2013-03-12

A Study of Volunteered Geographic Information and Social Media

Li, Ren-Yu

UNIVERSITY OF CALGARY

A Study of Volunteered Geographic Information and Social Media

by

Ren-Yu Li

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF GEOMATICS

CALGARY, ALBERTA

March, 2013

Acknowledgement

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Steve Liang for his continuous support of my MSc study. He provided me a great environment for research and offered me opportunities to work with other organizations. In addition, his guidance helped me in all the time of research and writing of this thesis. He is one of the best supervisors that I have in my life.

Besides my supervisor, I would like to thank my thesis committee members: Prof. Xin Wang, Prof. Danielle Marceau, and Prof. Simon Park for their insightful comments and questions. Their comments help to improve the quality of my thesis. I also appreciate their time reading my thesis.

I also would like to thank my PDFs Dr. Dong-Woo Lee and Dr. Mohamed Bakillah for their immense knowledge and assistants during my Master study. I really enjoyed working with them. I especially thank David Chang, the research assistant of my research group for his patience and wisdom. David is an experienced programmer and a project manager, and he is the person who taught me to work professionally.

I thank my fellows Shawn Chen, Chih-Yuan Huang, Ben Knoechel, and Mohammad Ali Jazayeri, and Han-Fang Tsai. Each of them has their strength, and I have learned a lot from them. They also left me pleasant memories.

Finally, I would like to thank my family who always love me and give me supports; my friends who spent their time having fun with me, encouraged me when I was upset, and attended my rehearsals before my defense. Without them, I would not have a great life in Canada.

I thank God! God is great. I met so many great people in Canada! Thank YOU.

Abstract

The Web 2.0 has changed the way that people communicate with each other. Web 2.0 platforms allow the creation and the exchange of user-generated contents (UGCs) on the Internet. The UGCs contain explicit and implicit geographic information. The quantity of geographic information continuously increases as number of Internet users expands. Since Volunteered Geographic Information (VGI) and social media have been considered as crowdsourcing data resources to harvest geographic information from the web, this thesis investigates two types of crowdsourcing data: VGI and social media data in three topics (i.e., researches).

The first research addresses the bottlenecks of current traffic data collections and proposes a solution that utilizes VGI to solve the problems. To collect and use VGI to solve traffic problems, we have reviewed the existing transportation-related mobile applications and designed and developed a complete front-end and back-end GIS for transportation. This research may be beneficial for the design of modern transportation system.

The second research aims to explore and extract local information from social media. To achieve the goal, this research proposes an algorithm to discover transient local communities (TLCs) in time evolving social media. The proposed model integrates a damping function to filter irrelevant connections over time, a graph-clustering algorithm to identify communities, and a geo-location proximity algorithm to gather geographically close users. The algorithm has been evaluated by using a Twitter dataset and its performance has been examined in terms of the ability to extract local information.

The third research aims to discover communities of interest (CoIs) in local social media. This research may benefit the local information broadcasting and local marketing. In order to discover CoIs from the noisy social media data collected, this research employs natural language processing to clean the short messages (e.g., tweets) and proposes different ways to build sociograms. This research also proposes a graph clustering algorithm that enhances

the fast-greedy optimization of modularity (FGM) with text similarity measures to eliminate the noises created by meaningless connections. The algorithm has been evaluated by using a Twitter dataset, and its performance has been examined in terms of the numbers of CoIs discovered under different conditions.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Algorithms

List of Symbols

Symbol	Definition
AR	Augmented Reality
CTC	Communication-based Transient Community
FGM	Fast-greedy Optimization of Modularity
GIS	Geographic Information System
GLC	Geospatial Locality Clustering
GPS	Global Positioning System
GUI	Graphical User Interface
ITS	Intelligent Transportation Systems
LBS	Location-based Service
LC	Locality Clustering
LTC	Location-based Transient Community
LSN	Local Social Network
MCL	Min-cut Clustering
NLP	Natural Language Processing
OSN	Online Social Network
OSNS	Online Social Network Service
SNA	Social Network Analysis
ToI	Topic of Interest
TLC	Transient Local Community
TTL	Time To Live
UGC	User Generated Content
VGI	Volunteered Geographic Information
VSM	Vector Space Model

Chapter 1

Background and Motivations

1.1 Web 2.0

Web 2.0 has changed the way that people communicate and share information with each other. Information flows on the Internet, and everyone can share pieces of information related to specific locations on the earth. By putting all the pieces shared by Internet users together, one can better understand the dynamics of environments in the past and present.

Web 2.0 was coined by Tim O'Reilly in 1999 to describe a second generation of the *World Wide Web* (WWW) that aims to enable people to collaborate and share information online. Tim O'Reilly outlines seven themes for Web 2.0, and they are: 1) web as platform, 2) harnessing collective intelligence, 3) data is the next Intel inside, 4) end of the software release cycle, 5) lightweight programming models, 6) software above the level of a single device, and 7) rich user experiences [85].

Web 2.0 focuses on data and contents, particularly the ability for people to create and interact with rich contents rather than just to be consumers. Web 2.0 believes that the harnessing of the "collective intelligence"[1] provides "wisdom of the crowds"[2].

In addition, Web 2.0 envisions the Internet from individuals. Everyone can create and share applications on the Internet. Each application is no longer a single software on a PC that can only be manipulated by one user; rather, the application is rapidly accessible to the public through the Internet using lightweight and standard models and web programming

[1]Collective intelligence is a shared or group intelligence that emerges from the collaboration and competition of many individuals

[2]Crowd presents a group of people

languages (e.g., RESTful[3] and XML[4]). As a result, there has been an explosion of public *Application Programming Interfaces* (APIs) allowing developers and users to call functions and data from multiple diverse resources.

Since data is important to Web 2.0, Web 2.0 systems thrive on network effects: databases get richer as more people interact with them, applications become smarter as more people use them, and marketing strategies are more effective when they are built around real-time online information. Consequently, such systems encourage participations in data sharing. Examples of Web 2.0 systems include web communities[5], hosted services, *Online Social Network Services* (OSNSs), forums, blogs, and mashups[6].

In the context of geographic information, the growing popularity of Web 2.0 systems have created huge set of publicly available data, which contains plenty of explicit geographic information, e.g., *Volunteered Geographic Information* (VGI), and implicit geographic information, e.g., social media data. The data quantity continuously increases as number of Internet users expands. For example, the use of positioning and wireless telecommunication technologies has increased the number of geo-tagged[7] information on the Internet. More VGI is available.

In addition, Stefanidis *et al.* argued that the information disseminated through social media conveys ambient geographic information for that the feeds often contain geographic footprints (e.g., the place where the tweets originate, and the geographic entities). The immense amount of data generated from social media provides opportunities to understand and extract useful geographic information.

[3]Representational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web

[4]Extensible Markup Language (XML) is a markup language created to structure, store, and transport data by defining a set of rules for encoding documents in a format that is both human-readable and machine-readable

[5]A web community is a website that organize and share particular types of content

[6]A mashup is an web application that combines and presents data from two or more sources in order to create new functionality

[7]A geo-tag is a standardized code that can be inserted into information in order to note its appropriate geographic location

Both VGI and social media have been considered as crowdsourcing data resources to harvest geographic information from the web, and they have attracted a lot of researches recently [94, 95]. This thesis focuses on studying the geographic aspects of both VGI and social media. The following sections provide a brief overview of this thesis's data resources.

1.2 Crowdsourcing Data

1.2.1 Volunteered Geographic Information

Volunteered Geographic Information, coined by the geographer Michael F. Goodchild [50], is a geo-referenced crowdsourcing geographic information (GI). The first VGI was created in 1507 when Martin Waldseemüller drew an outline of a new continent and labeled the continent America (i.e., geo-tag). Hundred years after, VGI has been collected through the Internet. For example, OpenStreetMap [8] allows the general public to create a patchwork of the world map; Google Earth [5] encourages volunteers to develop applications using their own data; and Wikipedia [14] adapts knowledge shared by the general public.

Although VGI is occasional geographic information shared by Internet users and may not be accurate, its high spatiotemporal resolution provides accessible references that answer the needs of the specific locality (e.g., real-time emergency managements for certain areas).

1.2.2 Social Media Data - Online Social Networking

Social media is a group of Internet-based applications that build on the concept of Web 2.0 to allow the creations and the exchanges of user-generated contents (UGCs). With the prosperity of online social networking sites (i.e., OSNSs), social media have become one of the most powerful sources for news updates.

There have been researches extracting geographic information from social media. For example, Cheng *et al.* utilized geo-referenced Twitter information to geo-locate messages based on UGCs [107]. Cano *et al.* leveraged temporal information in social media to identify

relationships between geo-locations and the social constructs associated with places [29].

In addition, social media contain a lot of geographic information that enables us to detect natural events (e.g., typhoon, earthquake, and flu) at near real-time [3, 97], identify emerging leaders of community, and identify optimal communication methods to broadcast information to communities [17, 52].

1.2.3 Surge in the Use of Crowdsourcing Data

There are three factors that have come together to create the surge in the use of crowdsourcing data.

1) Growing interest in online social networking and availability of faster Internet. The Smartphone adoption rate has reached over 27 percent around the world [11]. People with Smartphones spend twice as much time on social media platforms such as Facebook, Flickr [2], and Twitter, and Smartphone users share events and interests through the Internet [25].

2) Online mapping tools such as Google Maps are plug-ins. With the easy-accessed mapping APIs, more VGI is available. In Web 2.0, more techniques and APIs are available to the public, a user now can use a mapping API like Google Maps, talk about GPX, and geo-tag photos [99].

3) Cheap mobile digital cameras and GPS receivers. Smartphones' cameras can capture real-time events, and the photographs can be automatically geo-tagged with coordinates (i.e., Smartphones' GPS data). For example, Ushahidi [12] is a crowd-based application that considers photographs as one type of VGI resources.

Collectively, crowdsourcing data shows the dynamics of our earth surface, and the data quantity continuously increases. Crowdsourcing data have become promising data resources to understand our environments.

Although, crowdsourcing data can be useful in a range of areas: planning, disaster management, and environmental monitoring [47], due to the scope of this MSc thesis, this thesis focuses on the following research areas. The first part of the work started in 2009, and at the

time this research created one of the first VGI systems for transportation applications. At the same time, several researches also started exploring the use of VGI in transportation [93]. Comparing to those researches, this research is different in that it provides a client-server architecture for such VGI system for transportation applications. The second part of the thesis focuses on the discovery of communities in social media data.

The remainder of this chapter introduces the state-of-the-art in the use of crowdsourcing data (i.e., the use of VGI in transportation and the community discovery in social media data), with an emphasis on the challenges that are currently faced. We conclude this chapter with an outline of the main research objectives of this thesis.

1.3 The Use of VGI for Transportation

The rapid increase in road traffic appears to be one of the major problems facing urban and suburban areas in recent years. Traffic congestion in the United States are costing Americans 68 billion each year in wasted time and fuel [10]. Besides, bad road conditions not only cause road accidents, but also impact the emergency services responding to collisions or calls for service. Overall, a more efficient transportation system can also translate into reducing total resource use and CO2 and other pollution emissions.

Monitoring real-time traffic may help control traffic flows. The most traditional way to collect traffic is to deploy fixed sensors such as road-side inductive loop detectors [20]. However, these sensors are expensive to deploy and maintain, and they are not scalable. As a result, the traffic data collected usually cover main roads only, and this prohibits the government from acquiring a good understanding over the traffic dynamics.

To tackle the problems mentioned above, municipalities and researchers started to design and develop *Intelligent Transportation Systems* (ITSs) that utilize synergistic technologies and systems engineering concepts to improve transportation systems, so that the ITSs can save lives, save time, and save money. For example the utilization of *Mobile Wireless Sensor*

Networks (WSNs) for environmental monitoring, hazard detection, and decision making [32]. We have noticed that the conventional approaches for traffic data collection have limitations. In order to obtain higher spatiotemporal resolution traffic information, we may utilize the Smartphones technology.

Smartphones have GPS receivers, compasses, gyroscopes, accelerometers, and cameras. One can detect Smartphones' orientations, geo-locations, and speeds by using these sensors. Smartphones can also connect to other devices through the Internet and the Bluetooth. Now, over twenty-seven percent mobile phone users are using Smartphones, and Smartphones have been regarded as personal sensing devices that collect volunteered sensor data, e.g., geo-locations, speed values, and accelerations.

People with Smartphones check-in locations and share events and interests through the Internet [25]. The high penetration rate and the sensing capability of Smartphones have moved Smartphones from pure communication tools to networked mobile sensing devices. The increasing popularity of Smartphones suggests that collecting data from Smartphones has been growing several times faster than from the fixed sensors [59]. Using Smartphones as traffic data probes could be one of the most promising methods for traffic data collection.

1.3.1 Existing Crowd-based Systems for Transportation and Limitations

There are two ways to collect volunteered traffic information: opportunistic sensing and participatory sensing [68]. Opportunistic sensing tasks sensors equipped on mobile nodes (e.g., car or Smartphones) to collect data around their vicinities. Participatory sensing is a concept of obtaining information from groups of people to form a body of knowledge.

Existing mobile applications for transportation such as Google Map Mobile [6], IntelliONE.com, Microsoft Research Nericell [80], and MIT CarTel [57] only adopt opportunistic sensing as their sensing approaches. No human interpretation is involved in their data collection. For example, CarTel uses probe vehicles with GPS receivers to monitor the vehicles' movements. NeriCell focuses on monitoring road and traffic conditions using Smartphones.

Different from those opportunistic sensing system, Waze [13] collects both sensor readings (i.e., opportunistic sensing data) and semantic traffic information (i.e., participatory sensing data) from Smartphone users. For example, Waze supports a "pave" function to automatically record traces and a navigation function to guide a mobile unit through a route to a destination. Waze also provides a user-friendly GUI that allows users to contribute and visualize traffic information. In addition, Waze integrates Foursquare to enable check-in. However, Waze has the following limitations.

First, Waze only targets online users, which means that Waze collects GPS positioning locations only. However, not every Smartphone has data connection (e.g., 3G); most of the time, Smartphones collect wireless positioning locations. In addition, a Smartphone can collect GPS positioning locations without accessing the Internet. Even though the offline GPS positioning locations are not published at near real time, and the wireless positioning locations are not always accurate, these locations and the events at these locations can still provide valuable information. For example, municipalities not only want to monitor dynamic traffic flows, but are also interested in understanding the triggers of traffic congestions. Examples of these triggers could be indoor activities, such as sales for big brand of luxury products, or big concerts, etc. In addition, even though the wireless positioning locations may not be always accurate, the locations can provide accessible references to users when GPS signals are not available. When excluding these locations and events, valuable information may be missed.

Second, a Smartphone has several sensors, and each sensor can potentially help understand traffic dynamics and enable special features. Making the most of Smartphone sensors can create special user experiences and maximize the information that we can retrieve from Smartphones. For example, cameras can record real time traffic events, and accelerometers can help identify potholes and differentiate different road conditions. In addition, by using a compass, gyroscopes, a GPS receiver, and a camera, one can implement Augmented

Reality (AR)[8] for navigation. However, Waze misses some features that can be useful for transportation applications.

Finally, it is important to understand users' preferences and behaviors. Users' check-ins and paths are very valuable information for municipalities. By knowing users' paths and their favorite check-in locations, one can have more insight into users' behaviors (e.g., preferences and driving behaviors) and the traffic dynamics (e.g., average speeds of road). However, Waze provides the check-in function to record GPS positioning locations and the pave function to record new roads, but the data collected are for business only and are not available for research purposes.

The above-mentioned limitations have hindered the integration of Waze with existing ITSs. In addition, the systems mentioned are mobile applications. There are limited researches about the server architecture of a *Geographic Information System*[9] (GIS) (i.e., VGI system) for transportation. This thesis proposes an example design of the essential components for an adequate GIS for transportation (both mobile and server application).

1.3.2 Challenges for Creating Crowd-based System for Transportation

Since data collected by Smartphones are mostly geo-tagged information, in order to gather the data and make efficient use of them, it is necessary to create a GIS designed for transportation applications.

A GIS for transportation consists of a mobile application and a server. The mobile application allows users to frequently contribute and visualize spatiotemporal traffic information through different ways using Smartphones, while the server is able to handle frequent requests from users.

Because the GIS allows users to share information through different ways at anywhere

[8] Augmented Reality is a type of virtual reality that duplicates the world's environment in a computer: objects are overlaid on a camera screen, and the information displayed is updated while users move

[9] *A Geographic Information System is a system designed to store, manipulate, analyze, and visualize all kinds of geographically referenced data*

anytime, the information captured is different from that obtained by using the traditional ITS fixed sensors. For example, the traffic information may contain users' geo-locations, photographs that capture real-time events, and other information that cannot be sensed through conventional sensing approaches (e.g., contextual information, such as human feelings, emotions, etc.). The information can be valuable to traffic management for that the fixed sensors can only provide numerical data.

In addition, users can voluntarily share geographic information (i.e., VGI) to the GIS at anywhere anytime; the collected information can be very up-to-date and may have different spatiotemporal scale and density. The information can supplement existing ITSs [87, 89].

Moreover, the GIS is able to not only collect high-resolution traffic information, but also broadcast traffic information more efficient. Historically, one can turn on the radio and wait for the traffic report every fifteen minutes, but this is not efficient. The GIS can support both passive and active way for users to perceive traffic information. It can broadcast real-time events to users to prevent them from entering traffic. It can also allow users to visualize their current traffic conditions on maps. However, creating a GIS for transportation is not easy. The sub-sections list the challenges for designing a complete front-end and back-end GIS including the design of a mobile application and a server application.

1.3.2.1 The Design of Mobile Application

A typically GIS for transportation has a mobile application to collect VGI, and the mobile application should accomplish three tasks: 1) to motivate users to share traffic data, 2) to collect useful traffic information, and 3) to help users make decisions faster.

First, to motivate users to contribute traffic data, the *Graphical User Interface* (GUI) of a location-based mobile application must be user friendly and as easy and intuitive to use as possible. For example, buttons should not be small and interactions should be straightforward. Alongside an attractive GUI, the use of game mechanics (e.g., gamification) can also provide incentives to users to participate in data sharing. For example, Foursquare presents

its front-end as a game and uses a reward system to motivate users to accomplish things.

Second, in order to collect useful traffic information, functions supported by the mobile application needs be able to collect traffic information that reflects road networks and traffic dynamics (e.g., events and traffic flows).

Finally, the mobile application should potentially help reduce time spent in traffic and accidents. For example, users should be able to visualize near-by traffic conditions (e.g., events), so that they can make informed decisions. The challenges mentioned must be carefully handled in order to design a functional GIS for transportation.

1.3.2.2 The Design of Server Architecture

To turn mobile devices into sensor probes and collect high spatiotemporal resolution traffic information (i.e., VGI), a user should be able to share and visualize the information shared by other users, as well as, check-in locations at anywhere anytime. Consequently, requests to server can be frequent (e.g., user's trace). A GIS for transportation also needs to have a server that is capable of handling the frequent requests and responses efficiently.

Since VGI shared by users is geo-tagged, when storing the VGI to database, it is always preferable to preserve the VGI's location metadata, so that one can benefit from its spatial relationship. Naïve design of server architecture is to have powerful servers and large memory spaces to handle frequent requests and responses and store shared contents. Conventional data storage could be spatial relational databases, such as Oracle and PostGIS. However, powerful servers are very expensive. In addition, it is not economical for individuals and research organizations to use expensive servers and large memory spaces. Moreover, the relational databases are not suitable for transportation applications.

Relational databases mix stale and fresh tuples in the same database. However, for transportation applications, user-shared VGI becomes stale at some point. For example, a user who queries near-by traffic condition cares about the current traffic dynamics, not about yesterday's traffic conditions. The stale tuples degrade the query performances when

the data sizes of the databases increase.

Moreover, the queries of spatial relational database involve a lot of relational operations, such as *spatial join* and *contains*. These queries can become expensive operations when the data size grows. The relational architectures may degrade the scalability[10] and the query performances of databases when the number of users increases [70].

Designing a data service that preserves the spatial relationship between VGI and keeps up with fresh information enables better index for raw data and ensures better query performances. Designing customized server architectures can be an interesting research challenge.

In summary, the first major contribution of the thesis is the design of the TrafficPulse, a complete front-end and back-end GIS that harness the ubiquitous smartphones for transportation applications.

1.4 The Discovery of Local Communities in Social Media

After studying the technical architecture of a GIS, this thesis continues to study research questions related to social media. This section starts with the reasons for investigating OSN data.

The Internet has revolutionized the way that people interact and communicate with others. OSNSs (one type of social media) such as Twitter and Facebook allow millions of users to generate and update *Topic of Interest* (ToIs), as well as, share information through messages. A large amount of information is collected by these services every day. Since such OSN information reflects current events and users' current interests, the shared contents are highly location and time specific. As a result, mining the *Local Social Networks* (LSNs) and discovering local communities have the potential to help us update geographic information and route geographic information to right audiences efficiently.

Several researches have investigated data mining for OSN data. Some studies explored

[10]Scalability measures a system's ability to handle growing amounts of work in an efficient manner or its ability to readily enlarge/expand in response to that increased demand

the importance of geo-locations in communication networks [16,22,73]. For example, Blondel *et al.* analyzed the mobile phone network in Brussels and observed that there were spatial borders (e.g., municipality borders) and linguistic borders for communication-based communities [22]. This observed phenomenon suggests the strong attachment between frequency of communication and geo-location in OSN data.

