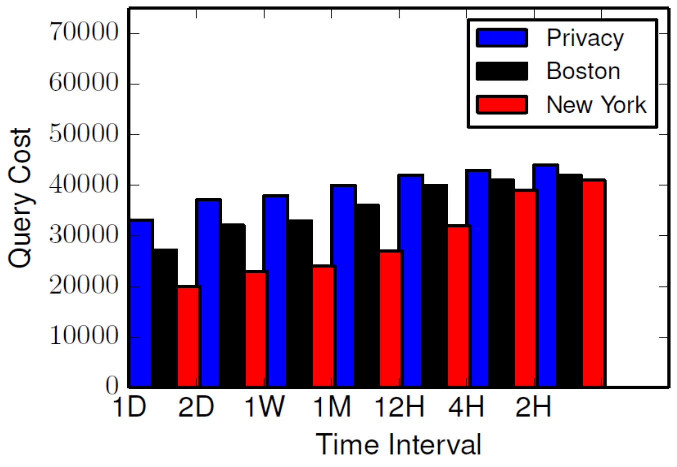

Figure 5-2 Impact of T on query cost (H=hours, D=days, W=weeks, M=months)

Consider the figure above where we identified a set of diverse time intervals varying from 1-hour to 1-month. For each time interval, we estimated its efficacy in sampling as against the theoretical value of the conductance. In other words, we ordered the time intervals in the each of these time intervals and compared the query cost to achieve a relative error of less than 5%. The figure above shows the results for three keywords. The orders based on theoretical conductance and experimental performances are consistent.

Graph Walker

GRAPH-WALKER determines the probability for MICROBLOG-ANALYZER to transit to and sample each neighbor of a user for aggregate estimation. The design ranges from simply choosing each neighbor uniformly at random (i.e., simple random walk) to a carefully designed algorithm that takes into account certain topological properties of the graph produced by GRAPH-BUILDER.

The existing random walk techniques have two main problems: (1) although they produce asymptotically unbiased samples after a burn-in period, the number of transitions required for the burn-in is usually high; and (2) while they can be combined with mark-and-recapture to estimate SUM and COUNT queries based on the samples, the query cost often rises to a prohibitively high level for practical purposes. The reason underlying these problems is the inability of traditional random walk techniques to estimate the probability for a node u to be chosen as a sample. Note that while simple random walk is known to have a stationary distribution that assigns probability proportional to a node's degree $d(u)$, it is still impossible to compute the exact probability for a node to be accessed (i.e., ${d(u)}/{2|E|}$ where E is the set of all edges) unless one knows the total number of edges in the graph.

To know the exact probability for a node to be accessed by Metropolis-Hastings random walk (i.e., 1=$|V|$ where V is the set of all vertices), one has to know the total number of nodes in the graph. Clearly, neither piece of knowledge is available *a priori* in our case - and estimating them (e.g., by using mark-and-recapture) requires a very high query cost. With the knowledge of $p\ (u)$, the probability for a node to be taken as a sample, one can simply apply the Hansen-Hurwitz estimator to generate an *unbiased* estimation for any SUM or COUNT query defined as *f*(*u*)=*p*(*u*), where *f*(*u*) is the result of applying the SUM or COUNT query over *u* itself. This avoids the usage of mark-and-recapture and, as a result, significantly reduces the query cost required for answering SUM and COUNT queries.

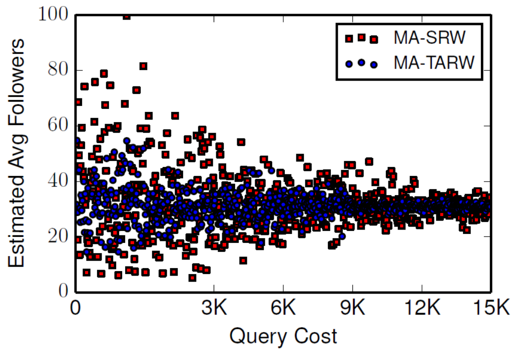

Figure 5-3 Twitter: Estimated Avg(followers)