However, most studies for mining OSN data focus on global scale data analyses, and the number of studies related to LSNs is limited. As a result, we have limited insights into the LSN structures and local users' behaviors. The LSNs are therefore worth exploring. In addition, people interact differently under different conditions, and users' behaviors may be quite different on a local scale (i.e., city level). For example, Marlow *et al.* found that online users' social structures change as their locations or environments change [75].

Moreover, there was research supporting that the prevalence of local communities can reveal important insights about the microeconomics of competition and the role of location in competitive advantage [88]. And the previous study stressed the importance of monitoring the dynamics of local communities. Therefore, this part of the thesis explores the local communities in LSNs.

This thesis explores the techniques to discover two different types of local communities. The first type is to discover transient local communities. Discovering transient local communities in dynamic OSNs allows us to discover local information (i.e., geographic information) faster. The discovered local information also helps scientists, marketers, and governments to understand users' social behaviors, adjust business models, and understand citizens' needs.

The second type is to discover local *Communities of Interest* (CoIs). Communities of interest are groups of people who share common interests or passions. CoIs have been applied in a variety of environments ranging from characterizing the online shopping behavior of individuals and routing local knowledge to detecting fraud in telephone networks [31, 35]. Discovering CoIs helps understand groups dynamics that can be used for various aspects.

Typical technique to identify communities in social networks is to cluster users. However, such clustering task is challenging because the user-shared information (e.g., messages) may contain errors (e.g., typographic errors). Moreover, communications between users change overtime. Thus, it is necessary to apply data cleaning and advanced data analysis technologies such *Text Mining* (TM) and *Social Network Analysis* (SNA) to extract desired information.

Social network analysis (i.e., graph analysis) and text mining (TM) are two main streams for community discovery in networks. There have been several researches combining graph analysis and text mining. For example, search engines such as Google and Yahoo used TM to rank search results and graph metrics to measure centralities of the hyperlink networks [26]. Another important TM plus SNA application was the summarization of texts by calculating centrality measures in word networks where vertices represent words, and vertices are connected if they appear in the same text unit [42]. Nevertheless, the focus of SNA and TM are quite different. The later sections introduce the related work in each research field.

1.4.1 Related Work for Identifying Communities in Social Networks

1.4.1.1 Social Network Analysis

This section provides SNA studies that are relevant to this thesis (i.e., the discovery of local communities in social media). Before applying SNA techniques to discover communities, one should denote the relationships between users (or objects) systematically in graphs. This section starts with the implementation of graphs, as it is the fundamental technique to prepare the OSN data for SNA, and then provides the overview of the related work.

Implementing Graphs

Graph is a formal way to depict a social network. It enables to depict huge datasets systematically. Graphic representations of interpersonal relations are sociograms, where vertices represent individuals and edges present the relationships among individuals. Edge weights depict the strengths of association between two individuals. Each vertex can have

several attributes such as location(s), share contents, and age, etc.

Since the number of studied vertices can be huge, it is impossible to visualize or analyze the data at the single-vertex level. There are measures for graphs that allow in-depth understanding of features and characteristics of graphs, called graph metrics. Graph metrics can be applied to static and dynamic graphs to derive relationships between vertices and understand patterns that could not be done by merely looking at the contents obtained from OSN data.

Different metrics allow us to derive different information. For example, density metric indicates the cohesion of network; and centrality metric (e.g., degree centrality, closeness centrality, betweenness centrality, eigenvector centrality) highlights the most important vertices of a graph. Other measurements such as path and diameter measure the connectivity of a graph. Advanced graph metrics such as clustering, cliques, enable the discovery of communities in a graph [53].

There are also other implementations for graphs. For example, adjacency matrix and adjacency list [39]. An adjacency matrix is a matrix to represent graph relationships, where rows and columns are labeled with graph vertices. The elements of the matrix denote the relationship between vertices. Adjacency matrices have been widely used for collaborative filtering techniques, such as item-based and user-based recommendations [18,106]. Although adjacent matrix enables fast retrievals for relationships (O(1)), it requires large space for storage (O(N $\times$ N)) and is not always applicable for large-scale SNA.

On the other hand, adjacency list consists of a list of vertices, and each vertex is the root of a linked list that connects the adjacent vertices. Adjacency list has been widely used for representing sparse graphs. However, it is slow in finding the presence or the absence of specific edge. Collectively, each implementation has its strengths and weaknesses, and the implementation of a graph is chosen based on research requirement.

Building Sociogram

When implementing sociograms, one typical question is given a set of vertices (i.e., users or individuals), how to establish edges and assign edge weights. There are two ways to assign edges: either using explicit social network properties or using implicit users' preferences.

Explicit social network properties can be telephone communications, email interactions within an organization, and mentions (the "@" mark) in tweets. For example, Adamic and Adar searched HP lab's email network by observing email interactions [16]. Kamath and Caverlee identified transient communities in Twitter by observing users' communications (i.e., mentions) [60].

Implicit user preferences can be user-shared topics, citations, and URLs. Author Co-citation Analysis (ACA) is one type of study that maps researchers to their corresponding research field by observing their common citations [100]. Web Co-link Analysis (WCA) is another type of study that maps researchers based on the hyperlinks/URL prefixes (e.g., http://facebook.com/, http://www.bbc.co.uk/news/) [61, 96].

From the above examples, we can conclude that building sociograms based on different properties allows us to observe different phenomena. Finding appropriate properties to study graphs is an interesting research question.

Discovering Communities

There are graph clustering models for discovering communities in a graph (e.g., sociograms). The graph partition approach consists in dividing vertices into k disjoint clusters of predefined size and minimizing the numbers of edges lying between clusters. For example, K-means iteratively re-classifies vertices into k disjoint clusters based on the distances to the centres of cluster. Kernighan-Lin algorithm swaps pairs of vertices to minimize external edges connecting different clusters [63].

The min-cut clustering algorithm (MCL) is also an example graph clustering algorithm. It models the network as water pipe and calculates the maximum flows from every vertex to an artificial vertex and find the minimum cuts to identify communities of each vertex [49].

Another type of graph clustering approach is hierarchical clustering, which displays several levels of grouping of clusters. There are two types of hierarchical clustering algorithms: agglomerative (bottom-up) algorithms [48] and divisive (top-down) algorithms [46,72]. Hierarchical clustering does not require a preliminary knowledge of the cluster size and number. However, discrimination between clusters heavily depends on the adopted similarity model, and clustering result may not be ideal when the hierarchical structure of graph is not obvious.

Instead of analyzing raw data itself, spectral clustering projects initial dataset into a feature space and clusters data in the feature space through standard clustering techniques, such as K-means clustering. Many spectral clustering models cluster graphs by using eigenvectors derived from different matrices. For examples, models proposed by Donath and Hoffman and Yang and Liu use eigenvectors of adjacency matrix [40, 102], while the model proposed by Fiedler uses eigenvectors of Laplacian matrix [43]. However, different authors disagreed on which eigenvectors should be used, and the exact computation of eigenvectors for large-scale data is impossible [45].

Modularity optimization is so far the most popular clustering approach for detecting communities in graphs, and the Newman-Girvan modularity is the most popular modularity model. It is based on the idea that a random graph is not expected to have a community structure [83]. By maximizing the total edge weight difference between the actual graph structure and the random graph structure using quality function Q, one is supposed to obtain the best clustering results. There are many modularity optimization approaches to optimize Q.

The first algorithm to maximize modularity is the greedy method proposed by Newman [81]. *Greedy Optimization of Modularity* (GM) is a hierarchical clustering method that assigns each vertex to a cluster, and successively merges two clusters to form a bigger cluster to maximize the value of Q. Since GM involves many redundant operations, *Fast Greedy Optimization of Modularity* (FGM) proposed by Clauset et al. enhances the GM algorithm

by using sparse matrix [34]. (This thesis has developed an enhanced algorithm based on the FGM algorithm to cluster users. Details of the new FGM-based algorithm can be found in section 4.3.)

Since the results of GM are not always accurate enough, other types of optimization algorithms, such as simulated annealing [65], extremal optimization [23], and spectral optimization [82], have been developed to improve the accuracy of clustering. However, these algorithms are very expensive models with high computational complexity. To improve the performance, Blondel *et al.* proposed a different greedy clustering model (i.e., Louvain method) [21].

Overall, generic graph-clustering algorithms (the algorithms mentioned above) are mostly computationally expensive because they consume a lot of memory. As a result, it is not applicable to apply generic graph-clustering algorithms to dynamic social networks that require frequent updates on results. In addition, the generic algorithms may fail when network size is too big.

Dynamic clustering algorithms and divide and conquer approaches (e.g., Google's Map Reduce) have been proposed to tackle this issue. For example, dynamic clustering algorithms have been enhanced based on the MCL algorithm and applied to dynamic social networks [19, 33, 60]. (This thesis has developed an enhanced algorithm based on MCL algorithm to cluster users. Details can be found in section 3.7.)

Saha and Mitra proposed an alternative dynamic algorithm for MCL [19]. The proposed algorithm maintains high quality clusters and re-clusters low quality clusters in presence of insertion and deletion of edge. Kamath and Caverlee further modified Saha and Mitra's model and proposed a sub-graph clustering algorithm to boost the performance of clustering [60].

Damping Social Network

However, in dynamic social networks, the strengths of association between two vertices

vary over time. As an OSN evolves, the connections between vertices may become stale and their graph-space distances may change. This problem must be addressed in order to analyze a dynamic network.

There are damping functions that have been applied widely to model decays in evolving networks for different purposes. Link-based ranking algorithms propagate page importance through links, and usually damping functions are required to damping link weights, so that a direct link implies more endorsement than a link through a long path. Linear, exponential, and hyperbolic decay functions were applied for ranking web pages [86, 103]. On the other hand, exponential decay was applied for removing stale connections in OSNs [60], and the damping function ensures that the connections preserved at certain timestamp are composed of users who have communicated with others recently.

In a dynamic social network, the majority of registered users are inactive. The users might either have only few connections to other users, or they were active in the past. By damping the users' strengths of communication (i.e., weights) at regular time intervals, one can identify currently active users while gradually remove inactive users who no longer have communications with others.

1.4.1.2 Text Mining

The previous section introduces the techniques for SNA, and the techniques focus on analyses in graph space. This section details the techniques for *Text Mining* (TM) that focuses on the analyses for text resources available in OSN data.

Text Mining is the discovery of new and previously unknown information by computers, which automatically extract information from a large amount of unstructured textual resources. As a result, TM can help extract relevant information from ONS data.

Text Clustering

Text clustering, also referred to as document clustering, is most commonly used approaches for identifying groups of similar documents in a corpus. Text clustering methods

can be used to automatically group the retrieved documents into a list of categories, as is achieved by enterprise search engines [30, 104]. If the documents belong to users, one can apply text clustering methods to group similar users [58].

Since text clustering usually involves analysis of words within documents, it is required to have data structures to store and represent the relationships between words and documents.

The classic structures are bipartite graph and Vector Space Model (VSM). A bipartite graph implementation is a graph that divides vertices into two disjoint sets, with one for documents and another for words. A VSM is referred to as bag-of-words model, which is a word-document matrix, where the rows correspond to words and columns correspond to documents. The entries of the matrix may be binary or frequency counts. By examining the degrees of intersection (e.g., common elements) between documents, one can determine how similar are the documents. Since VSM-based models are widely adopted in text clustering, the rest of the contents focus on the VSM-based approaches.

Text clustering groups similar documents into a coherent cluster and separate documents into different clusters if they are different. The clustering methods for TM are not too different from those graph-clustering methods mentioned previously. One may build a graph by assigning vertices with documents and assigning edges with the relationships between documents, and then apply hierarchical clustering model, density based clustering model, and graph partition models, such as K-means to cluster the documents.

In SNA, building graphs is most fundamental for graph clustering. To derive better clustering results, one needs to carefully select the attributes to establish edges. Similarly, in text clustering, one needs to carefully select the text similarity measure to establish the similarities (edges) between pairs of documents. For example, the density-based clustering models such as DBScan [76], rely heavily on the similarity computation.

To determine the similarities between documents, one can look at the text similarity measure as: given two bags of words (in two documents) Q and S, determine how similar

is Q to S. A variety of similarity or distance measures (VSM-based models) have been proposed and widely applied to measure the text similarities between documents. Examples of similarity measure are Matching, Dice; Overlap, and Jaccard Correlation Coecient; and Cosine Similarity (eq. 2.1 -2.5) [56].

The ideas of these measures are simply counting the number of words that occur in both bags and applying different ways to smooth denominators. However, different similarity measures score similarities differently, consequently the similarities calculated by using different similarity measures may not always be the same. Once the text similarities are determined, one can assign the edge weights with the similarities measured and run clustering algorithms to obtain a set of clusters.

$$Matching = |Q \cap S| \tag{1.1}$$

$$Dice = \frac{2|Q \cap S|}{(|Q| + |S|)} \tag{1.2}$$

$$Jaccard = \frac{|Q \cap S|}{|Q \cup S|} \tag{1.3}$$

$$Overlap = \frac{|Q \cap S|}{min(|Q|, |S|)} \tag{1.4}$$

$$Cosine = \frac{|Q \cap S|}{\sqrt{|Q| \times |S|}} \tag{1.5}$$

In addition to choosing similarity measures, in text clustering, it is also important to determine the frequency counts of word-document matrix's entries.

tf-idf (*term frequency - inverse document frequency*) that is most commonly used weighting method for TM, reflects how important a word is to a document in a collection or a corpus. It is often used as a term (word) weighting factor [92].

Since term weighting is performed to smooth the importance of all words (Some words are simply appear more frequently, and it is important to lower the impact of these words), one may have different weighting schemes to weight the importance of words within documents. For example, Srinivas *et al.* proposed their own weighting scheme for similarity measures.

Nevertheless, one can simply use *Term Count Model*, and the weights are just the counts of word occurrences.

Short Text Analysis

However, VSM-based approaches may fail when vector space is sparse or when there are synonyms and multi-faceted words. Miss matching is also another major reason that causes failures for VSM-based approaches. With the recent explosive growth of Internet use, social media, and blogging, short texts exist in different forms, e.g., messages, image captions, new feeds, tweets, etc. Short text is different from traditional documents in its sparse nature, typographic errors, and multi-faceted and semantically related words. These characteristics have hindered the performances of conventional machine learning and text mining algorithms [78]. For example, standard VSM-based approaches may fail when they are applied to short texts, because most of the words do not macth. Data enrichment and conceptual matching are therefore required to inflate information.

There are two main methods for data enrichment. The first method consists in identifying implicit relations between words by using web search engines such as Google [24, 91]; and the second method consists in using data repositories such as Lexical dictionary, WordNet, and Wikipedia [79, 84] to establish the relationship between words.

Continue the previous example (page 19), the text similarity measures of the two bags of words can be modified as follows. The sets Q and S can be constructed using the stemmed words instead of using the original words. Two words can be matched if they are synonyms according to the data repositories, such as WordNet. Then eq. 2.1 -2.5 can be applied to determine the similarities. However, a lot of multi-faceted words and semantically related words in short texts are not recorded in data repositories (e.g., new words). This problem limits the power of using data repositories.

Natural Language Processing

Natural Language Processing (NLP) is also an important part in TM because the shared

contents on the Internet are usually complex and difficult to extract useful information [28]. For example, Haythornthwaite *et al.* applied NLP techniques to model syntactical structures and extract nouns from online conversational contents [55].

NLP focuses on automated methods for natural human languages and includes what is generally referred to as *Computational Linguistics* (CPL). CPL computes statistics over large text collections to discover patterns for solving NLP problems. NLP problems cover sentiment analysis such as positive and negative comments; entity recognition such as identifying times, locations, events from emails; information extraction such as identifying user-interested attributes for products; and analysis of human language structures such as machine translations. There have been NLP packages such as *NLTK*[11] and *MALLET*[12] and software available to tackle sub-problems in NLP.

However, short texts such as tweets often contain highly irregular syntax and nonstandard use of English (as mentioned previously). These errors reduce the effectiveness of NLP techniques. As a result, sometimes it is required to convert short texts into a more standard form of English, so that standard NLP techniques can be more easily applied to them [62].

Collectively, SNA and TM have been widely applied on social media data analysis, and they have been applied to a diverse set of issues. Nevertheless, there are limited studies focusing on the discovery of transient local communities, as well as, the discovery of local communities of interest. The later sections first detail the challenges of the two research questions and then draw the research objectives of this thesis.

1.4.2 Challenges for Discovering Transient Local Communities

Identifying emerging communities is fundamental to discover emerging (transient) local communities that are formed in near real time. A transient community appears in certain time

[11]NLTK is a leading platform for building Python programs to work with human language data. http://nltk.org/

[12]MALLET is an integrated collection of Java code useful for statistical natural language processing. http://mallet.cs.umass.edu/

period under specific purpose in the networks. Information carried by transient communities varies depending on the reasons for forming groups. For example, *Communication-based Transient Communities* (CTCs) are groups of users who actively communicate with each other, and ToIs within communities may be widespread. *Location-based Transient Communities* (LTCs) are groups of users who are geographically close, and topics within communities are geographically associated with certain areas. We can also have interest-based transient communities, which are groups of users who share common interests.

Tobler's first law of geography has addressed *"Everything is related to everything else, but near things are more related than distant things"* [98]. Motivated by this law, this thesis aims to discover *Transient Local Communities* (TLCs) that is defined as short-lived collections of users that are frequently communicating with others and are also geographically close in networks.

Since the users belong to a TLC are mostly local people; they should tend to be concerned and talk more about local events, local politics, and local interests. They are also more likely to care about events in places near them. The discovered TLCs allow us to explore local topics and local interests more efficient. For example, Calgary hockey fans often care about Edmonton Oilers and Calgary Flames, and Calgarians watch ski hill events in Banff and Vancouver. However, there are three major challenges for discovering TLCs, and the current researches tackle each challenge separately. The challenges are as follows.

First, OSN data are usually large, "large" is employed here to refer to a network where the number of studied vertices is huge that it is impossible to visualize or analyze the data at the single-vertex level. Since the data volume is large, it is not applicable to frequently process all data to find the emerging connections and communities. There are dynamic clustering algorithms to cluster dynamic social networks. However, mostly these algorithms do not consider geo-locations [19].

Second, OSNSs support high update rates, so users can keep inserting new information, as

well as, communicating with others at anywhere, anytime. As a result, ToIs change quickly and the strengths of association between users vary over time. Consequently, it is difficult to identify current ToIs. Damping functions were applied to filter irrelevant connections over time [60]; however, there are limited researches considering damping connections (i.e., graph weights) when clustering networks.

Third, although there are algorithms for discovering CTCs [16] and LTCs (e.g., DB-Scan) [76]; there is no algorithm designed for discovering groups of users that are highly communicating and are also geographically close to group members. Partitioning the network based on one attribute causes problem in this research. For example, partitioning the network into communities based on frequencies of communication may result two conditions: 1) users that highly communicate within same groups may not be geographically close, and topics within groups may not be location specific; and 2) geographically close users may talk about similar topics in different groups.

To identifying TLCs in time evolving social media, this thesis proposes a model that integrates a damping function to filter irrelevant connections over time, a graph-clustering algorithm to identify communities, and a geo-location proximity algorithm to gather geographically close users.

1.4.3 Challenges for Discovering Local Communities of Interest

OSNSs provide platforms that allow people to interact and share information online, and information flows through the network. Routing information to the right audiences benefits information broadcasting and marketing. To route local information, it is necessary to determine who carry information and who are the audiences. The discovery of communities of interest help us identify the information carriers and audiences. There are also three challenges involved in this research, and the challenges are described as follows.

First, there are different types of user on OSNSs. Users' behaviors are different depending on the reasons for creating connections with others, and the shared contents vary according

to users' interests. Some of the shared contents and created connections are not helpful for identifying CoIs; instead, they make data noisy and make the network structure difficult to understand. For example, chatterboxes[13] and lifestreamers[14] may create connections or share contents without specific interests. Identifying relevant properties for data analysis ensures that the communities that will be identified are meaningful.

Second, users share many different types of information through OSNSs. For example, a user can share articles through short URLs or share short messages to describe events. Mostly, the shared contents are text-based and are unstructured. Since the shared contents may reflect users' interests, comparing the similarities between the contents may help determine the similarities between users' interests. Similarity measure is the most commonly used method in text mining to compare the text similarities between text-based contents. However, most widely adopted VSM-based approaches [27, 69] may fail when vector space is sparse (e.g., Twitter allows users to share less than 140 words for each tweet) or when there are synonyms and multi-faceted words [105].

Advanced text similarity measures involve short text analysis. However, short text analysis requires data enrichment to identify synonyms and multi-faceted words, and it also requires additional processes to handle syntactical problems. Normally, the data enrichment is computationally expensive, and it becomes unrealistic when the number of words being processed is huge. Moreover, text similarity measures disregard the communications (i.e., graph structures) between users.

Third, in large network data analysis, social networks can be represented as graphs. Graph clustering is another approach to detect CoIs in OSNs. However, results of purely graph-based clustering heavily depend on graph structures [21, 22]. In a real world social network, human interactions form a highly connected network where a giant component[15] connects most vertices together [90]. Without sampling the network and modeling the data,

[13] A chatterbox is a person who chats constantly

[14] A lifestreamers is a person who keeps updating his/ her daily life

[15] A giant component is a connected sub-graph that contains a majority of the entire graph's vertices

it is almost impossible to study the network. The highly connected property makes it difficult to discover CoIs by only analyzing graph structures.

Therefore, a hybrid approach that combines text mining with graph clustering is more desirable. This research proposes a clustering algorithm that integrates the fast-greedy optimization of modularity (FGM) with a text similarity measure to eliminate the noises created by meaningless connections.

In summary, the other two main contributions for this thesis are the two modified algorithms. The proposed algorithms overcome the problems of existing algorithms, and they enable the discovery of local communities that allow us to have more insights into the LSNs.

1.5 Research Objectives

To conclude, this thesis investigates three research questions: 1) how to design and implement a crowd-based GIS for transportation, 2) how to discover local communities and local topics, and 3) how to discover communities of interest in LSNs.

First research is chosen because the existing mobile applications have limitations, which have hindered the integrations with existing ITSs. In addition, the server architectures for those existing mobile applications are not available to the public. This thesis proposes an example design of the essential components for an adequate GIS for transportation (both mobile and server application).

The other two researches are selected because there are limited studies focusing on the discovery of transient local communities and local communities of interest. Transient local communities can potentially help us understand group dynamics and explore local topics and local interests more efficient; and local communities of interest can potentially benefit efficient information broadcasting and marketing.

In this thesis, TrafficPulse, a complete front-end and back-end GIS for transportation, is designed to provide a reference to readers who want to create similar systems for trans-

portation. This thesis also proposes two algorithms to discover transient local communities and local communities of interest in LSNs.

The remaining chapters of this thesis are organized as follows. Chapter 2 details the design and the system architecture of the crowd-based GIS for transportation. Chapter 3 details the algorithm proposed to identify transient local communities, and Chapter 4 details the algorithm proposed to discover communities of interest in LSNs. Chapter 5 provides the experimental results. Finally, conclusions and future work are drawn in Chapter 6.

Chapter 2

TrafficPulse

2.1 System Architecture

This chapter details both the front-end and the back-end system architecture of the GIS designed for transportation. The main contribution of this part of the thesis is the design and the implementation of TrafficPulse, the GIS that harnesses the ubiquitous Smartphones to overcome the limitations of existing systems for transportation. The proposed architecture provides reference to readers who are interested in creating similar systems. This chapter starts with TrafficPulse's core idea.

TrafficPulse is designed to assemble traffic information provided by Smartphone users. TrafficPulse allows users to contribute spatiotemporal traffic information and visualize the dynamics of urban traffic through the use of Smartphones. The idea of TrafficPulse (Figure 2.1) is to collect real-time traffic information shared voluntarily from mobile users, because people driving and walking on roads know the dynamics of traffic. Traffic data collected by TrafficPulse can also enable ITS researchers to develop new traffic estimation models and new traffic control logics.

As shown in Figure 2.1, TrafficPulse's front-end can collect both opportunistic (e.g., sensor data, geo-locations) and participatory (e.g., user-shared contents) sensing data, and offline GPS positioning and wireless positioning locations. After obtaining the volunteered information, the GIS would be able to conduct social network analysis, crowd analysis, and sensor data analysis etc. to derive patterns. Then the results could be used to provide different recommendation services to users. In this way, TrafficPulse could benefit the public by using the collected VGI and provide a pain-free approach to solve traffic problems.

This research has designed and implemented the TrafficPulse architecture. Figure 2.2

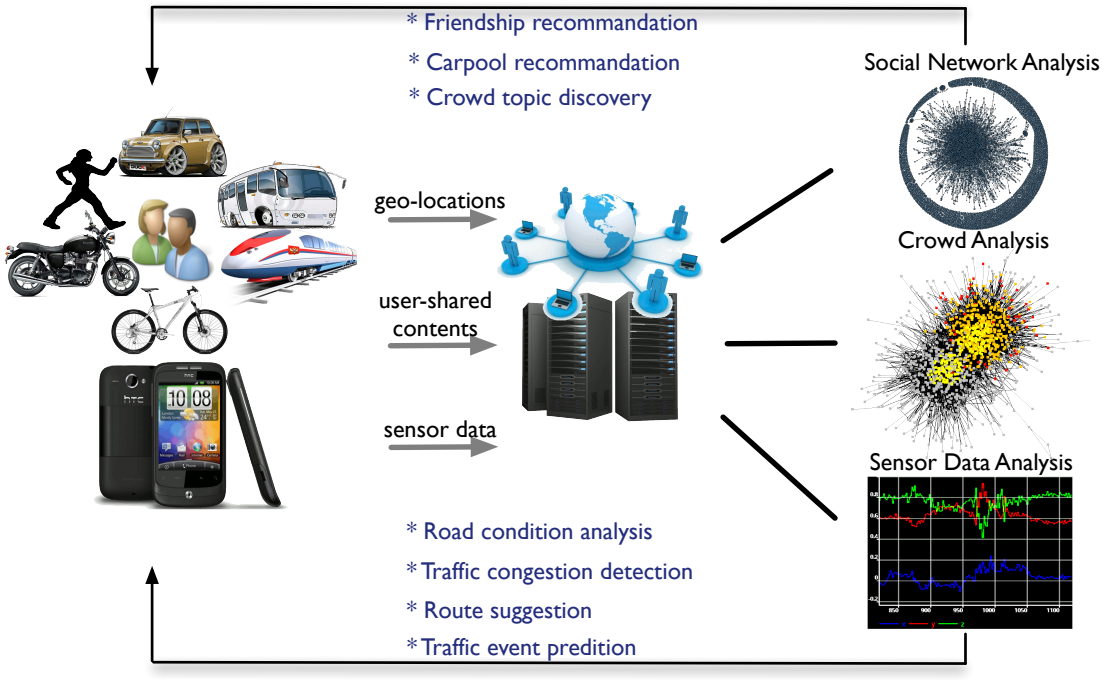


Figure 2.1: TrafficPulse's Idea

From: Li, R. Y., S. H. L. Liang, D. W. Lee, and Y.J. Byon. TrafficPulse: A Green System for Transportation, The First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems 2012 (MobiGIS2012), Redondo Beach, California, ACM Digital Library, 2012. Image used with permission.

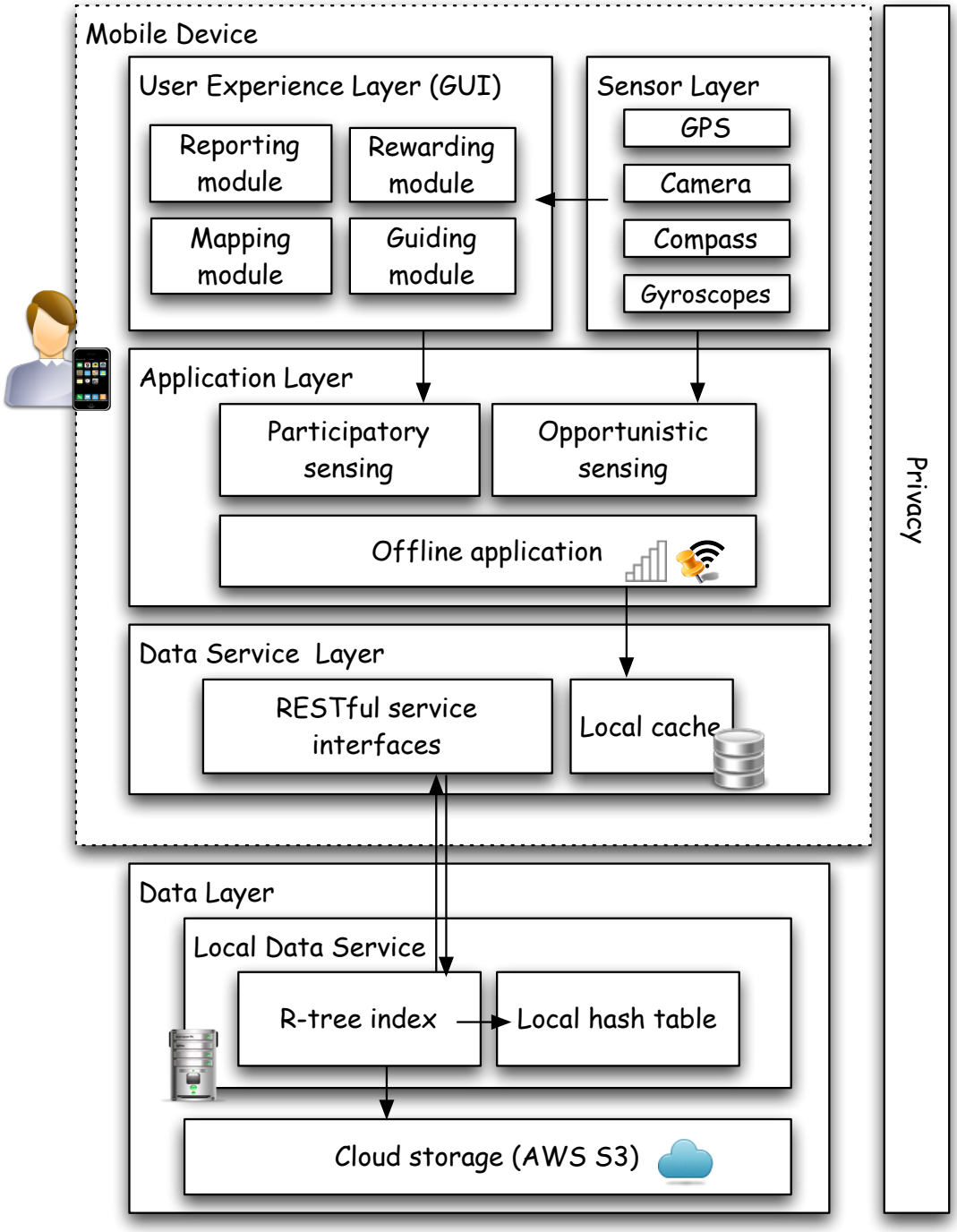


Figure 2.2: TrafficPulse's Architecture

From: Li, R. Y., S. H. L. Liang, D. W. Lee, and Y.J. Byon. TrafficPulse: A Green System for Transportation, The First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems 2012 (MobiGIS2012), Redondo Beach, California, ACM Digital Library, 2012. Image used with permission.

illustrates the architectural diagram. TrafficPulse's front-end (an Android mobile application) consists of *User Experience Layer*, *Sensor Layer*, *Application Layer*, and *Data Service Layer*, and its back-end (i.e., Geopot server) is the *Data Layer*.

User Experience Layer provides different modules that allow users to publish and visualize traffic data. Sensor Layer represents the Smartphone sensors used in this research. Application Layer manages the executions of applications, including opportunistic sensing tasks, participatory sensing tasks, and offline data collection. Data Service Layer handles security issues, as well as request/response to/from the back-end server. Finally the Data Layer (i.e., Geopot server) handles data storage and manages data insertions and retrievals. To describe the architecture in details, this chapter divides the discussion into two separate parts: *Client* and *Server*.

2.2 Client

The client (i.e., front-end) is designed to accomplish three tasks: 1) to motivate users to share traffic information, 2) to collect useful traffic information, and 3) to help users make decisions efficiently.

In order to motivate users to contribute traffic data, TrafficPulse is designed to have a user friendly GUI, which is easy and intuitive to use. In order to enable TrafficPulse to collect useful traffic information from Smartphone users, the support features have been carefully selected. TrafficPulse's front-end is also designed to be an extensible application, so that more value-added features can be supported.

2.2.1 User Experience Layer

There are four modules in the User Experience Layer, and each module has its main function. Their functions are described as follows:

2.2.1.1 Reporting Module

The reporting module allows users to share sensor data, current road conditions, and events to the public through participatory and opportunistic sensing.

Users can report traffic information at different levels of detail through participatory sensing methods. The GUI is shown in Figure 2.3. For example, users can click the buttons to report different road conditions (Figure 2.3 (a)), or users can comment current traffic conditions using the text boxes (Figure 2.3 (b)). Users can also take photos to record traffic conditions and then share the photos to the public (Figure 2.3 (c)). These VGI is published to server along with geo-location.

In addition to participatory sensing methods, users can also share traces and sensor data through opportunistic sensing. The sensor data, such as geo-locations, speeds, and accelerations are automatically collected and published whenever users turn on "auto mode" (Figure 2.3 (a) the green button). Again, this VGI is published to the server along with the geo-location.

2.2.1.2 Rewarding Module

The rewarding module is designed to create incentives for users to continuously use TrafficPulse. *Gamification* is the use of game design techniques that applies to non-game applications, in order to encourage people to adopt the applications [38]. Examples of systems that are based on giving reward points to users who share information on location-based platforms are Foursquare, and Gowalla. Motivated by these location-based services, TrafficPulse's front-end is designed to be presented as a "Green Karma" game.

Users are rewarded with "Green Karma" points every time they contribute traffic data. The idea behinds the "Green Karma" mechanism is to make users feel that they are doing something good to the society. Users' Green Karma values are accumulated and ranked, so that users can compete to be the "Green-est" (Figure 2.3 (d)). In order to make the game more interesting, TrafficPulse updates the colors of Green Karma icons accordingly based

on users' scores.

2.2.1.3 Visualization Module

The visualization module is designed to help users make informed decisions. This module provides four mapping functions: 1) mapping near-by users, 2) mapping latest trace, 3) mapping near-by public transit stations, and 4) mapping events (e.g., accident, traffic congestion).

The first function updates users' current traffic conditions by visualizing the number of current near-by users based on users' locations. Walking users should be able to update their near-by traffic conditions at anywhere anytime. However, when users are driving and may not be able to use Smartphones, they can get helps from the passengers (similar to using Google Maps and Waze).

The second function visualizes users' latest traces and times spent on their last travels, so that they can manage their times on each travel more efficiently. This feature was designed at the time when other similar systems (e.g., Waze) did not support this type of visualization. However, viewing traces were common features supported by cycling applications such as Cycle Watch [1]. Motivated by these location-based services, we considered this function when designing TrafficPulse.

The third function displays near-by transit stations on Google Maps. Although Google Maps allow users to view transit stations, the maps do not highlight the stations. Therefore it is difficult for users to determine how many stations are around at the moment. The third function allows users to visualize the relative orientations and distances between stations and users' current locations. The visualization helps users find the nearest stations faster.

The fourth function displays traffic events shared by users on Google Maps. Waze supports a similar function. This function enables users to update their near-by traffic events easily. Similar to the limitation of the first function, when users are driving and may not be able to use Smartphones, they can get helps from the passengers.

These four mapping functions mainly help users to make better decisions (Figure 2.3 (e)-(f) and Figure 2.4 (a)). We considered privacy issues when designing these functions. As a result, users can only view their latest trace; points (users' geo-locations) of near-by users; and texts (text contents) of events. Without showing users' names and identifiers, the personal information can be secured.

2.2.1.4 AR Guiding Module

The AR guiding module is designed to navigate users through AR. AR is a type of virtual reality that duplicates world's environments in computer. Mobile AR overlays information on Smartphones' camera screens, and the displayed information updates while users are moving.

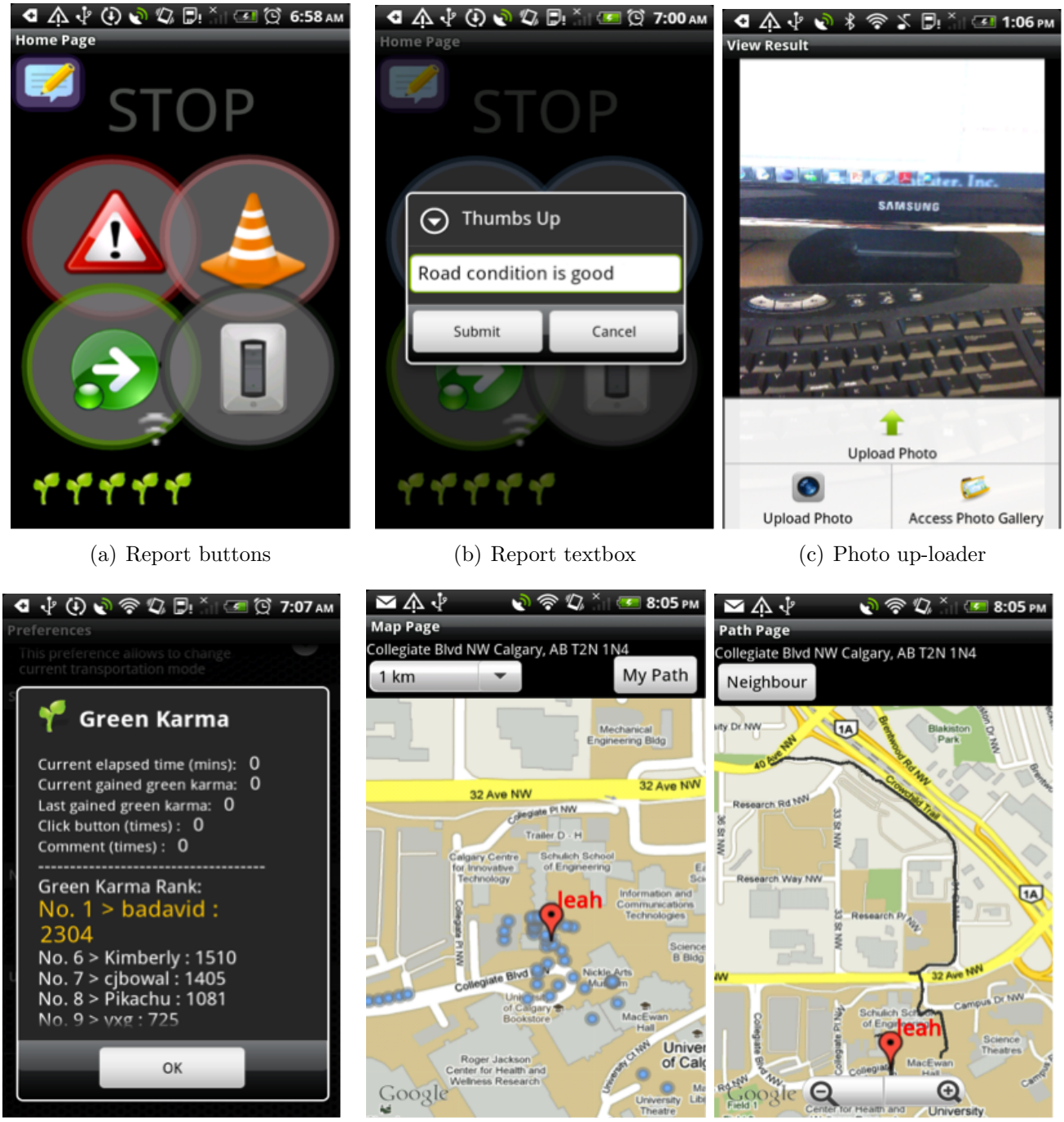
This AR module allows users to visualize near-by public transit stations and other relevant information, such as distances from users' locations to stations on their camera screens. Since the information is perceived through the way human live, this feature benefits those who are not map-readers and those who are new to a place and do not have enough sufficient knowledge.

Users can also use the public transit AR (Figure 2.4 (b)) and the mapping near-by public transit stations function (mentioned in the previous paragraph (Figure 2.4 (a))) together at the same time, so that they can find stations efficiently. A future work can be to increase the usability by including bus schedules and other relevant information, such as weather information.

2.2.2 Sensor Layer

Mobile sensors collect several different types of sensor data. Each Smartphone has a GPS receiver, a camera, a compass, and several gyroscopes and accelerometers.

A GPS receiver collects geo-locations, which can be used for checking in, recording traces, and marking locations of events. And the speed value derived from the GPS receiver can



(a) Report buttons (b) Report textbox (c) Photo up-loader

(d) Green Karma (e) View near-by (f) View latest trace

Figure 2.3: Screenshots of TrafficPulse GUI - 1

From: Li, R. Y., S. H. L. Liang, D. W. Lee, and Y.J. Byon. TrafficPulse: A Green System for Transportation, The First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems 2012 (MobiGIS2012), Redondo Beach, California, ACM Digital Library, 2012. Image used with permission.

(a) Map-based view of public transit stations

(b) Augmented Reality of public transit stations

Figure 2.4: Screenshots of TrafficPulse GUI - 2

From: Li, R. Y., S. H. L. Liang, D. W. Lee, and Y.J. Byon. TrafficPulse: A Green System for Transportation, The First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems 2012 (MobiGIS2012), Redondo Beach, California, ACM Digital Library, 2012. Image used with permission.

be used to detect user's transportation mode. If the user were driving fast, the speed values would be high. TrafficPulse triggers locking screens when it detects driving modes (i.e., higher speed values). Upon locking screens, users can view maps, but they cannot interact with TrafficPulse when they are driving.

A camera provides a different way to record real-time events. Users can report real-time traffic events by taking photos of them. A camera can also be used for generating AR applications.

A compass can not only display the direction of true north, but also collaborate with a GPS receiver, three gyroscopes, and a camera to realize AR applications. By knowing the readings of compass and gyroscopes, one can transform AR objects' true north coordinate system to camera's coordinate system, so that the information can be correctly displayed on the camera screen.

Accelerometers record real-time accelerations of a mobile device. The accelerometers can collaborate with other sensors to differentiate different road conditions. There were researches using accelerometer data to detect road conditions [80]. However, differentiating different road conditions is out of the scope of this research. TrafficPulse collects accelerations, but it does not consider any applications using the accelerometers.