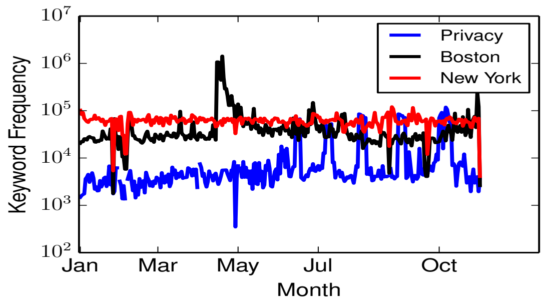

Figure 5-4 Frequency of keyword

The figure above shows that MA-TARW significantly outperforms MA-SRW. MA-TARW can be used to estimate the average number of followers of all users who tweeted privacy. MA-TARW converges to the true estimate and has a lower variance in its estimate within few thousand queries. We then perform a COUNT estimate of all users who tweeted privacy. The figure also shows the frequency of three keywords used in the evaluation over time - privacy (a relatively low frequency term with occasional spikes), New York (a perpetually popular and high frequency keyword) and Boston (keyword that has medium frequency but had a singular spike on Apr 15, 2014 when the Marathon Bombing occurred).

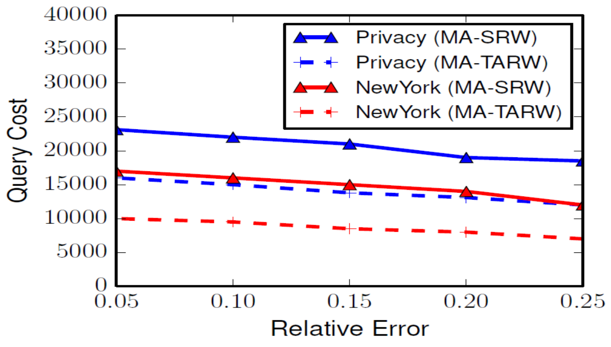

Figure 5-5 Twitter: Avg(Display Name)

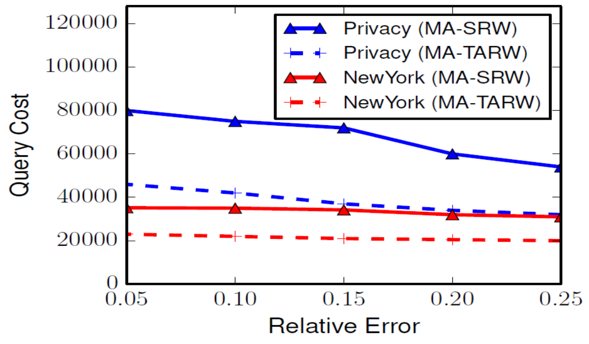

Figure 5-6 Google+: Avg(Display Name)

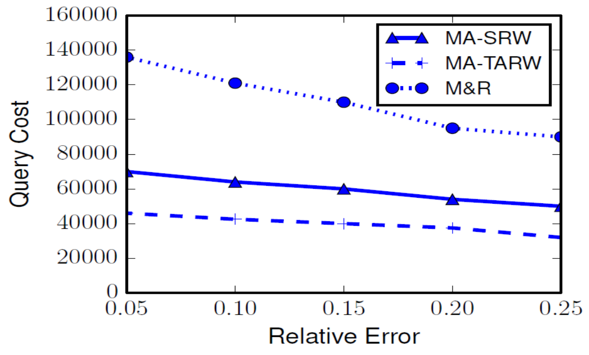

Figure 5-7 Google+:Count (male user tweets)

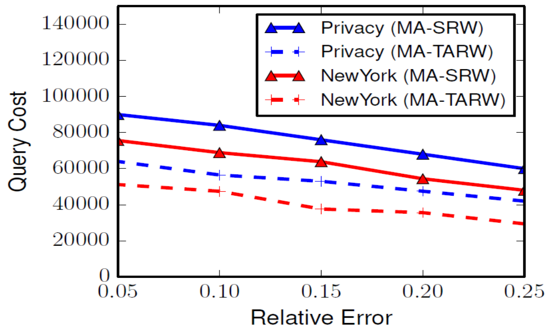

Figure 5-8 Tumblr: Avg(Likes)