In TrafficPulse, the Sensor Layer works closely with the User Experience Layer, and these sensors support the sensing tasks designed in the User Experience Layer. We can conclude that utilizing different sensors in Smartphone enables better understand of traffic dynamics.

2.2.3 Application Layer

The Application Layer manages three types of sensing tasks: opportunistic sensing tasks, participatory sensing tasks, and offline data collection including GPS and wireless positioning data.

2.2.3.1 Opportunistic Sensing

The opportunistic sensing in TrafficPulse refers to tasking users' sensor-equipped Smartphones to report information from their locations. TrafficPulse users can activate the opportunistic sensing mode on their GUIs, and the system will automatically collect sensor data, such as geo-locations, speeds and accelerometer data and transmit them to the Geopot server. Such numerical data can be further processed to derive useful information such as users' paths, average speeds etc.

2.2.3.2 Participatory Sensing

The participatory sensing allows users to participate sensing tasks and to report real-time traffic information around them at different levels of detail. As described in the User Experience Layer section, a user can report whether the traffic is moving smoothly by clicking buttons, texting comments, and taking photos of current traffic conditions. The semantic data can describe events that are not observable through conventional sensing approaches.

Example of "non-observable" events can be the breast cancer race in Calgary. By analyzing the sensor data collected by Smartphones, one can only observe the slow traffic movements, whereas the participatory sensing data provides the contextual information of that location, and therefore the data can help understand real-time events and monitor potential hazards, etc.

2.2.3.3 Offline GPS Location and Wireless Location Data Collection

As concluded in Chapter 1, collecting offline GPS location and wireless positioning data a critical feature for a system like TrafficPulse. Therefore, TrafficPulse targets not only online users, but also users who are temporarily offline or cannot temporarily access GPS signals.

TrafficPulse grants permissions to Smartphones to use both GPS and wireless positioning, so that Smartphones can collect location data provided by both. To store the offline GPS positioning data, TrafficPulse' front-end employs Smartphones' local caches to accomplish this job.

Whenever user reports a traffic condition, TrafficPulse checks whether the Internet is accessible, and if not, it activates an offline process (Figure 2.3 (a) grey Wi-Fi icon). Upon activating an offline process, the system stores the user's offline data temporarily in the Smartphone's local cache. When the Smartphone accesses the Internet, the stored data will be uploaded to the server.

TrafficPulse allows users to collect different types of information through different ways. However, the traffic data collected are not always reliable due to the weak GPS and wireless signals, system errors, or mistakes. For the collected numerical data, post processing is required. For example, in order to remove errors in a user's trace (i.e., a series of geo-locations), TrafficPulse examines the ratios of distance and time-interval between near-by geo-locations in a trace and eliminates points with ratio that lies outside a reasonable range.

In addition, the collected semantic data may contain errors as well. The errors could be both intentional and unintentional. Most of the crowdsourcing platforms face a similar problem. However, an early study conducted by IBM researchers in 2003 [51] examined the reliability of Wikipedia (the free encyclopedia that everyone can edit) and found that "*vandalism is usually repaired extremely quickly – so quickly that most users will never see its effects*". Although TrafficPulse has not been a peer-reviewed platform yet, this research assumes that this data reliability problem can be mitigated by techniques employed by other systems, such as Wikipedia. And the implementation of these techniques is out of the scope of this MSc thesis.

2.2.4 Data Service Layer

2.2.4.1 RESTful Service Interface

The design of Data Service Layer follows the fifth theme of Web2.0: lightweight-programming models. Any client can communicate to the TrafficPulse server (i.e., Geopot server) with HTTP-based RESTful APIs. To secure user-provided information and prevent user-shared

contents from being acquired for illegal use, TrafficPulse requires authentications for client-server communications.

When user logs in the system, the system coverts the user's password into an encrypted value using MD5[1]. Afterwards, a secret key (i.e., private key) is generated at the server side according to the MD5 value. Whenever the client sends requests to the server, the client uses the secret key for authentications.

Since the secret key is only valid before logging out, and a new key will be generated whenever the user log in the system. In this way, TrafficPulse can secure users' information and communications by frequently update the secret keys.

2.2.4.2 Local Cache

The local cache of a client consists of two parts: a preference document (i.e., PreferenceActivity in Andorid) and an in-memory SQLite database (i.e., SQLiteDatabase in Android). A preference document uses key-value pairs to store and retrieve user's preferences. Whenever user updates the contents in the preference panel (i.e., the Settings in TrafficPulse), the system updates the key-value table. The system retrieves the user's preferences and updates the GUI when the user returns to TrafficPulse mobile application.

An in-memory SQLite database is a small relational database for an Android application to store a collection of data items that can be organized as a set of formally described tables. TrafficPulse's front-end uses the SQLite database to temporarily store offline data and locations of public transit station. As mentioned previously, when Smartphone user cannot access the Internet, TrafficPulse stores the offline GPS locations at the local database (i.e., local cache). Whenever the Internet is accessible, the offline data in the SQLite tables are uploaded to the server, and the tables that store offline data are cleaned. Besides, to make sure the public transit stations stored in the database are near-by the user's current location; the tables that store public transit stations are updated according to the user's

[1]MD5 (Message-Digest Algorithm) is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value

geo-location.

2.3 Server: Geopot

To turn mobile devices into sensor probes and collect high spatiotemporal resolution traffic data, users should be able to check-in locations and search near-by traffic conditions at anywhere anytime. The scalability becomes a desirable property of the TrafficPulse system.

Naïve solutions to maintain high scalability is to deploy more powerful servers to handle frequent requests and responses. However, it is not economical to use expensive servers, especially for research organizations. In addition, users' requests are location specific; it is also desirable to couple the data storage with spatial indices.

Conventional spatial databases may solve the problem. However, the databases may fail to meet the demand of mobile systems because of the relational operations and the increasing number of stale tuples. The queries of spatial relational databases involve a lot of relational operations such as *spatial join* and *contains*, and these operations become expensive operations when the data sizes grow. In addition, relational databases mix stale and fresh tuples in the databases, the stale tuples degrade the query performances when the data sizes of databases increase.

To keep high query performance, to minimize the cost of building and maintaining data storage, and to keep up with fresh geographic information in the database, Geopot has its unique server architecture that tackles these issues.

Geopot [70] is designed to utilize existing Cloud storage and coupled data storage with spatial indices. Figure 2.5 demonstrates the implementation of the Geopot server. The server has two parts: a local data service and a Cloud-based data service (i.e., Amazon Simple Storage Service). To maintain the overall performance of the data service when data accesses from users increase, instead of storing end user raw data in the local data service, Geopot forwards them into an external Cloud storage, so that the clients can access raw data

directly from the Cloud storage rather than from the local data service. This architecture alleviates the burden to provision resources when data size grows and keeps the size of the local data service as small as possible.

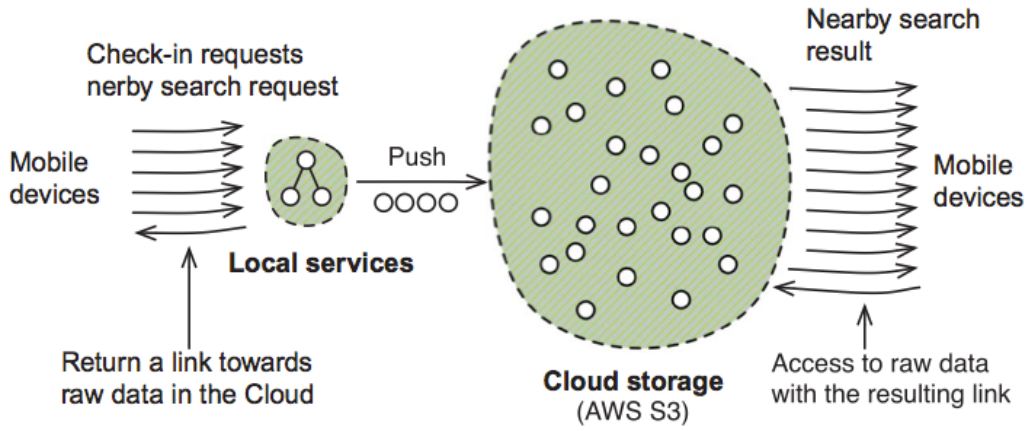


Figure 2.5: Geopot Scale-out Architecture

From: D. W. Lee and S. H. L. Liang. Geopot: a Cloud-based geolocation data service for mobile applications, International Journal of Geographical Information Science, 25(8):1283-1301, 2011.

Since Geopot aims to minimize the size of the local data service, the local data service contains only local hash tables, which store fresh data and lightweight spatial indices, which store the keys to access raw data in both the local hash tables and the Cloud Storage.

Figure 2.6 demonstrates the workflow of Geopot. Users publish their VGI through RESTful APIs (2.2.4.1 RESTful Service Interface), and each request contains the geo-location (i.e., latitude and longitude) and the metadata required to query the server. Geopot server then receives users' requests and distributes them evenly to the internal threads by using sharding, which partitions incoming requests to internal threads horizontally [36]. The reasons of using internal sharding for the local data service are to prevent internal locking of each thread and to make the system extendable to add more shards.

Each thread has its R-tree for spatial data indexing. The reasons for deploying R-tree to each internal thread are to make the server highly concurrent and to prevent lock contentions occurring when multiple threads access an R-tree at the same time.

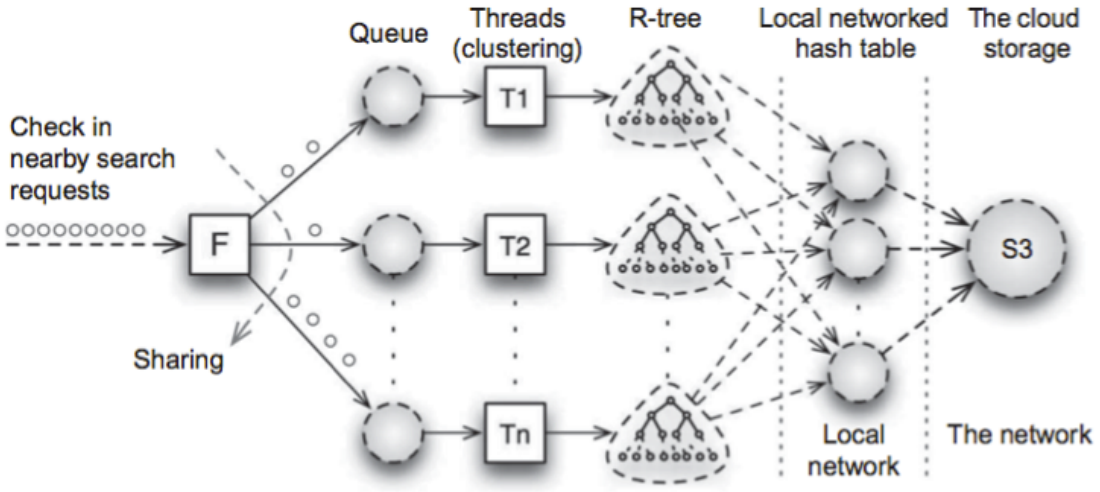


Figure 2.6: Geopot Server Implementation

From: D. W. Lee and S. H. L. Liang. Geopot: a Cloud-based geolocation data service for mobile applications, International Journal of Geographical Information Science, 25(8):1283-1301, 2011.

However, searching R-trees becomes inefficient when the lengths of the trees grow (R-tree is a balanced search tree). In order to maintain lightweight spatial indices, instead of inserting raw data into the R-trees, Geopot inserts only metadata of clusters that represent groups of geographically close data into the R-trees (Figure 2.7).

A metadata consists centroid coordinates of cluster ("lat", "lon" in Figure 2.7); a hash key generated by the centroid coordinates of the cluster ("hash" in Figure 2.7), and the radius encompassing the cluster ("r" in Figure 2.7). The centroid of the cluster is used as a spatial index of R-tree, and the hash key generated is used to access the local networked hash table(s) in which the members of the cluster are temporarily stored.

Geopot performs a modified online Leader-Follower method [41] to group geographically close data into clusters, because it has been observed that using online clustering is more effective than the K-means [74] clustering method. Graph partition approaches such as K-means need a fixed number of clusters in advance, and the approaches may fail to update clusters correctly when location points are dispersed throughout the study area. The modi-

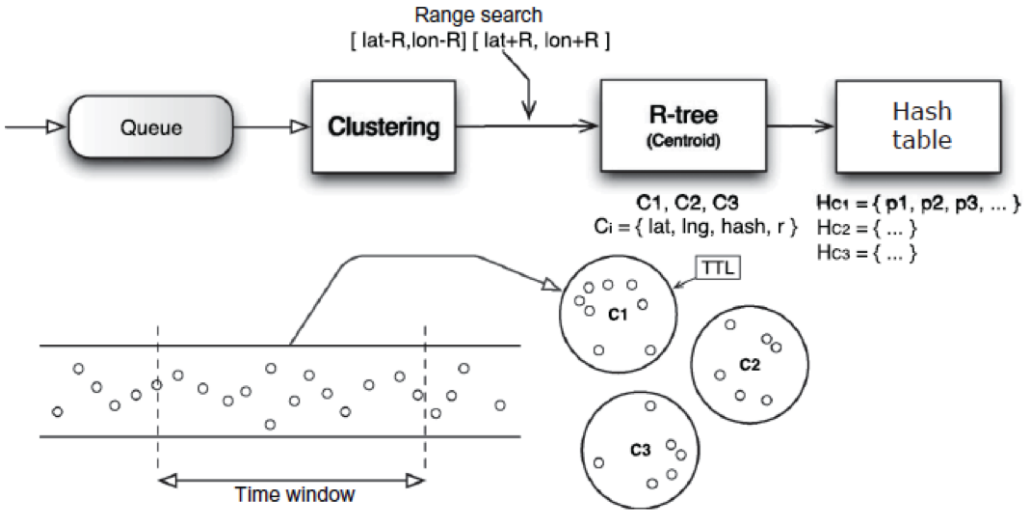


Figure 2.7: Geopot R-tree Index Method

From: D. W. Lee and S. H. L. Liang. Geopot: a Cloud-based geolocation data service for mobile applications, International Journal of Geographical Information Science, 25(8):1283-1301, 2011.

fied solution spontaneously creates a new cluster if no current cluster is sufficiently close to incoming data; otherwise, it simply joins the data into current clusters, as well as, alters the cluster centres.

This architecture is to ensure that each R-tree is always a compact size that fits into the main memory to keep pace with the growing data volume. Also because it is impossible to store all VGI in the local data service as the data volume increases, the local data service is designed to store recent data only (i.e., data within a particular time window).

In order to remove the stale tuples from the local data service, it is necessary to record the expired timestamp (i.e., Time-to-Live) for each cluster. To achieve this, when a new cluster is created, a corresponding Time-to-Live ("TTL" in Figure 2.7) value is attached to the cluster. A TTL value is used to update a cluster when its content in local hash table is regarded stale. Hash tables of clusters are deleted when they are expired. Upon deleting a local hash table, the contents of the local hash table will be published into the external Cloud storage.

Please note that the Geopot design is not part of the original contribution of this thesis. Geopot has been designed and implemented by the GeoSensor Web Lab. However, Geopot is an integral part of the TrafficPulse system. Therefore a brief introduction is included here. Interested readers can refer to [70] for more details.

2.4 Experimental Results

A data collection campaign was held, and the campaign recruited 50 students on University of Calgary campus to be TrafficPulse users to collected VGI. Since the TrafficPulse team has developed the back-end, and the evaluations can be found in [70], this thesis only evaluates the mobile VGI application.

Results show that the trace data collected from TrafficPulse users reflects most of the city's road network (Figure 2.8). Due to lack of battery power and GPS signal blockage, data streams contributed by users can be either continuous or discrete. Nevertheless, results imply that as the number of users grows, TrafficPulse is able to collect much higher resolution traffic data.

Figure 2.8: Crowd-sourced GPS Traces

Chapter 3

The Discovery of Transient Local Communities

As mentioned previously, discovering TLCs (transient local communities) allows us to explore local topics and local interests more efficiently. In order to identify TLCs in time evolving social networks, this thesis has proposed an algorithm that integrates a damping function to filter irrelevant connections over time, a graph-clustering algorithm to identify communities, and a geo-location proximity algorithm to gather geographically close users. This chapter starts with the definition of time-evolving social network in this thesis.

3.1 Definition of Time-evolving Social Network

This thesis defines the time-evolving OSN as an undirected graph G. $G(t) = (V(t), E(t))$ is a snap shot of G at time t. $V(t), E(t)$ are sets of vertices and edges at time t, where each vertex $v = (id, g(id))$ corresponds to users who have an identifier (id) in a social network with geo-location $g(id)$ ($g(id)$ can be null). Each edge $e = (u, v, w_t(u, v))$ represents the communication time between two users u and v with weight equals to $w_t(u, v)$ (Figure 3.1).

3.2 Data

This research chooses Twitter data near Calgary area as resource for evaluations. The tweets were collected using Twitter Search API with geo-location and relevant tags such as "yyc" and "calgary ". The purpose is to collect tweets about Calgary but shared by people from different places. Since users who are not local people can also be concerned about issues related to Calgary or highly communicate with users from Calgary. This type of data set could help assess whether the proposed methodology allows us identify more geographic information.

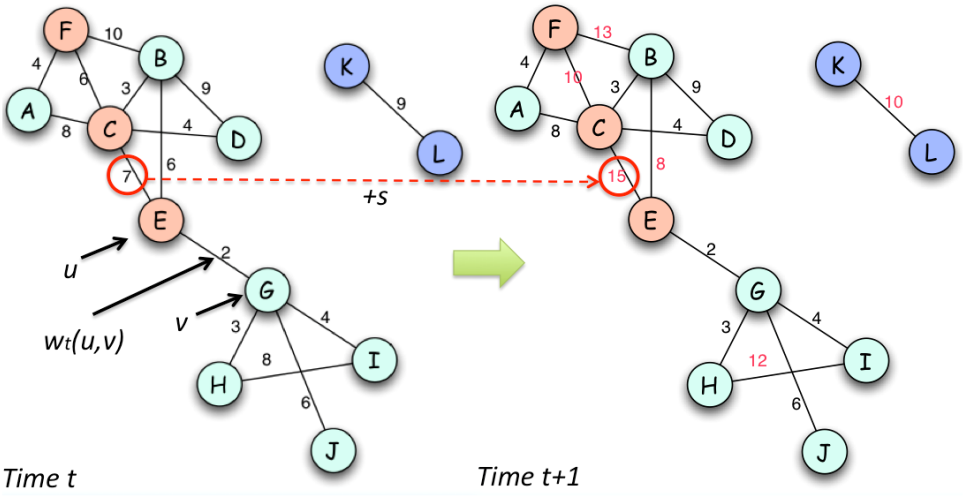


Figure 3.1: Time Evolving Graphs

As mentioned at the beginning, each user is attached with a $g(id)$ (geo-location). If the collected tweets have metadata that contain geo-locations, the algorithm assigns the geo-locations to the users who contribute the tweets. Otherwise the algorithm assigns users' geo-locations by mapping their hometowns recorded in their personal information using GeoNames APIs [4]. Limited by the spatial resolution of the hometown information, this research only examines city level phenomena.

This research uses 200-day Twitter data to test the algorithm. The time period of the sample of tweets is between November 19^{th}, 2010 and June 8^{th}, 2011 (see Table 3.1 for details). The dataset consists of 58,697 users and 211,030 updates ("tweets"). Figure 3.2 shows the profile of the tweets. As shown in Figure 3.2, the dataset is non-continuous. It is continuous from November 19^{th} to November 30^{th}, 2010 (day 0-11), January 21^{th} to January 30^{th}, 2011 (day 63-71), March 21^{th} and 22^{th}, 2011 (day 121-122), April 15^{th} and 16^{th}, 2011 (day 145-146), May 16^{th} to May 19^{th} 2011 (day 176-179), and May 26^{th} to June 8^{th}, 2011 (day 186-200). Although the data set is mostly non-continuous, it does not affect the validity of the experiments for testing the efficiency of the algorithm in terms of discovering

geographically close users and geographic information (details in Chapter 5.1).

In Twitter messages, users can employ @⟨*username*⟩ within tweets to reference other users. This research regards a message sent by user u_1 including @⟨u_2⟩ as a communication between users u_1 and u_2. This research has built an undirected graph by establishing edges between all pairs of u_1 and u_2 discovered and assigning the communication times between users as weights. In the 211,030 tweets data, the total number of tweets that contain @⟨*username*⟩ syntax is 75,510, and the total number of user pairs forming these tweets is 83,981.

Table 3.1: Twitter Dataset

Property	Total
Users	58,697
Total Tweets	211,030
Messages(@⟨u⟩)	75,510
User Pairs	83,981

3.3 Exponential Decay for Social Network

Figure 3.1 denotes an example of time-evolving social networks. The figure shows that the strengths of association between some users change overtime (e.g., vertex C and vertex E), while some do not. Users who long longer communicate with other users do not provide new information. In order to explore the new interests and topics in LSNs, the stale connections have to be removed. This research has applied the exponential decay algorithm proposed by [60] to remove the stale connections. The exponential decay algorithm is described as followings.