Consider an aggregate query to estimate the average display name length of Twitter users who tweeted privacy. In contrast to AVG (#followers) shown above, this requires substantially smaller number of queries as this measure has a lower variability than that of number of followers. Figure – 5.5 that MA-TARW seems to leverage this aspect by essentially “skipping” such edges (which would have often been intra-level edges) Next we evaluate our algorithms on Google+. Figures -5.6 and 5.7 show the performance of estimating the average display name length and count of male users (gender is generally missing from Twitter profiles, and hence we did not use it as a condition above) who posted privacy during the time period. We notice that MA-TARW outperforms the competing algorithms. It must be noted that the absolute query cost is much higher than in Twitter. This is to a large extent due to the fact that APIs of Google+

(such as Activity search) returns at most 20 results per invocation compared to 200 in Twitter’s timeline API. Finally, we evaluate our algorithms on Tumblr. Here, we evaluated the average number of likes obtained by posts with *textual* content containing the keyword privacy. Figure 5.8 show that MA-TARW has the best performance.

Chapter 6

User Interface Description

The user interface of the system is implemented with HTML5, Java script, D3.js .The system logic and algorithm is implement using python because there are lot of extensive libraries available like tweepy, networkx and matplotlib which eases coding and provide standardized solutions that enhances the portability of the system. The Microblog platforms in this case twitter is queried by feeding the keyword/user ID as input to the twitter API. The Aggregate input signifies if a SUM/AVG/COUNT is to be derived and Attribute input accepts keywords like #icebucketchallenge etc. Clicking on Estimate submit option, directs to the Microblog and Input data selection page.

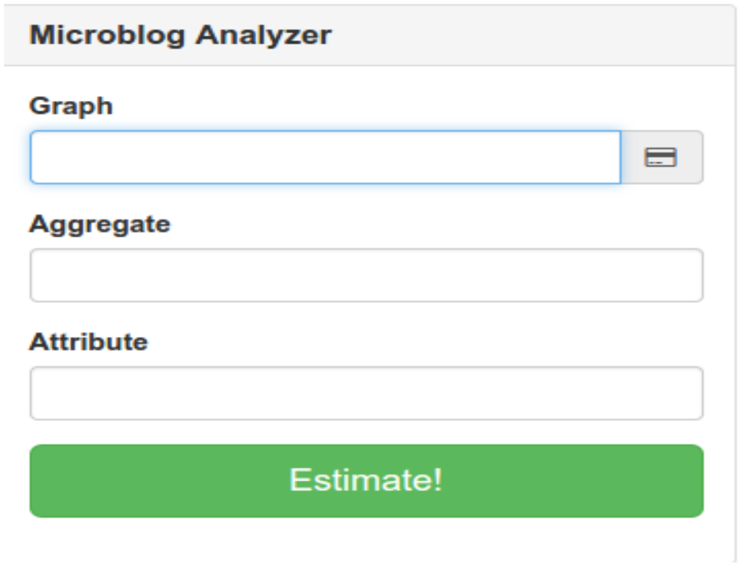

Figure 6-1 Query Aggregate and Keyword Selection

As depicted in the image below, the Microblog and data selection page lists different options to run the algorithm, Microblogs like Twitter, Google+ and Yelp etc. Theoretical Graphs like Barbell graph, network graph etc. and load file to provide the input file which contain nodes and edge list.

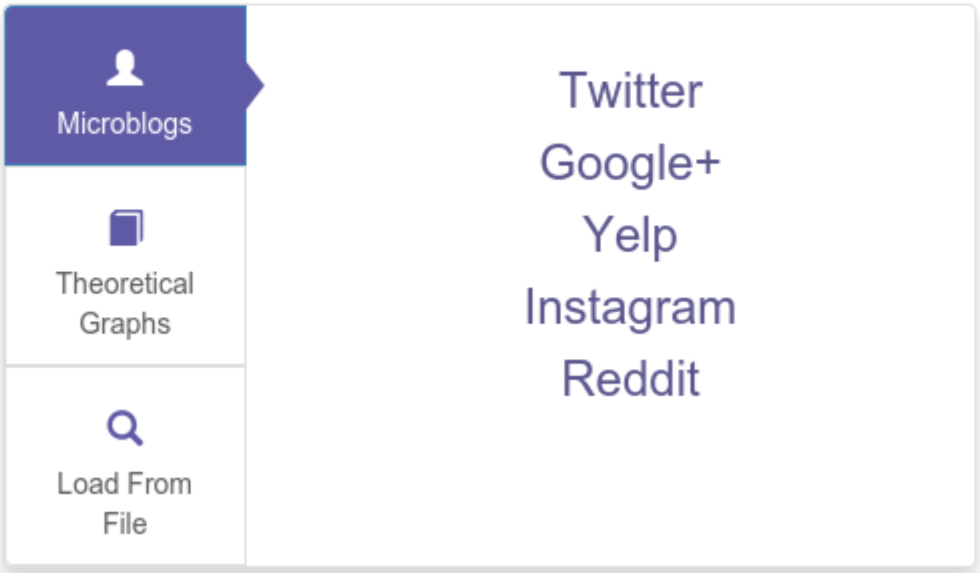

Figure 6-2 Microblog and Input data selection

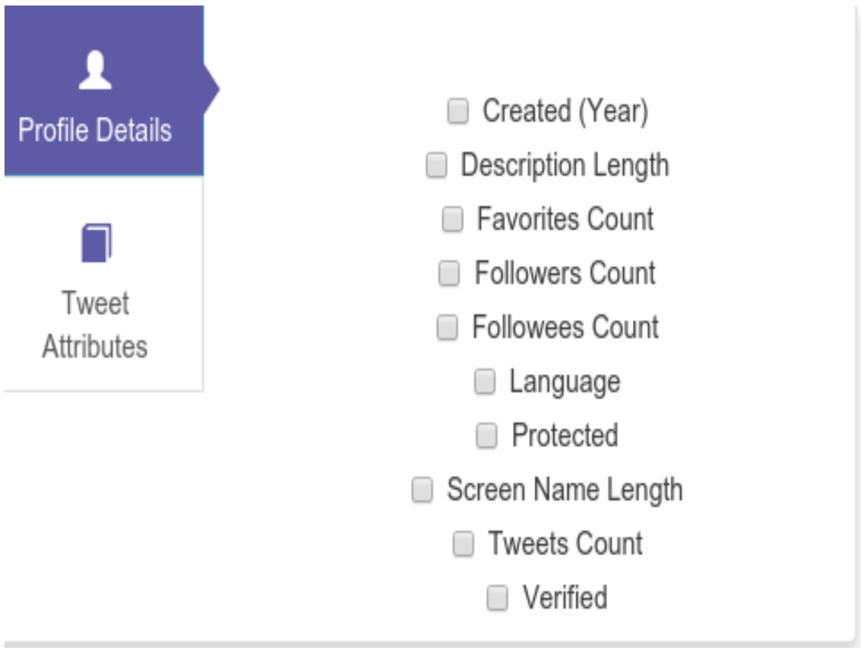

Figure 6-3 Profile Attributes Selections

The profile attribute selection page Figure 6.3 displays multiple options, on each of which the aggregate queries are run. For instance, AVG of followers, COUNT of followee etc. When the user details are fetched from the twitter api, only these attributes are maintained. Also in Figure 6.4 we will be able to select the tweet attributes to fetch and store from the API's resulting JSON.