Suppose that the latest communication time between two users u, v in a network (graph) is $\tau(u, v)$, and T_{now} is the current system time. For each edge that has weight $w_t(u, v) > 0$, the weight in the next time interval can be updated with the equation:

$$w_{t+1}(u, v) = w_t(u, v) + s - \log(T_{now} - \tau(u, v)) \times \xi \tag{3.1}$$

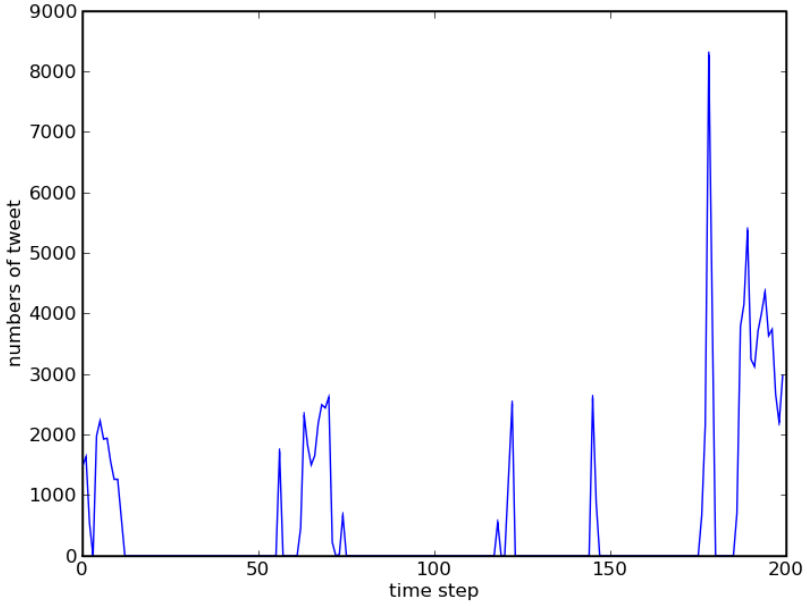


Figure 3.2: Twitter Data Profile
The data set is mostly non-continuous

In eq. 3.1, $\log(T_{now} - \tau(u, v)) \times \xi$ is the damping function. s is the total number of messages exchanged between time t and $t + 1$. ξ is the damping coefficient that controls the power of damping. The larger the value of ξ, the faster the inactive connections will be removed. In addition, the damping time interval is another factor that controls the damping efficiency.

The damping time interval needs to be carefully chosen depending on the graph connectivity, because even though the damping coefficient is small, the damping function decays the graph fairly quickly if the damping time interval is small. For example, if one updates the graph every one-hour when the majority in the graph updates one message every day, after twelve hours, most edges in the graph are removed. On the contrary, during the World Cup events, users exchange messages frequently. In order to keep up with fresh topics, the decay time interval should be small. This research has setup an experiment to examine the appropriate damping coefficient and damping time interval. Details are provided in Chapter

5.1.1.

3.4 Geospatial Locality Clustering

After removing inactive users, the proposed algorithm has discovered temporal clusters representing strongly communicating and geographically close communities in the built sociogram. The proposed algorithm enhances a MCL-based clustering algorithm for clustering the time evolving sociogram. Before detailing the proposed algorithm, the next sub-section starts with the definition of the MCL algorithm.

3.4.1 Min-cut Clustering Algorithm (MCL Algorithm)

The problem of graph partitioning consists in dividing the vertices in graph into groups, such that the connections between the groups are minimal and the connections within the groups are maximal. Identifying the minimum cuts to iteratively separate a graph into two sub-graphs is one way to approach graph partitioning. Since the MCL algorithm has been modified, and the modified algorithms have been applied to the time-evolving networks, this research continues the previous works and proposes a MCL-based algorithm for discovering TLCs.

MCL algorithm [44] is based on minimum cut trees, which were defined by Gomory and Hu [49]. The theory of minimum cut trees states that for any graph G, there is a minimum cut (min-cut) tree T_G of G. The min-cut tree is formed by recursively inspecting the path that connects two vertices s (source), t (sink) in G and finding the minimum cut to separate them.

In graph theory, a minimum cut of a graph is a cut, who's cut-set has the smallest sum of weights. That is, for two subsets S and T of V (V is a set of vertices in G), such that $S \cup T = V$, $S \cap T = \emptyset$, $s \in S$ and $t \in T$, one can find a cut $c(S,T)$ in G that separates s and t, and the sum of the weights of the edges crossing the cut has the minimum values. The

sum of the weights of the minimum cut is the weight of the tree edge that connects s and t.

The maximum-flow minimum-cut theorem states that in a flow network (i.e., G), the maximum-flow passing from a source to a sink is equal to the minimum capacity (minimum-cut) that is formed when removing in a specific way from the network causes the situation that no flow can pass from the source to the sink.

By finding the maximum-flow/ minimum-cut values between all pairs of vertices in the network, one can determine the maximum-flow between vertices. Gomory and Hu have observed that with n-1 cuts (n is the number of the vertices in G), one can obtain a tree T_G that is flow-equivalent to G, and the T_G is the minimum cut tree.

The idea of MCL is when we want to find the community around a source s, but do not care about the sink, we can introduce an artificial sink t. The MCL algorithm (min-cut clustering) expands the graph G into G' by connecting each vertice in V to t with an edge weight (i.e., capacity) of α (Algorithm 3.1 line 3). After expanding the graph, the MCL algorithm builds the minimum cut tree $T_{G'}$ for G', and then removes t from $T_{G'}$. After removing the artificial sink t from $T_{G'}$, all the connected components become clusters of G.

Algorithm 3.1 Min-cut Clustering Algorithm

Require: $G = (V, E), \alpha,\ artifical\ sink\ t$
1: $V' = V \cup t$,
2: **for** $v \forall V$ **do**
3: $E \leftarrow w(t, v) = \alpha$
4: $E' = E$
5: **end for**
6: $G' = (V', E')$
7: *Calculate the minimum cut tree* $T_{G'}$ *of* G'
8: *Remove* t *from* $T_{G'}$
9: *Return all connected components as the clusters of* G

The value of α is determined with eq. 3.2. The value serves as an upper bound of the inter-community edge capacity and a lower bound of the intra-community edge capacity.

$$\frac{c(S, V\backslash S)}{|V\backslash S|} \leq \alpha \leq \frac{c(P, Q)}{min\{|P| - |Q|\}} \tag{3.2}$$

Where $P \cap Q = \emptyset$ and $P \cup Q = S$

When the value of α is close to 0, the minimum cut between t and any other vertex of G become a trivial cut c($\{t\}$, V), which isolates t from the rest of the vertices. As long as G is a connected graph, only one cluster will be produced after clustering.

On the other extreme, as α is close to infinity, the clustering algorithm produces n trivial clusters, all singletons. For values of α between these two extremes the number of clusters will be between 1 and n. The exact value depends on the graph structure of G and the distribution of the weights over the edges. When implementing this algorithm, we often need to recursively try smaller values of α in order to determine the best value.

3.4.2 Locality Clustering Algorithm

The Locality Clustering Algorithm (LC Algorithm) was proposed in [60] to overcome the weakness of MCL algorithm and identify CTCs (communication-based transient communities) in dynamic networks.

A time-evolving OSN can be represented as a series of snap shots of graph (at different time interval). Running MCL algorithm for each snap shot is time and memory consuming, especially when the graph for each snap shot is big. The LC algorithm does not run the MCL algorithm on a whole graph. Instead, the LC algorithm deals with the graph clustering at edge level, and it runs MCL algorithm only under certain conditions.

Whenever a new edge is joining the graph (i.e., network), the LC algorithm examines the quality changes of the cluster(s), to which the inserted edge connects. When the edge insertion meets the re-clustering criteria, the algorithm re-clusters the edge's vertices and the vertices within the cluster(s), to which the inserted edge connects. Otherwise, no clustering needs to perform. Since each time only some parts of the graph are processed, the LC

Algorithm lowers the computation cost.

In the LC algorithm, each vertex must belong to one cluster. Therefore, a new cluster is created if any of the two vertices that formed the inserted edge do not belong to any cluster ($CreateNewCluster$). After assigning vertices into the right clusters, the LC algorithm uses the four criteria listed below (refer to Algorithm 3.2) to determine whether re-clustering is required:

Criterion 1

If two vertices of the inserted edge belong to same cluster C_u, the insertion may change the clustering quality of C_u. In this case, the LC algorithm runs sub-graph clustering on cluster C_u (Algorithm 3.3).

Criterion 2

If the two vertices of the inserted edge belong to two existing clusters C_u, C_v, and if the quality of clustering for two clusters (Cq_u and Cq_v)($CheckCutQuality$) are less than or equal to α in spite of this edge insertion, then re-clustering is not required because the qualities of the clustering are maintained.

$$CheckCutQuality(S) = \frac{c(S, V \backslash S)}{|V \backslash S|} \qquad S = C_u \text{ or } C_v \tag{3.3}$$

Where V represents the total vertices in the current snap shot.

Criterion 3

On the contrary, if the two vertices of the inserted edge belong to two existing clusters C_u, C_v, and if the value of merging ($CheckMerge$) for the two clusters, C_u and C_v is larger than or equal to α, the algorithm merges two clusters.

$$CheckMerge(C_u, C_v) = \frac{2c(C_u, C_v)}{|V|} \tag{3.4}$$

Where V represents the total vertices in the current snap shot.

Criterion 4

Lastly, if either Cq_u or Cq_v is larger than α, and the value of merging (eq. 3.4) is smaller than α, then the quality of clustering has reduced. The algorithm selects both clusters and runs sub-graph clustering on them (Algorithm 3.3)

To apply the LC algorithm to the collected dataset, when running the experiment, the collected OSN data was first divided into n snap shots. The value of n was 200, because this research set the damping time interval (each snap shot) to one-day (see Experimental Results for details). We ran the MCL algorithm on the first snap shot (time t_0 - t_1) to get a set of initial clusters. After obtaining the initial clustering results, the LC algorithm was executed whenever an new edge was inserted. The process was repeated from time t_1 - t_2 to t_{n-2} - t_{n-1}.

It is worth mentioning that the LC algorithm only run MCL algorithm when inserting edges meet Criteria 1 and 4. The LC algorithm contracts the graph (*ContractGraph*) by considering all the vertices outside the target cluster(s) (i.e., C_u and C_v) as one vertex when re-clustering the graph. Since the majority (i.e., vertices) in the graph is considered as one vertex, the LC algorithm only has to cluster a small number of vertices each time, largely reducing the cost for computations.

3.4.3 Geo-location Proximity

To discover TLCs, this thesis has proposed the geospatial locality clustering algorithm (GLC algorithm) (Algorithm 3.4 line 33, and Algorithm 3.5). And the GLC algorithm is one of the original contributions of this thesis's work. The GLC algorithm is the combination of LC algorithm and geo-location proximity algorithm. The idea of geo-location proximity is during the consecutive process of inserting edges (the edge insertions in the LC algorithm), the algorithm picks up the "largest local communities"(targets) discovered through the LC algorithm and consecutively join geographically close users into the targets. Through the

Algorithm 3.2 Locality Clustering Algorithm

Require: $G = (V, E), \alpha$

for $t = begin \rightarrow end$ **do**
  for all $e \in E(t) \cap E(t-1)$ **do**
    $e' = ExponentialDecay(e)$
    if $e' \leq 0$ **then**
      $break$
    end if
    if u $\notin V(t)$ **v** $\notin V(t)$ **then**
      $C''(t) = CreateNewCluster(C'(t))$
    else
      $C''(t) = C'(t)$
    end if
    $Determine\ C_u(t)\ and\ C_v(t)$
    if $Cu(t) = Cv(t))$ **then**
      $G''(t) = ContractGraph(G'(t), Cu(t))$
      $C(t+1) = SubgraphClustering(G''(t), \alpha)$
    else
      $Cq_u = CheckCutQuality(G'(t), C_u(t))$
      $Cq_v = CheckCutQuality(G'(t), C_v(t))$
      if $Cq_u \leq \alpha\ and\ Cq_v \leq \alpha$ **then**
        $C(t+1) = C''(t)$
      end if
      if $CheckMerge(C_u(t), C_v(t)) \geq \alpha$ **then**
        $C'''(t) = MergeClusters(G'(t), C_u(t), C_v(t))$
        $G''(t) = UpdateGraph(G'(t), C'''(t))$
        $C(t+1) = C'''(t)$
      end if
      if $(Cq_u > \alpha \cup Cq_v > \alpha) \cap CheckMerge(C_u(t), C_v(t)) < \alpha$ **then**
        $C'''(t) = C_u(t) \cup C_v(t)$
        $G''(t) = ContractGraph(G'(t), C'''(t))$
        $C(t+1) = SubgraphClustering(G''(t), \alpha)$
      end if
    end if
  end for
end for

Algorithm 3.3 ContractGraph and SubgraphClustering

Require: $G(t) = (V(t), E(t))$, *cluster* S, α
  Contract $G(t)$: *Reduce* $G(t)$ *by replacing vertices in* $V(t) \backslash S$ *with a new vertex* x
  $E'(t) = \emptyset$, $V'(t) = \{x\}$
  for all $v \in V(t) \backslash S$ **do**
    for all $s \in S$ **do**
      $sum = 0$
      if $(v, s) \in E(t)$ **then**
        $sum = sum + w(v, s)$
      end if
    end for
    $E'(t) \leftarrow (v, s) = sum$
    $V'(t) \leftarrow s$
  end for
  $G'(t) = (V'(t), E'(t))$
  Run Min Cut Clustering Algorithm (*Algorithm* 3.1) *on* $G'(t)$

iterative process, the users within the clusters gradually become both geographically close and highly communicating with others.

In the experiment, at each edge insertion, the GLC algorithm obtained the CTCs discovered by the LC algorithm and picked up the largest CTCs as targets. For each target, the GLC algorithm determined whether the majority in the target locates in close proximity to a certain area (representative geographic area), and if yes, the algorithm assigned the center of this representative geographic area as the centroid (c) of the target (i.e., the largest local communities).

After getting the centroids of targets, for each target, the GLC algorithm identified the target's connected communities (i.e., communities that have more than one connection). If any, for each connected community (K), the GLC algorithm examined whether users within K were geographically close to the centroid of the target, and the geographically close users were then joined into the target. The GLC algorithm assumes that the users are geographically close to the target if their distances to the centroid are less than the threshold denoted with *radius* (eq. 3.5, Algorithm 3.5, from line 7 to 9,). We can express this condition as follows:

$$d = |g(id) - c| < radius \qquad \forall v = (id, g(id)) \in K \tag{3.5}$$

The value of *radius* should be determined based on the region of interest. Due to the limited resolution of the location data, this research targets the city-level communities (i.e., Calgary), and the radius value is 70 km.

When implementing the GLC algorithm, one observation has been made. Mostly, the targets did not have connections (edges) to the geographically close users (vertices) in their connected communities. This is because two Calgarians can be interested in the same topic and connect to the same user, but they do not have connections to each other. Nevertheless, the relationships between the non-connected and geographically close vertices are still important. These geographically close vertices are intermediates in the network and may help group geographically close vertexes in the next iteration.

Even though the GLC algorithm joins the geographically close vertices into their targets, without creating real connections between these geographically close vertices, these non-connected vertices are very likely to be removed from clusters when new edges are inserted and the algorithm re-clusters these vertices.

To overcome this problem, the GLC algorithm creates dummy edges to connect the geographically close vertices to the targets (Algorithm 3.5, from line 10 to 12). In order to make the dummy edges stable enough within current time interval, the weights are set to α. If the values of the weights are smaller than α, it is likely that the GLC algorithm will cut the sub-graphs through these connections when new edges are inserted. This chapter only explains the GLC algorithm, and the experimental results of the GLC algorithm are detailed in Chapter 5.

Algorithm 3.4 Geospatial Locality Clustering Algorithm

Require: $G = (V, E), \alpha$

for $t = begin \rightarrow end$ **do**
  for all $e \in E(t) \cap E(t-1)$ **do**
    $e' = ExponentialDecay(e)$
    if $e' \leq 0$ **then**
      $break$
    end if
    if u $\notin V(t) \cup$ **v** $\notin V(t)$ **then**
      $C''(t) = CreateNewCluster(C'(t))$
    else
      $C''(t) = C'(t)$
    end if
    $Determine\ C_u(t)\ and\ C_v(t)$
    if $Cu(t) = Cv(t))$ **then**
      $G''(t) = ContractGraph(G'(t), Cu(t))$
      $C(t+1) = SubgraphClustering(G''(t), \alpha)$
    else
      $Cq_u = CheckCutQuality(G'(t), C_u(t))$
      $Cq_v = CheckCutQuality(G'(t), C_v(t))$
      if $Cq_u \leq \alpha\ and\ Cq_v \leq \alpha$ **then**
        $C(t+1) = C''(t)$
      end if
      if $CheckMerge(C_u(t), C_v(t)) \geq \alpha$ **then**
        $C'''(t) = MergeClusters(G'(t), C_u(t), C_v(t))$
        $G''(t) = UpdateGraph(G'(t), C'''(t))$
        $C(t+1) = C'''(t)$
      end if
      if $(Cq_u > \alpha \cup Cq_v > \alpha) \cap CheckMerge(C_u(t), C_v(t)) < \alpha$ **then**
        $C'''(t) = C_u(t) \cup C_v(t)$
        $G''(t) = ContractGraph(G'(t), C'''(t))$
        $C(t+1) = SubgraphClustering(G''(t), \alpha)$
      end if
    end if
    $C'(t+1) = GeolocationProximity(C(t+1))$
  end for
end for

Algorithm 3.5 Geolocation Proximity

Require: $G(t) = (V(t), E(t)),\ C(t),\ radius,\ \alpha$

$C_{top}(t) = BiggestClusters(C(t))$
for all $c \in C(t)$ **do**
  $C_{centroid} = GetRepresentativeAreaCenter(c)$
  $Sub\_C(t) = ConnectedClusters(C(t), c)$
  for all $sub\_c \in Sub\_C(t)$ **do**
    for all $vertex \in sub\_c$ **do**
      if *vertex has geolocation* **then**
        $d = DistanceToCenter(vertex, C_{centroid})$
        if $d < radius$ **then**
          if $edge(vertex, c) \notin E(t)$ **then**
            $CreateEdeges(vertex, c)$
            $AssignEdgeWeight(vertex, c, \alpha)$
          end if
          $C(t) = AppendNodeToCluster(vertex, c)$
          $C(t) = RemoveNodeFromCluster(vertex, sub\_c)$
        end if
      end if
    end for
  end for
end for

Chapter 4

The Discovery of Local Communities of Interest

This part of thesis focuses on the discovery of communities of interest in local social networks. This research has proposed a framework to identify communities of interest in local social networks. This research has examined the relevant network properties for building sociograms, as well as, applied natural language processing to clean short texts. In addition, this research has proposed a clustering algorithm that improves the graph-based fast-greedy optimization of modularity (FGM) by using text similarity measure. This research uses Twitter data as resource, and the proposed framework is designed based on the Twitter data collected.

4.1 Data Collection

Since there was no LSN data available online and no APIs allowing us to collect LSN data, this research has proposed a model to sample Twitter accounts near-by Calgary area (city level). Conventional approach used for sampling social network accounts is breadth-first search (BFS) algorithm. The BFS algorithm starts from a "seed vertex" and searches its neighbors. Then for each neighbor, the algorithm visits its unexplored neighbors. The process is repeated until certain number of users is reached. In order to collect an LSN dataset, the model proposed modifies the BFS sampling algorithm to sample Twitter accounts (Figure 4.1).

Twitter has attracted different types of users ranging from readers, lifestreamers, and connectors to marketers and celebrities. The users' behaviors vary depending on their purposes to join Twitter. For example, the lifestreamers tend to post tweets to record their daily lives, and the readers tend to follow breaking news. In addition, Twitter is a highly

dynamic social network; users can be active in some time periods, but less active in other time periods. Since the purpose of this research is to identify CoIs, this research excludes isolated users (i.e., lifestreamers) and less active users.

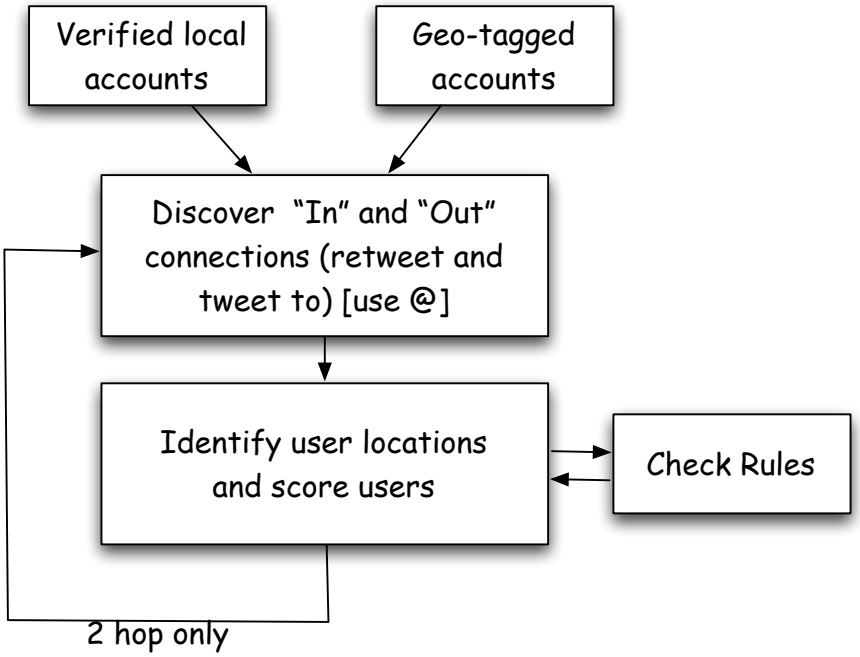


Figure 4.1: Modified Breadth-first Search

To make sure the accounts collected by the modified model would be currently active accounts and geographically close to Calgary, the modified BFS is designed as a scoring model that evaluates how likely the user is frequently communicating with Calgarians and is living near-by Calgary area. Several rules were defined to score users, and the rules are described as follows.

Rule 1: Score the seed vertices with 10, because they are well identified

Rule 2: Score the neighbors of discovered users with 0 at the beginning

Rule 3: Increase the score if a user's place or hometown is Calgary

Rule 4: Determine a user's score by calculating the average from top 25 % friends' scores,

indicating the penalty for an incorrect prediction

Rule 5: Increase the score if a user highly communicate to local users (e.g., over 10 times in and out degree communications)

Rule 6: Reduce the score if a user has high out degree and low in degree communications, because it may be a span

Rule 7: Increase the score if a user includes more than five #yyc, #calgary hash tags in tweets, indicating it highly concerns about events near-by Calgary

Rule 8: Stop scoring a user if his or her score reaches 10

The data collection started from a set of well-identified local accounts (i.e., seed vertices), such as geo-tagging users, universities and local broadcast media. When running the modified BFS, the model used the indicated eight rules to collect the accounts whose locations are near-by Calgary and frequently communicating with the seed vertices (i.e., one-hop accounts), as well as, the accounts whose locations are near-by Calgary and frequently communicating with the one-hop accounts (i.e., two-hop accounts).

Since this research aims to use the short texts in tweets to identify users' interests, the model has collected the most recent 3,600 tweets for each identified account (whenever possible). At the end of the data collection, the modified model has collected 7,878 users and 8,057,836 tweets (Table 4.1).

To evaluate the performance of this scoring model in terms of the ability to discover Calgarians, a small experiment was set up to compare the identified accounts with the 1000 Calgarian Twitter accounts listed on twitaholic.com[1]. The experimental results indicated that 87 % of the local accounts listed on twitaholic.com were discovered by the proposed model.

[1]twitaholic.com is a website designed to track twitter users, and also lists the most popular tweeters in a certain area

Table 4.1: Twitter Dataset

Property	Total
Total Users	7,878
Sharing Link Users	1,928
Total Tweets	8,057,836
Messages(@$\langle u \rangle$) of Study	447,867
User Pairs of Study	69,378

4.2 Building Sociograms

Twitter is an OSN (social media) platform that allows users to interact with friends and strangers and share free form contents. Mostly the shared contents and the user-created connections are complicated and are not always helpful for identifying CoIs. Twitter data is therefore difficult to analyze. For example, users who are friends may not have same interests. Nevertheless, the uses of SNA and TM techniques can help discover patterns from the OSN data.

As mentioned in Chapter 2, identifying relevant network properties to build graphs is important for SNA. This research has examined the collected data and defined the useful properties to build graphs. In the collected data, there are sharing-link users who carry special network properties. To examine whether these properties help discover CoIs, this research has validated the usefulness of these sharing-link users and their properties.

4.2.1 Sharing-link Users

Results of previous studies have demonstrated that information flowed through the URLs in Twitter, and the reciprocated relationships in Twitter exhibited certain level of homophily[2] [66]. In addition, Wu *et al.* [101] examined the number of URLs initiated by different types of users and found that the general public contributed large amount of URLs even though they were not as active as the bloggers. Based on the observations made by the previous

[2]Homophily is a tendency that a contact between similar people occurs at a higher rate than among dissimilar people

studies, this research has examined the URLs in the collected tweets.

The following are observations made from the collected data. First, there are accounts sharing links with the public, and the share links are mostly from particular websites. For example, CalgaryHerald, one of the biggest broadcast media in Calgary has shared links with the URL prefix "http://www.calgaryherald.com/"; BikeBike Inc, a company features bicycles has shared links with the URL prefixes "http://www.bikebike.ca/" and "http://www.bikecalgary.com/". Second, the accounts that have similar interests tend to share links from the same websites. For example, BikeBike Inc and Cyclepalooza[3] have shared links from Bike Calgary website.

To examine whether the sharing-link accounts share links for purposes, this research has examined whether these sharing-link users have more followers. The choice of the number of followers is based on the studies demonstrating that information in Twitter flowed through "following" [54]. In addition, "following" in Twitter can be a representation of interest, and the number of followers can be a representation of prestige. Therefore, if the sharing-link users have more followers than that of the general users, the former users are very likely to be the "leaders" of the CoIs. This research believes that once the "leaders" are found, it will be easier to identify the members in CoIs (e.g., audiences and consumers) by simply examining the graph structure. More specifically, this research considers that if a first user follows a second user, it is an indication that the former is interested in some of the topics of interest of the latter.

Figure 4.2 indicates the numbers of follower for the sharing-link users and the non-sharing-link users. To increase the readability, the results are expressed in logarithmic form. As shown, the majority of sharing-link users have 1000 - 4000 followers ($2^{10} - 2^{12}$), while most of the non-sharing-link users have around 100 - 300 followers ($2^7 - 2^8$). Results demonstrate that the sharing-link users have relatively high degree centralities in the Twitter network.

[3]Cyclepalooza is a volunteer-directed, community-based, bicycle-themed festival promoting arts, culture, events, and fun in Calgary

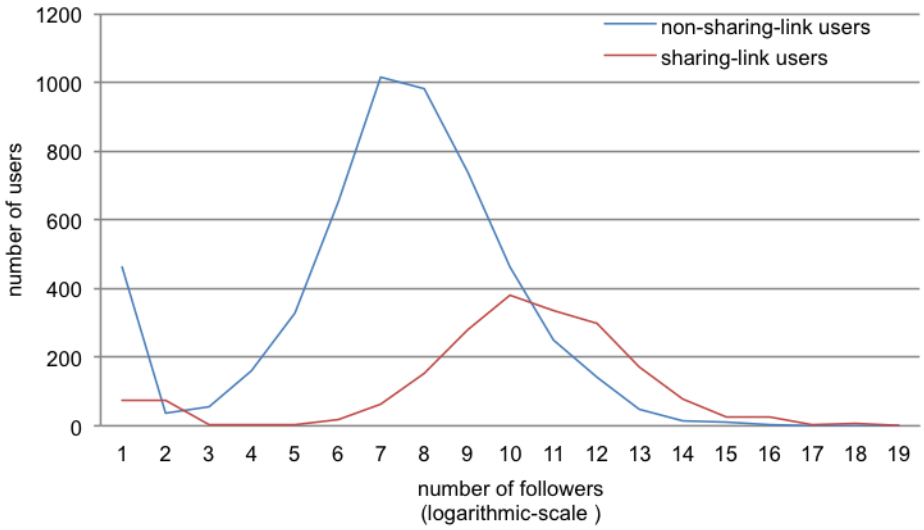


Figure 4.2: Distribution of Follower Numbers in Logarithmic Scale

Blue line represents the distribution for non-sharing-link users, and Red line represents distribution for sharing-link users

4.2.2 Selecting Attributes

In order to select appropriate properties to build sociograms, this research has examined three properties in the collected data, and they are 1) the short texts, 2) the mention symbols "@", and 3) the URL prefixes.

4.2.2.1 Short Texts

Short texts are so far the major text resource for most mobile applications (e.g., Foursquare). Users now use limited words to describe feelings and events. The short texts therefore can contain a lot of information about the users. In Twitter, users share tweets[4]. Not like documents, such short messages are sparse in nature because the numbers of words that users can share are limited. In addition, users can share short URLs in their tweets, and the short URLs re-direct to other webpages. Tweets and short URLs contribute two types of short text resource that can be utilized for identifying CoIs: tweets and web page titles

[4]A tweet is a short message less than 140 characters including short URLs

derived from the short URLs. In order to evaluate the validity of using short texts to build sociograms, this research has processed the short texts with NLP techniques (details in Chapter 4.4) and extracted nouns to get bags of words for users. This research has also examined the short texts in the collected tweets carefully and had the observations as follows.

One of the main observations made about these short texts is that there are semantically related words in the tweets, such as "iPhone", "iPad", and " Smokehouse", "BBQ". Another example of the semantically related words is polysemy, e.g., "apple" (one can regard "apple" as a general term for Apple products or as a fruit). Besides, the tweets and the web page titles contain a lot of local business names and multi-faceted words, such as "Rhino Amsterdam" and "Shane Homes" that require local knowledge to identify their relationships with other words. Moreover, these "local" words have higher frequencies of use because of commercial issues, and this has negative impact on the text similarity measure.

This research has examined whether it is possible to use word dictionaries such as WordNet to establish the word similarities between the semantically related words in the tweets. However, experiments showed negative results. For example, WordNet failed to identify the relation between "iPhone" and "iPad" and the relation between "BBQ" and "smoke". In addition, WordNet showed a high similarity score (0.8) for "road" and "lane", while it showed a low score (0.4) for "road" and "cyclist", which means that word dictionaries are not always reliable. It is not hard to imagine that word dictionaries work better for predefined human languages than for natural human languages.

Another observation is that the web page titles in the "Calgarian network" describe users' interests well (details in Chapter 5). However, the titles are too short to construct reliable relationships between users. The observations mentioned denote that using text clustering to cluster short texts requires sophisticated processes, which increases the difficulty to discover CoIs by relying on pure short text analysis. Nevertheless, with appropriate text processing, the short texts can be used to derive the similarities between users.

In this research, instead of using text similarity to build graphs, this research has examined the network structure of the collected data and built two sociograms using two relationships between users: the communication times and the common URL prefixes between users. And this thesis aims to integrate short text with graph structure to discover CoIs.

4.2.2.2 Mention Graph

The first type of graph is built using so-called "mentions". In Twitter messages, users use @$\langle username \rangle$ (i.e., mentions) within tweets to reference other users. This research regards a message sent by user u_1 including @$\langle u2 \rangle$ as representing the communication between u_1 and u_2. An undirected sociogram can be built by establishing edges between all pairs of u_1 and u_2 discovered and assigning the communication times between users as the edge weights.

4.2.2.3 URL Prefix Graph

The second type of graph is built using the common URL prefixes between users. Motivated by WCA, this research has built another undirected sociogram by establishing edges between users who have more than three common URL prefixes. The choice of URL prefixes is based on the previous research demonstrating that information in Twitter flowed through the short URLs in tweets. This research believes that the flow of information through the short URLs help indirectly to capture the interests of users.

To build the URL prefix graph, the shared short URLs in tweets were processed to obtain their original URLs where the URL prefixes were then extracted. The URL prefixes were sorted by their frequency of use. However, some of the users (in the collected data) have shared some URLs from some websites once, and the one time URL prefixes tended to be more irrelevant to users' interests. Therefore, this research only considers the URL prefixes with frequency of use larger than one.

The URL prefix graph is built by adding edges between users who have more than three URL prefixes in common. The weights of the edges are determined with eq. 4.1. The

proposed weighting model (eq. 4.1) first normalizes the importance of each URL prefix (scores) for each user. Then the model determines the edge weights by summing up the multiplications of the normalized scores of the common URL prefixes between the users (eq. 4.2).

$$\forall freq_i > 1 \qquad URLPrefixScore_i = \frac{freq_i}{\sum_{i=1}^{n} freq_i} * 100 \tag{4.1}$$

$$Weight_{u_1 u_2} = \sum_{i=1}^{n} URLPrefixScore_{iu_1} * URLPrefixScore_{iu_2} \tag{4.2}$$

In eq. 4.1, n represents the total number of URL prefixes with frequency larger than 1, and in eq. 4.2, $URLPrefixScore_{iu_1}$ and $URLPrefixScore_{iu_2}$ represent the URL prefix score of common URL prefix i for user u_1 and u_2 respectively.

The reason of normalizing the URL prefixes is because users shared different amounts of URLs. Normalization is required in order to compare the importance of each URL prefix to every person. And the reason of assigning edge weights by using eq. 4.2 rather than simply counting the number of co-occurrences is because the proposed scoring model may make the community structure clearer. Users who have more common URL prefixes with higher normalized scores are more similar (i.e., higher scores), and users who have more common URL prefixes with lower normalized scores are less similar (i.e., lower scores).

It is worth mentioning here, in this research, it has been observed that the URL prefixes can be further divided into two groups: multiple-interest and single-interest. A multiple-interest URL prefix refers to a website that contains several categories, such as broadcast media websites. This type of website usually has several categories starting with the same prefixes (e.g., http://www.calgaryherald.com/business/ and http://www.calgaryherald.com/sports/). On the contrary, a single-interest URL prefix refers to websites that only have one interest. To examine the helpfulness of the multiple-interest and single-interest prefixes to the discovery of CoIs, the multiple-interest URL prefixes are further divided into the category level.

However, using only single-interest URL prefixes to build the URL prefix graph pro-

vides better clustering results than using both prefixes. This is because that the proposed model (eq. 4.1 and 4.2) normalizes the importance of each prefix with the sum of frequencies of use (eq. 4.1). Any insertion of high frequency prefix affects the importance of others. Since the multiple-interest prefixes are usually top sharing items, they decrease the weights (i.e., similarities) for most connections in the graph. Moreover, URL prefixes such as "http://www.calgaryherald.com/news" interest readers from multiple domains and create irrelevant connections.

Even though the multiple-interest prefixes have high frequency of use, analyzing their contribution to community structure is more complex than that of single-interest. Based on the above-mentioned observations, this research uses single-interest prefixes to build the URL prefix graph.

4.3 Enhanced Fast-greedy Optimization of Modularity

4.3.1 Fast-greedy Optimization of Modularity

Modularity measures the quality of a division of a network into communities. If the divisions are created based on a good model, then the density of edges within communities is high while the density of connections between communities is low. The idea of modularity is that a random graph is not expected to have a community structure. By maximizing the difference between the actual graph structure and the random graph structure (expressed with the quality function Q), one can obtain the optimal division. The quality function can be defined as [45]:

$$Q = \frac{1}{2m} uv(A_{uv} - P_{uv})\delta(C_u - C_v) \tag{4.3}$$

A_{uv} represents the adjacency matrix that denote the exact relationships between vertices in a graph, m is the total number of edges, and P_{uv} is the expected connections between two vertices u and v when the graph is a random graph. $\delta(C_u - C_v)$ yields 1 if vertex u and vertex v are in the same cluster, 0 otherwise.

The most straightforward approach to implement this algorithm is storing the adjacency matrix of the graph and progressively merging pairs of rows and columns as corresponding communities are merged (i.e., greedy optimization modularity (GM)). However, it has been proved that GM causes a lot of redundant processes and wastes memory [34].

FGM proposed by Clauset *et al.* enhances the greedy optimization modularity. The authors have observed that only vertex pairs that are in the same clusters contribute to the sum, i.e., Q, and the above equation can be rewritten as follows [45]:

$$Q = \sum_{c=1}^{n} [\frac{l_c}{m} - (\frac{d_c}{2m})^2] \tag{4.4}$$

Where n is the number of clusters, l_c is the total number of edges joining vertices of cluster c, and d_c is the sum of the expected random graph degree of the vertices of c.

By progressively merging communities that can achieve the largest quality change (ΔQ), one can obtain the largest value of Q, which means a better clustering result. Since joining two communities with no connection between them will not change the value of Q, the authors enhanced the GM model by introducing a sparse matrix storage and a special data structure (i.e., max-heap) to keep track of the largest quality change of cluster pairs.

As shown in Algorithm 4.1, FGM first assigns each vertex into one community (i.e., cluster) (line 1-4). For each pair of communities (vertices) that has at least one edge between them, the FGM calculates the value of ΔQ (according to the graph connectivity) and stores the value into a sparse matrix Q (line 9).

Each row of the sparse matrix stores the values of ΔQ from one community u to all other communities v, to which u connects. The max-heap H is then employed to store the largest ΔQ value of each row of Q and the corresponding community ids u, v that contribute the largest ΔQ value (line 13-15).

After the initialization of Q and H, in the consecutive process, FGM iteratively pops out the object in H that has the highest ΔQ value and merges the corresponding communities

until only one community is left (line 17-29). It also progressively merges pairs of rows and columns in Q as the corresponding communities are merged.

Algorithm 4.1 Fast-greedy Modularity Optimization

Require: *social graph* $G = (V, E)$, *sparse matrix* Q, *max heap* H, *cluster set* C

1: **for all** $v \in V$ **do**
2: $cid = AssignCluster(v)$
3: $C \leftarrow cid$
4: **end for**
5: **for all** $u \in C$ **do**
6: $count = 0$
7: **for all** $v \neq u \in C$ **do**
8: **if** u *has connection to* v **then**
9: $Q(u, count) = CalculateDeltaQ(u, v, G)$
10: $count = count + 1$
11: **end if**
12: **end for**
13: $maxQ,\ max\_v = max(Q(u, :))$
14: $heapObj = (u, max\_v, maxQ)$
15: $H(u) \leftarrow heapObj$
16: **end for**
17: **while** $size(H) > 0$ **do**
18: $(pop\_u, pop\_max\_v, pop\_maxQ) = heappop(H)$
19: $support\_u = GetNumberOfConnection(pop\_u)$
20: $support\_v = GetNumberOfConnection(pop\_v)$
21: **if** $support\_u < support\_v$ **then**
22: $merged\_node = u$
23: $merging\_node = v$
24: **else**
25: $merged\_node = v$
26: $merging\_node = u$
27: **end if**
28: $MergeCommunities(Q, H, merged\_node, merging\_node)$
29: **end while**

FGM successively merges two communities (i.e., merged and merging community) to form a bigger community, and it considers three scenarios when merging a pair of communities (Algorithm 4.2 *Merge Communities*): 1) community k is connected to both merged and merging community, 2) k is connected to merged only, and 3) k is connected to merging only. Meanwhile, the ΔQ values in Q related to k, merged and merging community must be

updated. When Q is updated, the largest ΔQ value for each row may be changed. Every time a pair of communities are merged, FGM sorts each row of Q to obtain the max value of the row and updates H accordingly (Algorithm 4.3 *Update MaxHeap*).

In Algorithm 4.3, H is updated based on the update of Q, which means that the order of merging relies on the updated values of ΔQ. In addition, the ΔQ values are calculated based on the connectivity of communities relative to the rest of the graph. Consequently, the clustering results are very sensitive to graph structures.

In a real word social network, graph structure is usually noisy. FGM may fail to cluster the graph accurately. To address this problem, this research has proposed an enhanced FGM algorithm that introduces text similarity measure to the *Update MaxHeap* (under Algorithm 4.2) to make the algorithm less sensitive to graph structures (line 28).

Algorithm 4.2 Merge Communities

Require: $Q,\ H,\ merged\_node, merging\_node$
1: $merged\_node\_connections = FindConnectedNodes(merged\_node)$
2: $merging\_node\_connections = FindConnectedNodes(merging\_node)$
3: **for all** $k \in merged\_node\_connections \cap merging\_node\_connections$ **do**
4: $\quad UpdateQ(k, merged\_node, merging\_node)$
5: $\quad UpdateMaxHeap(Q, H, k)$
6: $\quad UpdateMaxHeap(Q, H, merging\_node)$
7: **end for**
8: **for all** $k \in merged\_node\_connections$ *and* $k \notin merging\_node\_connections$ **do**
9: $\quad UpdateQ'(k, merged\_node, merging\_node)$
10: $\quad UpdateMaxHeap(Q, H, k)$
11: $\quad UpdateMaxHeap(Q, H, merging\_node)$
12: **end for**
13: **for all** $k \in merged\_node\_connections$ *and* $k \notin merging\_node\_connections$ **do**
14: $\quad UpdateQ''(k, merged\_node, merging\_node)$
15: $\quad UpdateMaxHeap(Q, H, k)$
16: $\quad UpdateMaxHeap(Q, H, merging\_node)$
17: **end for**

Algorithm 4.3 Update MaxHeap

Require: $Q, H, target\_u$

1: $rowQ = RetrieveRow(Q, target\_u)$
2: $maxQ, max\_v = max(rowQ)$
3: $heapObj = (target\_u, max\_v, maxQ)$
4: $H(target\_u) \leftarrow heapObj$

4.3.2 Enhanced Model with Text Similarity Measure

The enhanced algorithm compares the text similarity between communities when Q and H are updated (i.e., to execute Algorithm 4.5 *Enhanced Update MaxHeap*). The idea is any pair of communities that has higher text similarity and has more than one connection between them should be part of the same CoI, and the merge priority should be increased. By setting the right condition for text similarity measure, the proposed algorithm compensates the issue of sensitivity of original clustering algorithm.

To integrate the text similarity measure into FGM, the algorithm has to determine when to merge two communities based on the text similarity measure and when to merge them based on the graph structure. To achieve this, a threshold T is created to determine whether the text similarity between a pair of communities is high enough to increase the priority to merge. It is worth mentioning that T determines the strength of text similarity measure and the chosen value depends on the quality of the text resource and the graph structure. For example, if the community structure of the graph is weak, the value of T should be low, and if the average scores of text similarity are high, the value should be high and vice versa.

This algorithm introduces an extra matrix S, which is similar to Q to record the text similarities between users. The matrix S is initialized by calculating the text similarity between vertices (communities) (Algorithm 4.4, line 10). To initialize each object of H, the proposed algorithm considers two scenarios: for each row, 1) if the largest value of the row of S is smaller than T, the algorithm sorts the row of Q, and selects the object with largest ΔQ to update H; and 2) if the largest value of the row of S is larger then T, the algorithm uses the ΔQ object with the highest text similarity to update H (Algorithm 4.4, line 14 -

22).