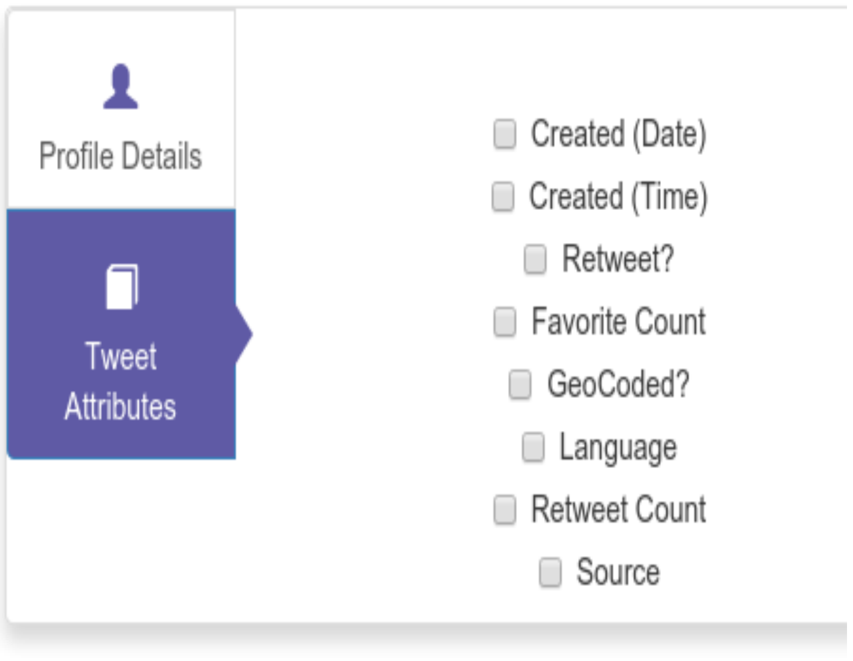

Figure 6-4 Tweet Attributes Selection

This is achieved using tweepy, which is a twitter library for python and has a stack of available options. The API request to twitter timeline returns all the information of user u and also information of the follower of u as JSON object. The *json* format is designed to capture all the events happening during the random walk. The UI loads these *json* files and populates the nodes, edges of the graph. The JSON object is iterated to fetch the details like tweets, followers, followee, location, time etc. The experiments are carried out on both theoretical graphs and real world social networks such as yelp users, Google plus and twitter users.

The graph is constructed using a python library called networkx. This library has wide collection of functions and their implementation. Further, the random walk is being performed using the networkx library function to travel from one level to another level by in bottom-top-bottom direction. During this walk we will also calculate the probability of each node using its in-degree and out-degree (connected neighbors). Hence the probability estimation, nodes, node degree, will be create as a JSON object and written to a file, which can be fed to the UI to display.

The results show that MA-TARW outperforms MASRW. The query cost of random walks is displayed as shown in Figure below. The UI here shows only the ID of a particular node, which is an integer. For a small graph, the result can be observed clearly and as the size of the graph grows, the UI shows how the random walk performs over different set of nodes.

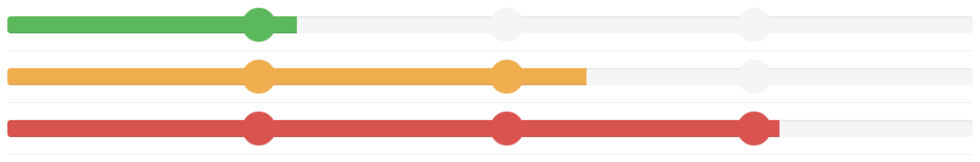

Figure 6-5 Query Cost

Chapter 7

Conclusion

In this thesis, we built a MICROBLOG-ANALYZER system to enable aggregate estimation over Microblog platform by dynamically constrcting a level-by-level sub-graph of the microblog and performing sampling by a novel topology aware random walk. Firstly we described the architecture and methodology of the system and its components along with the terminologies like user-time line information, level-by-level subgraph, topology aware random walk, GRAPH-BUILDER and GRAPH-WALKER that together constitutes the MICROBLOG-ANALYZER.

We provided theoretical analysis and extensive experimental studies over real-world social networks to illustrate the limitations of the existing design and advantages of our novel idea where the user-timeline interface is leveraged to bypass these limitations on the search API. Our methods demonstrated efficiency in the number of API calls made to the micro blogging service provider, which should be as few as possible in generating the approximate aggregate.

References

1. Saravanan Thirumuruganathan, Nan Zhang, Vgelis Hristidis, Gautam Das "Aggregate Estimation Over a Microblog Platform," SIGMOD, 2014.
2. L. Lovasz and R. Kannan. Faster mixing via average conductance. In STOC, pages 282–287, 1999.
3. http://mkurant.com/publications/

Biographical Information

Satishkumar Masilamani received his bachelor of Engineering in Computer Science in 2007 from Sri Venkateshwara College of Engineering, Bangalore under visvesvaraya technological university, Belgaum, India. He has worked for 2 years in Mphasis and 3 years in Oracle SSI. He started his Master in Computer Science at the University of Texas at Arlington in Fall 2013 and joined the Database Exploration Lab in Summer 2014. He also worked as Web Developer Graduate Research Assistant in Center for online Development at University of Texas Arlington from May 2014. His areas of interest are Database, Data Mining, Data Analytics and Visualization.