During the iterative process of merging communities, the algorithm updates the text resources for the new communities, so that the text similarities between the new communities can be determined in the next iteration (line 36). Afterwards, the algorithm updates the ΔQ values for community k, merged community, and merging community, as well as, updates S and the objects in H by running Algorithm 4.2 *Merge Communities* and Algorithm 4.5 *Enhanced Update MaxHeap*.

4.4 Text Similarity

4.4.1 Natural Language Processing

As mentioned previously, it has been observed that tweets and web page titles are very noisy and sparse, and it is not reliable to build sociograms based on them. In addition, word dictionaries fail to establish the similarities between words in tweets. This research has applied NLP (natural language processing) to tackle this issue. This research has chosen *NLTK* to process the short texts in the collected tweets because the entire algorithm is implemented by Python.

As mentioned in Chapter 2, NLP has been applied to understand human language structures for decades. There are several techniques to process human languages. Based on the characteristics of the collected tweets, this research has applied regular expression to valid words, as well as, applied spelling check to fix typographic errors. Since the previous study stated that nouns are more informative (see later section), this research extracted nouns from the short texts (Algorithm 4.6). Details for each part are described as follows.

4.4.1.1 Regular Expression

The most fundamental tool for NLP is regular expression, which is a formal language for specifying text strings. In Twitter, users include short URLs and hash tags[5] to inflate the

[5]A hash tag is a string starting with # symbol, and is used to mark keywords or topics in a tweet

Algorithm 4.4 Enhanced Fast-greedy Modularity Optimization

Require: $social\ graph\ G = (V, E), sparse\ matrix\ Q, max\ heap\ H, cluster\ set\ C$ $similarity\ matrix\ S,\ threshold\ for\ similarity\ \varepsilon$

1: **for all** $v \in V$ **do**
2: $cid = AssignCluster(v)$
3: $C \leftarrow cid$
4: **end for**
5: **for all** $u \in C$ **do**
6: $count = 0$
7: **for all** $v \neq u \in C$ **do**
8: **if** $u\ has\ connection\ to\ v$ **then**
9: $Q(u, count) \leftarrow CalculateDeltaQ(u, v, G)$
10: $S(u, count) \leftarrow CalculateTextSimilarity(text\_u, text\_v)$
11: $count = count + 1$
12: **end if**
13: **end for**
14: $maxS,\ max\_v = max(S(u, :))$
15: **if** $maxS < T$ **then**
16: $heapObj = (u, max\_v, maxQ)$
17: **else**
18: $max\_S,\ max\_v = max(S(u, :))$
19: $most\_similar\_Q = Q(u,\ max\_v)$
20: $heapObj = (u, max\_v, most\_similar\_Q)$
21: **end if**
22: $H(u) \leftarrow heapObj$
23: **end for**
24: **while** $size(H) > 0$ **do**
25: $(pop\_u, pop\_max\_v, pop\_maxQ) = heappop(H)$
26: $support\_u = GetNumberOfConnection(pop\_u)$
27: $support\_v = GetNumberOfConnection(pop\_v)$
28: **if** $support\_u < support\_v$ **then**
29: $merged\_node = u$
30: $merging\_node = v$
31: **else**
32: $merged\_node = v$
33: $merging\_node = u$
34: **end if**
35: $MergeCommunities(Q, H, merged\_node, merging\_node)$
36: $UpdateTextResource(merged\_node, merging\_node)$
37: **end while**

Algorithm 4.5 Enhanced Update MaxHeap

Require: $Q, S, H, target\_u$

$rowQ = RetrieveRow(Q, target\_u)$
$rowS = RetrieveRow(S, target\_u)$
$maxS, max\_v = max(rowS)$
if $maxS < T$ **then**
    $maxQ, max\_v = max(rowQ)$
    $heapObj = (target\_u, max\_v, maxQ)$
else
    $most\_similar\_Q = Q(target\_u,\ max\_v)$
    $heapObj = (target\_u, max\_v, most\_similar\_Q)$
end if
$H(target\_u) \leftarrow heapObj$

contents. Since, this thesis does not focus on the hash tags analysis, this research has applied regular expression to exclude short URLs and hash tags from the tweets.

In addition, users drag out words in spoken language to emphasize something. For example, users wrote "goooooood" rather than "good" to emphasize their feelings. Similarly, Twitter users drag out words to emphasize the contents/ feelings. As a result, there are many words like "hmmmmmm", "ooooooh", and "moooommmm" in the tweets. These words need to be corrected before we can further process them. This research has also applied regular expression to normalize the "emphasized" words.

4.4.1.2 Orthographic Normalization and Spelling Check

There are two types of typographic errors in tweets: intentional and unintentional typographic errors. Spelling check tackles the unintentional typographic errors, and orthographic normalization takes care of intentional typographic errors. The orthographic normalization assumes that the errors are from intentional spelling errors [62]. However, the tweets collected contain both types of errors.

There are two common types of orthographical errors in the tweets: frequently used phrases are shortened into acronyms or shortened by using phonetic spellings. Examples of these errors are "ASAP (as soon as possible) ", and "c (see) ya (you) 2moro (tomorrow)". To

disambiguate these words, this research created a look-up table for the common acronyms and the phonetic spelling words.

Twitter users also spell words wrongly. There are models for correcting spellings. Noisy channel model is one of the famous framework [64, 77]. The idea of the noisy channel model is given the original word w, find the correction word c, out of all possible corrections that maximizes the probability of c. By Bayes' theorem, the noisy channel model can be written as follows.

$$argmax_{w \in V} = \frac{P(w|c)P(c)}{P(w)} \tag{4.5}$$

$$argmax_{w \in V} = P(w|c)P(c) \tag{4.6}$$

In eq. 4.5 and 4.6, $P(w|c)$ is the probability that w would be typed when the author meant c. This is also referred to as the channel model. $P(c)$ is referred to as the language model, and it can be any n-gram models[6]. The value of $P(c)$ represents the probability that a correction c stands on its own. This can also be interpreted as how likely is c to appear in an English word. $argmax_{w \in V}$ is equals to choosing the one that gives the largest probability score. Since $P(w)$ is the same for every possible c, we can ignore it.

In order to enumerate the possible corrections c of a given word w, one needs to check the number of edits it would take to turn w into the c, which is equal to finding the edit distance between two words. The most widely adopted edit distance is DamerauLevenshtein edit distance [37, 71]. An edit can be a deletion (remove one letter), an insertion (add a letter), a substitution (change one letter to another) or a transposition (swap adjacent letters). By summing up all the edits, one can determine the edit distance to turn w into the c. Normally, the spelling errors are within edit distance two. This research has used the spelling check function provided by $NLTK$ to correct the spelling errors.

In addition to orthographic normalization and spelling check, there are also syntactic

[6] An n-gram is a contiguous sequence of n items from a given sequence of text or speech

ambiguities in the tweets. For example, a user can tweet "@James thank you! @Alice really enjoys the events!" The first @ is a symbol meaning the author is posting to a certain user (James), and the second @ represents the subject of the sentence, and its removal would be grammatically improper. In order to solve this problem, this research identified all the words starting with the @ mark and replaced the words with "she". In this way, the grammatically structures for the second types of sentences are remained unchanged, and the word "she" will not effect the POS tagging results (details in next sub-section) for the first sentence either.

4.4.1.3 Categorizing Words - None Phase Text Retrieval

We have been taught to remember the corresponding parts of speech of words in school because they compose sentences in daily conversations. There are seven main *parts of speech*(POSs)[7] that we are familiar with: nouns (NN/NNs), verb (V), preposition (PRP), adjective (ADJ), conjunction (CNJ), pronoun (PN), and adverb (ADV). Although they seem to be trivial, these "word classes" are useful categories for many language-processing tasks.

However, in real word scenarios, we usually deal with more than seven parts of speech, for example, nouns can be further divided into proper nouns and common nouns, and verbs can be further divided into main verbs and modal verbs. Moreover, one word can have several POSs, and this causes ambiguities. For example, the "refuse" and "permit" for the following sentence can be both verbs and nouns.

"They refuse to permit us to obtain the refuse permit"

In NLP, solutions for POS-tagging fall into two categorizes: knowledge of neighbors (e.g. n-gram analysis) and knowledge of word probabilities. An example for previous category could be, an adjective never comes before a verb. An example for the later could be, a word "man" is rarely used as a verb. By analyzing the neighbors of different POSs and the word

[7]A part of speech, which is also known as a word class, a lexical class, or a lexical category, is a linguistic category of words

probabilities, one can build linguistic trees that annotate the linguistic structures. Different POS-tagging models have been proposed to increase the accuracy of tagging, e.g., Hidden Markov Models [67]. By processing the above sentence with the POS-tagging models, one will get a list of tagged words:

" [('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')] "

So far, the most commonly used POS tagging algorithm is the Penn Treebank Tags [9]. This thesis has applied the Penn Treebank Tags to tag the words in the tweet sentences. Since the POS-tagging model examines the neighbors of POSs, to disambiguate the tagging, this research decomposed the tweets into sentences by using ".", ":", and "!" as delimiters. Each sentence was then processed with the Pen Treebank tagging algorithm to get a set of POS-tagged words.

After POS-tagging tweets' words, the nouns were extracted. This choice of nouns is based on the researches demonstrating that the nouns and the noun phrases tend to be the most informative elements of any sentence [55]. However, it is most likely the top most frequently used nouns extracted by the Penn Treebank algorithm are the top frequently used words in human conversations. These words increase the overall text similarities and lead inaccurate results. Consequently, this research excluded the top 2,000 most frequently used words provided by Wiktionary [15].

4.4.1.4 Local Terms Removal

As mentioned in the previous research, local users should tend to be concerned and talk more about local things, and local people also tend to use local words or abbreviations for place names. They are also more likely to care about events in places near them. In this research, it has been observed that after removing hash tags, removing short URLs, checking spelling, and extracting relevant nouns, there are still nouns that frequently appear in the

extracted word lists, examples of these words are “cal”, “calgary”, “yyc”, “ alberta”, “ab”, “edmonton”, and “ vancouver”. Mostly these words are place names that are geographically close to Calgary. In this research, such words represent near-by locations were removed.

4.4.2 Text Similarity Measures

After applying the NLP algorithm (Algorithm 4.6) to the tweets and the web page titles. Each user has a list (bag) of words (nouns) that represent his/ her interest. To measure the similarities between two bags of words (i.e., similarities between users ’ interests), this research has applied three similarity measures: Dice, Cosine, and Overlap (eq. 4.7 - 4.9) and examined how does different similarity measures affect the clustering results. This chapter only describes the proposed framework designed for the collected LSN data, and the experimental results can be found in Chapter 5.

$$Dice = \frac{2|Q \cap S|}{(|Q| + |S|)} \tag{4.7}$$

$$Overlap = \frac{|Q \cap S|}{min(|Q|, |S|)} \tag{4.8}$$

$$Cosine = \frac{|Q \cap S|}{\sqrt{|Q| \times |S|}} \tag{4.9}$$

Algorithm 4.6 Natural Language Processing For Tweets and Web Page Titles

Require: *tweets* (*or web page titles*)

```
1: for all tweet ∈ tweets do
2:   if is tweet then
3:     Remove hash tags, and short URLs
4:     Replace @ with she
5:   end if
6:   tokens = Get_Tokens(tweet)
7:   tokens′ = Replace_Orthographic(tokens)
8:   line = Merge_Tokens(tokens′)
9:   sentences = Split_Line_To_Sentences(line)
10:  for all sentence ∈ sentences do
11:    tokens = Get_Tokens(sentence)
12:    tagged_tokens = Get_POS_Tagged_Tokens(tokens)
13:    nouns = Extract_Nouns(tagged_tokens)
14:    for all noun ∈ nouns do
15:      Remove_Punctuations(noun)
16:      Remove_Stop_Words(noun)
17:      if noun is not an English word then
18:        Spell_Check(nouns)
19:      end if
20:      noun = Get_Stem(noun)
21:      if length(word) < 2 then
22:        Remove noun
23:      end if
24:    end for
25:  end for
26: end for
```

Chapter 5

Experimental Results

5.1 The Results of The Discovery of Transient Local Communities

As mentioned in Chapter 3, this thesis has proposed an algorithm that modifies the min-cut based clustering algorithm with geo-location proximity to discover TLCs. Four experiments have been set up to assess whether the proposed GLC algorithm aids in the discovery of TLCs. The first experiment examined which damping parameter is suitable for the collected data. The second experiment examined whether the discovered communities have more abbreviations of place names in their messages. The third experiment examined whether these discovered communities enable us to explore local issues. Finally, the fourth experiment explored the most active users in the local communities.

5.1.1 The Damping Functions

This research uses 0.25 as the damping coefficient and chooses 24 hours (one day) as the damping time interval. The choice of these values can be explained as follows: as shown in Figure 5.1 (a), mostly the edge weights $w_t(u, v)$ are equal to 1, which means that most users updated only one message during the experiment time period (200 days). This research therefore hypothesizes that during a day some users updates only one to two messages. This implies that important topics may be missed if the algorithm updates weights every one hour.

This research has set the update time interval equal to one day and examined the damping results for the 200-day data under different damping coefficients. As demonstrated in Figure 6.3 (b) - (d), a suitable graph structure is obtained by choosing 0.25 as the damping coefficient. The graph contains certain amount of inactive users, but the amount is not large

enough to dominate the graph structure. With damping coefficients 0.1 and 0.2, there are still too many inactive users (2,500 - 4,500 user-pairs). The large number of inactive users lowers the performance of graph clustering and affects the quality of topics.

5.1.2 The Discovery of Place Names and Abbreviations

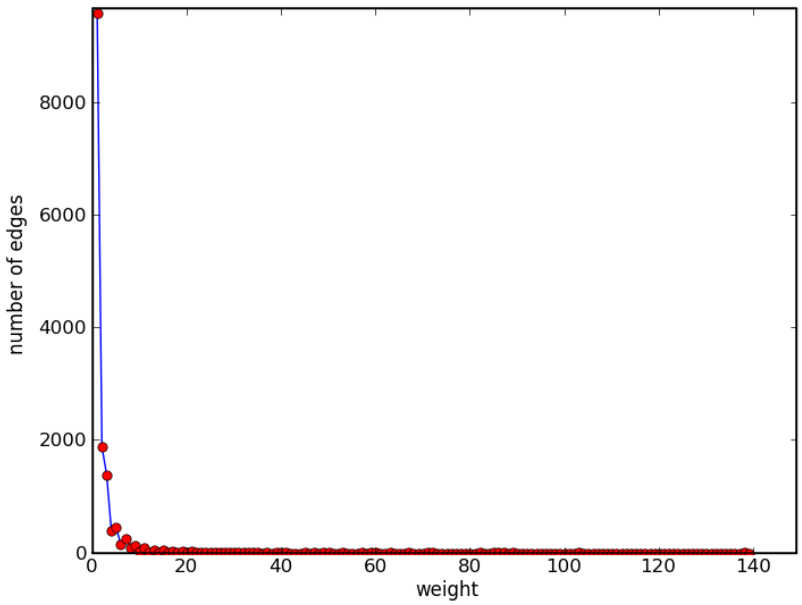
This research also hypothesizes that local people use more local words and abbreviations in their messages. If most users within the discovered communities are geographically close, the proposed algorithm should be able to identify more local words or place abbreviations.

However, it is difficult to discover all local words in the tweets. To simplify, this experiment examined certain place names and place abbreviations. Since the targets of this research is Calgarians, the abbreviations and place names that are interested are "calgary", "cal", "ab" and "yyc". These place names and place abbreviations were extracted from the tweets within communities discovered through both the GLC and the LC algorithm.

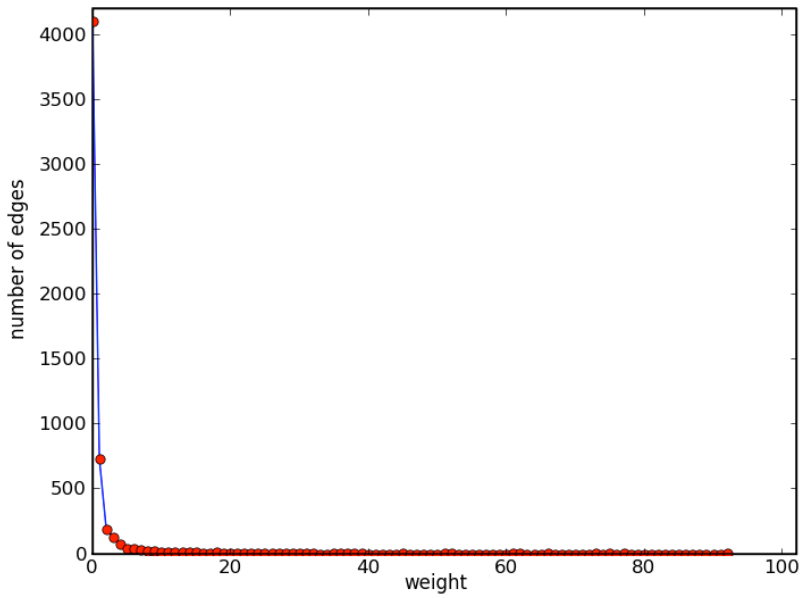
As shown in Figure 5.2, the red line and blue line present the numbers of place names and place abbreviations obtained from the TLCs and the CTCs discovered by the GLC algorithm and the LC algorithm. The results show that more place names and place abbreviations were discovered by the proposed algorithm. To provide more evidence to support the hypothesis, this experiment also examined whether the proposed algorithm identifies more local users.

Figure 5.3 demonstrate the numbers of local users identified by both algorithms. Results show that in general, the GLC algorithm outperforms the LC algorithm. The GLC algorithm discovered two to three times more users. In addition, by overlaying Figure 5.2 with Figure 5.3 one can see that the values of zero in Figure 5.2 (i.e., no place names and place abbreviations discovered) match the positions in Figure 5.3, where no local user is discovered.

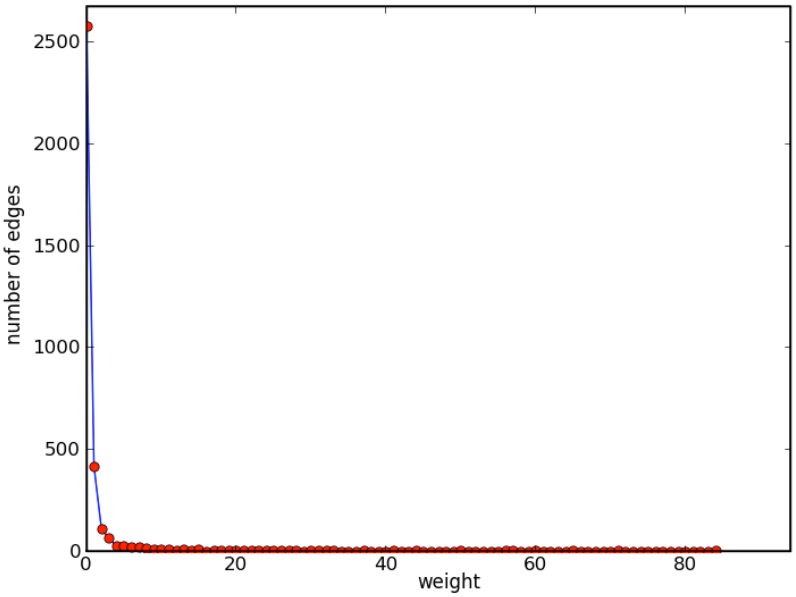
The results provide evidences that Calgarians have contributed more place names and abbreviations. This finding supports the finding from Lieberman [73]: local people are interested in local issues in OSNs. In fact, the proposed GLC algorithm discovered more



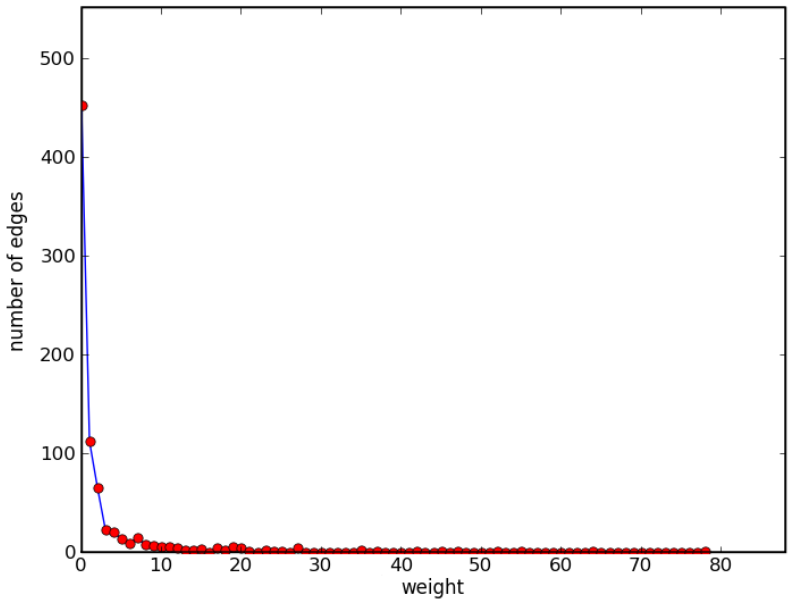
(a) Distribution of connections (weights) for the 200-day Twitter data



(b) Distribution of connections with a damping coefficient of 0.1



(c) Distribution of connections with a damping coefficient of 0.2



(d) Distribution of connections with a damping coefficient of 0.25

Figure 5.1: Distributions of Connections Formed after Damping the 200-day Twitter Data

Sub-graphs demonstrate the weight distributions obtained by applying different damping coefficients. As the value of damping coefficient increases, the number of inactive users decreases

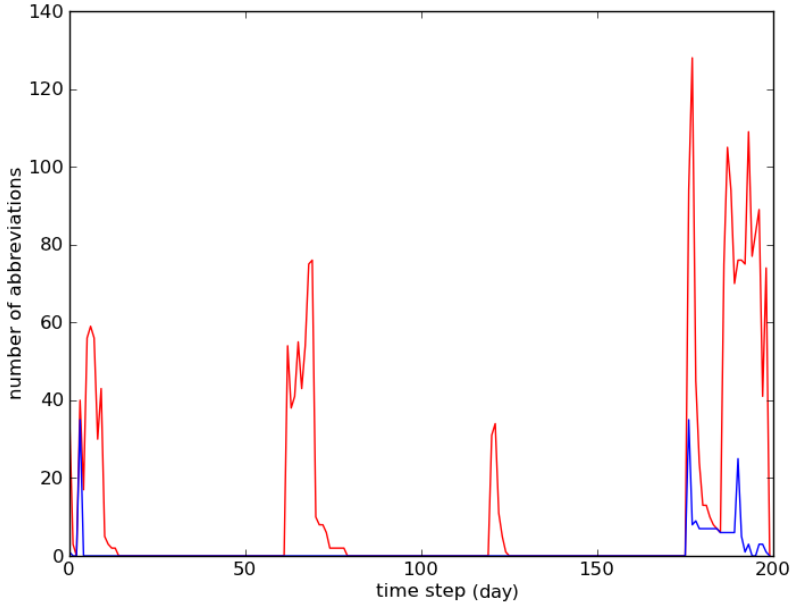


Figure 5.2: Numbers of Local Place Names and Abbreviations Discovered within Communities

The red line represents the numbers of local place names and place abbreviations obtained from TLCs, and the blue line represents the number of local place names and place abbreviations obtained from CTCs

local community names in the TLCs, e.g., "Brentwood", "Crowfoot", "Shaganappi", but these words were not included in this comparison.

5.1.3 The Discovery of Community Topics

In addition, this research hypothesizes that the discovered TLCs allow us to explore more local issues. This experiment examined the most commonly used words in the communities discovered through both algorithms and compared the results. Here, a word is regarded as valid if the word appears in more than three tweets. The valid words were sorted according to their frequency of use in ascend order (Table 5.1 and 5.2).

In this experiment, ten days (time steps) were randomly selected, and the words discovered within the TLCs and the CTCs at these time steps has been examined and compared.

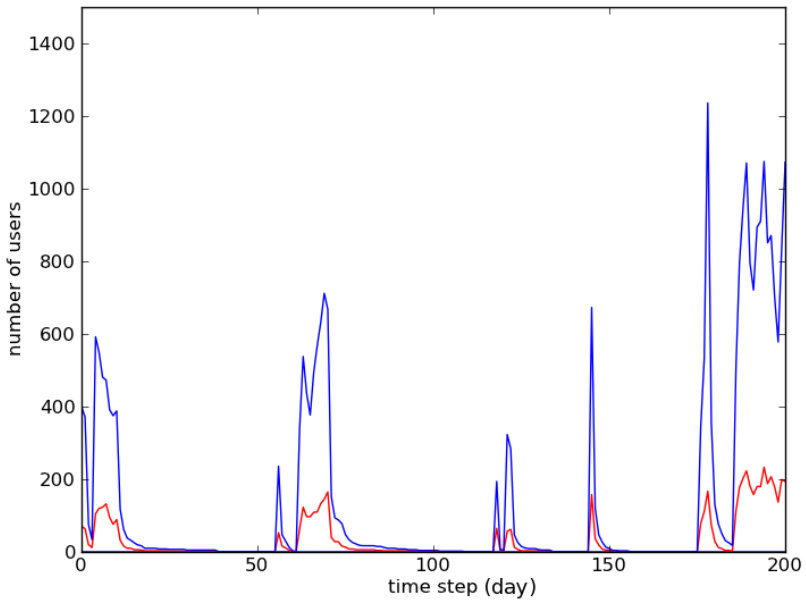


Figure 5.3: Numbers of Local Users Discovered within Communities

The blue line represents the numbers of local users discovered through the GLC algorithm, and the red line stands for numbers of local users discovered through the LC algorithm. Results show that the GLC algorithm outperforms the generic algorithm in discovering local users

Table 5.1 and 5.2 show the words discovered from the CTCs and the TLCs. The results support the hypothesis. The TLCs allow us to explore more words that are relevant to the geographical area being studied. As shown in Table 5.1, the words discovered in the TLCs are more meaningful. In many cases, these words represent local events or interests. On the contrary, the words discovered in the CTCs (Table 5.2) are random and difficult to get meanings out of them.

For example, the results in Table 5.2 show that at time step 7, the words extracted from the CTCs are "one find"; and at time step 195, the words extracted are "friend damn like news". Similar results occur at time step 6, 10, 68, and 196. This information is not sufficient enough to understand the communities.

In contrast, the GLC algorithm can find more meaningful information to understand the

communities. For example, the results at time step 63 show that the proposed algorithm discovered "awesome made next secondary like videos yyccc eu vou enmax chinatown hprodeo love live great parade calgary yyc". By using these words as keywords to search the tweets in the TLCs, one can find that some local users were talking about Enmax's CEO at time step 63. At the same time, some other local users were exchanging messages about a parade in Calgary Chinatown. This experiment further examined the contributors of these tweets and observed that they are not the same person. Different users with different tweets contribute these keywords. For example, the tweets that contain "Enmax" at time step 63 are "on holden's departure from #enmax #yyccc #yyc", "for this journalism exclusive: enmax ceo's monaco trip questioned", "breaking news: embattled enmax ceo gary holden is immediately stepping down from the company" and so on.

Similarly, the results at time step 177 show that both the GLC algorithm and the LC algorithm captured the Citadel fire event (Citadel is a community name in Calgary). Both algorithms discovered "citadel". When tracing back to the tweets that contain the keyword, one can find that people were talking about the fire at Citadel, and the information was spreading by different users. Since the GLC algorithm extracted more local users, the frequencies of use of the keywords in Table 5.3 are much higher than that are in Table 5.2. In addition, the words extracted by the GLC algorithm provide more detail. For example, the GLC algorithm discovered "nw", which is Caglary's northwest district where the community locates.

The results at time step 196 are much more interesting. The words discovered in the TLCs are geographically meaningful. Local users can easily make sense of them. For example, a local user can intuitively group "june" and "mountain and "flames and "canucks" together. This is because these words are the keywords in their daily lives. Calgarians often go to mountains in summertime, and Calgary hockey fans keep their eyes on Vancouver Canucks hockey team and Calgary Flames hockey team. When tracing back to the tweets that contain

Table 5.1: Most Commonly Used Words Extracted by Using the LC Algorithm

Time step	List of commonly used words (frequency from low to high)
6	one
7	one find
10	las el de para tampico
63	N/A
68	tudo bem
121	N/A
177	on-scene news clean-up begins thoughts video flames watching massive blaze multiple sure multi-home families fire homes citadel calgary yyc
187	mas para su lols attendees yyc leigh chang en decir cioconference de que
189	like attendees yyc leigh chang cioconference
196	damn either twitter green today news good city lol old ok

the keywords, one can find that local users were exactly exchanging message about hiking and hockey.

Results in the tables also demonstrate that the GLC algorithm is able to discover city name and place abbreviations such as "yyc" and "ab". And usually these place names have the highest frequencies (i.e., appears in the most tweets).

5.1.4 The Discovery of Locally Active Users

This research also aims to explore the most active users in the local communities. Another experiment was set up to examine active local users. The experiment examined the number of connections that the CTC and TLC users have. Here, the users who have less than three connections are regarded as inactive users; otherwise, the users are regarded as active users. Table 5.3 and 5.4 order the active users by the numbers of connections in descend order. The sample time steps in this experiment are the same as the time steps selected in Table 5.1 and 5.2.

As shown in Table 5.3 and 5.4, the GLC algorithm allows us to discover more locally active users. For example, the results at time step 10 demonstrate that the GLC algorithm discovered the following users "melroseredmile jaiboencanada harpsinyyc markusoff vixterl calgarybeacon yycgal klaszus turnipthebeets", while the LC algorithm only discov-

Table 5.2: Most Commonly Used Words Extracted by Using the GLC Algorithm

Time step	List of commonly used words (frequency from low to high)
6	ableg cookies lol night shop stuff yeg new calgary yyc
7	happen job fast hope definitely believe nov dec great wonderfully succinct blog post homophobia lol yeg salvation army night ableg calgary yyc
10	event love night mentions poor city website stream loses connection sue water mobrobash sure las para tampico yyccc calgary yyc
63	awesome made next secondary like lol videos yyccc eu vou enmax chinatown hprodeo love live great parade calgary yyc
68	year job health tonight new 515 well launch regina picture back tweerdos prfightclub infrastructure make fluoride sure would end tudo great bem yyccc good calgary yyc
121	look beautiful yyccc lol ab canada calgary yyc
177	book week street effective moves soon ab watching nw work guys event awesome latest yyccc video crowd trying tame multi-home home news families coming yeg fire live well flames multiple blaze massive citadel calgary yyc
187	police snow parenting asi hubiera por eso yyccc life forecast looking calgaryill forrest sum igual saludos attendees play drums shake soggy blues festivals great still lots ill 630pm rainy wait wet cioconference june dance sambafied rain summer que calgary yyc
189	news tunnel twitter civil food eso year listen great whats space sunshine tunderpass video attendees trending obviously sunday sun cioconference amazing yyccc que calgary yyc
196	tweet site challenge june campaign mountain canada watch wear hear comments #fcmhfx fag police discrimination launch top wait happy win damn time miss flames hat love around stn jersey rock good green canucks show city rights gay help today calgary yyc

ered “jaiboencanada”. Similar results occurs at time step 187, the GLC algorithm discovered “misskatsuragi kuwindayyc melroseredmile cioremotecomm ericksk8r grdrollerderby mediapirate alex_ruiz cbcoutside vanityplatefail calgaryvipers ericmusicbaum 660new zackhewitt thedirtyground”, whereas the LC algorithm only discovered “misskatsuragi cioremotecomm rigstarcomminc”.

In addition, this experiment attempted to find the dominant local users in the discovered TLCs at each time step. For instance, from time step 6 to 10 (in Table 5.4), the following users “vixterl”, “calgaryherald”, “melroseredmile”, “calgarybeacon” and “nenshi” frequently appeared and highly communicated with others. One can identify that “calgaryherald” and “calgarybeacon” are local media, and “melroseredmile” is a local business.

From time step 63 to 68, the dominant users in Table 5.4 are “nenshi”, “calgarybeacon” and “polyticaltdoes”. One can identify that “nenshi” is the mayor of Calgary and “polyticaltoes” and “calgarybeacon” are local media. And from time step 177 to 200, the dominant user is “nenshi”, and there is no particular local medium dominating the communities. In summary, local media play a very important role in forming TLCs. However, the GLC algorithm found that Calgary mayor (Naheed Nenshi) was always one of the most active users during the experimental time period. This finding echoes what media believes that the reason Nenshi won the mayoral election is partially due to his strong presence in new social media, such as Facebook and Twitter [7].

Table 5.3: Most Active Users Extracted by the LC Algorithm

Time step	List of active users (frequency from high to low)
6	vixterl twowheelgeek michellescabar gregtwhite
7	vixterl twowheelgeek jonincalgary michellescabar gregtwhite
10	jaiboencanada otto_ivan powertowe pptorresm
63	N/A
68	gabi_reznik polyticaltoes figueiredo_d brunogarcia0 lucas_daloia coracoraline fuc_kin
121	N/A
177	bizboxtv lisaostrikoff cassieneil missmitto thedawe darren_krause jjkcanada
187	misskatsuragi heroofwarfare cioremotecomm _fabrizi0_ rigstarcomminc migueltha alcachofasesina aasthaae belindapop
189	heroofwarfare kaaykaay_ cioremotecomm brokenmason rigstarcomminc erikaaamcarthur fabrizi0 draytongaga migueltha xxsaaarahx tooterser101
196	musicbaum that_angela toriklassen maplebot ladyseraphina kikkiplanet

Table 5.4: Most Active Users Extracted by the LC Algorithm

Time step	List of Active Users (frequency from high to low)
6	vixterl calgaryherald melroseredmil calgarybeacon espyexperience kimmyneutron standardtoaster a_picazo robertmcbean jabest jasonvanrassel
7	vixterl jillianwalker polyticaltoes calgaryherald melroseredmile kimmyneutron volunteercal wendy drewpanderson wildrosebrewery calgarybeacon electdanielle jabest darren_krause
10	melroseredmile jaiboencanada harpsinyyc markusoff vixterl calgarybeacon yycgal klaszus turnipthebeets
63	thorntales nenshi collenbe guivendrame calgary marketcreperie polyticaltoes calgarybeacon uofcprojectnext yyc_counselling hprcalgary formerjourno
68	gabi_reznik nenshi openfilecgy calgaryabhomes polyticaltoes calgarybeacon robertmcbean ucalgary fergusonabhomes calgarytoday theatrecalgary calgaryabcondo anime_addict_a_picazo twscalgary kidsincowtown lauriegriffinpr klaszus jabest guivendrame 88styles jcvphotography
121	yyctwestival roadhousecgy mountroyal4u marquisgroup highergcafe trondfrantzen gccarra lisaostrikoff calgarybizcaf weebdacat
177	gostlund bizboxtv lisaostrikoff melroseredmile cassieneil eves1 abfashionweek missmitton globalcalgary nenshi kerjayyc prairies amberschinkel calgaryfolkfest thedawe darren_krause jjkcanada
187	misskatsuragi heroofwarfare kuwindayyc melroseredmile cioremotecomm ericksk8r grdrollerderby mediapirate alex_ruiz eyeopenerbob cbcoutside vanityplatefail calgaryvipers ericmusicbaum julianbrass 660news zackhewitt thedirtyground
189	misskatsuragi heroofwarfare jaiboencanada nenshi kaaykaay_ aldjohnmar yycsamba cioremotecomm ericksk8r yycgal endeavorarts kylaadelilah rozzielee calgarytransit melroseredmile asifpremji kool1015calgary globalcalgary fulltimeslacker calgarymarathon
196	musicbaum nenshi fastandramos kuwindayyc asifpremji distress_centre endeavorarts that_angela artcentralyyc jasonl_ markusoff ivape_ca cameraguyrob kidsupfrontcalg edmcapitals calgarytoday gostlund makeoneyarns justinbieber

5.2 The Results of The Discovery of Local Communities of Interest

As mentioned in Chapter 4, this thesis has proposed framework to discover communities of interest in the collected LSN data. This research has employed the NLP techniques to clean the short texts and proposed a graph clustering algorithm to enhance the fast-greedy optimization of modularity (FGM). In addition, this research has also proposed different ways to build sociograms. This research conducted several experiments to evaluate the performances of the enhanced FGM algorithm and the usefulness of the sociograms.

5.2.1 Evaluation of Generic and Enhanced Fast-greedy Optimization of Modularity

5.2.1.1 Simulation

To test the performance of the enhanced FGM algorithm, one small experiment was set up to examine the clustering results for a simulated graph. The small social graph consists of twelve vertices and fifty edges (connections) was generated (Figure 5.4), and there are three communities ([K,L] [C,E,F] [B,D,A,G,I,J,H]) for this simulated graph. The users in the communities were assigned with the well-identified accounts in the collected data, and their text resources were assigned with the web page titles that well demonstrate the users' interests. The purpose of this simulation is to investigate how do text similarity measures affect clustering results.

Four scenarios were created to test the sensitivities of the FGM and the enhanced FGM. Table 5.5 shows the simulation results. As shown, the connection between E and G is too weak, so neither the FGM and the enhanced FGM have the ideal clustering results (i.e., [K,L] [C,E,F] [B,D,A,G,I,J,H]). However, in this simulation, the FGM was very sensitive to the weight changes (i.e., intensity of communications), which means that the FGM may not be appropriate for clustering noisy graphs. On the contrary, the enhanced model provided more stable clustering results. The enhanced FGM was not as sensitive as the FGM, and it was flexible enough to adjust the clustering results when the graph structure changed (e.g.,

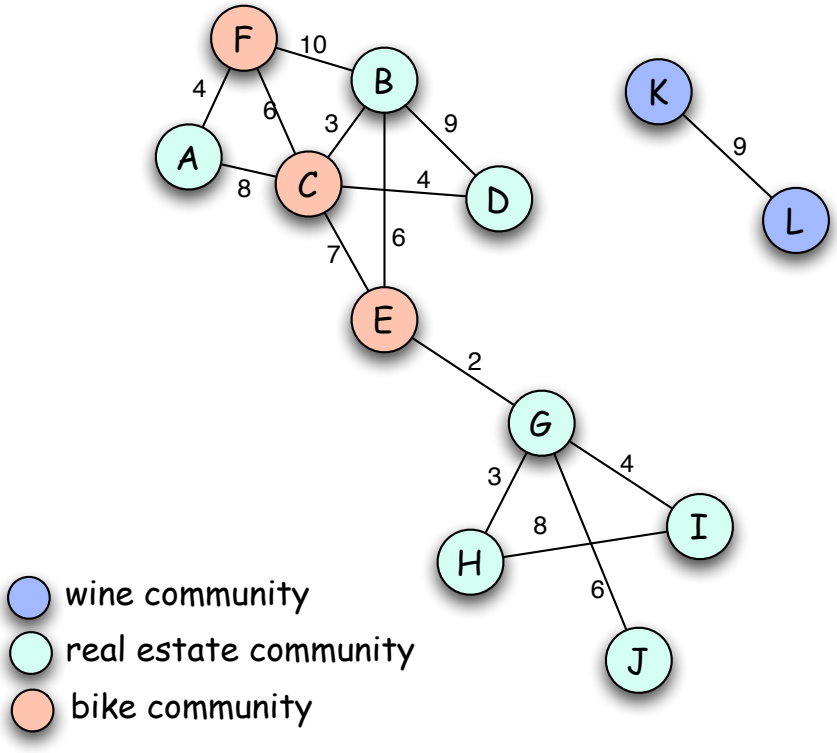


Figure 5.4: The Simulated Graph

results were modified after inserting AB).

Table 5.5: The Simulation Results

Scenario	Generic FGM	Enhanced FGM
original	[K,L] [B,D,F] [A,C,E] [G,I,J,H]	[K,L] [B,D] [A,C,E,F] [G,I,J,H]
change BF to 7	[K,L] [B,D,A,C,E,F] [G,I,J,H]	[K,L] [B,D] [A,C,E,F] [G,I,J,H]
change BC to 7	[K,L] [B,D,A,C,E,F] [G,I,J,H]	[K,L] [B,D] [A,C,E,F] [G,I,J,H]
change BF to 3	[K,L] [B,D,E] [A,C,F] [G,I,J,H]	[K,L] [B,D] [A,C,E,F] [G,I,J,H]
Insert AB = 2	[K,L] [B,D,A,C,E,F] [G,I,J,H]	[K,L] [B,D,A,C,E,F] [G,I,J,H]

5.2.1.2 Real World Scenarios

After testing the simulated graph, another experiment was set up to examine the real word scenarios using the collected Twitter data. In this experiment, NLP techniques have been applied to clean the short texts (i.e., tweets and web page titles) and get bags of words

for the users (i.e., the collected accounts, see Chapter 4.1). This research also applied the enhanced FGM algorithm with three different similarity measures: Dice, Cosine, and Overlap similarity measure, to compare the performances of the enhanced FGM and the FGM algorithm in terms of their ability to discover CoIs. Three scenarios were examined: 1) Mention graph with edge weights larger than two. In this scenario, the users who have communicated with each other more than two times are regarded as valid relationships, 2) Mention graph with edge weights larger than ten, and 3) the whole URL prefix graph. To evaluate, this experiment examined the fraction of the discovered communities that are relevant to a list of authoritative communities under different conditions.

In this experiment, the users' interests were identified by manually checking their sites. This experiment summarized users interests and created a list of authoritative communities. For example, users who have interests such as "web design", "web developer", "interactive design", "graphic design", and "technology" are grouped into one community and users who have interests or jobs such as "journalist", "writer", "journalism", "story teller", "publisher", "reporter", and "editor" are grouped into another community. There were total 30 authoritative clusters in the collected dataset.

Since the enhanced FGM algorithm integrates text similarity measures into the FGM, the enhanced algorithm has to determine when to merge two communities based on the text similarity and when to merge them based on the graph structure. As mentioned in Chapter 4, T determines whether the text similarity between a pair of communities is similar enough to increase the priority to be merged. Since the authoritative communities for users were determined, this experiment examined the distributions of the text similarities between users who belong to the same the CoIs (i.e., authoritative communities) and determined the T values based on the distribution.

In this experiment, the text similarities between the same CoI users were calculated with the three similarity measures mentioned previously (i.e., eq. 4.7-4.9). Because this research

uses the tweets and the web page titles as the text resources, this experiment has examined the distributions for both text resources and chose the T values based on the distributions (Figure 5.5).

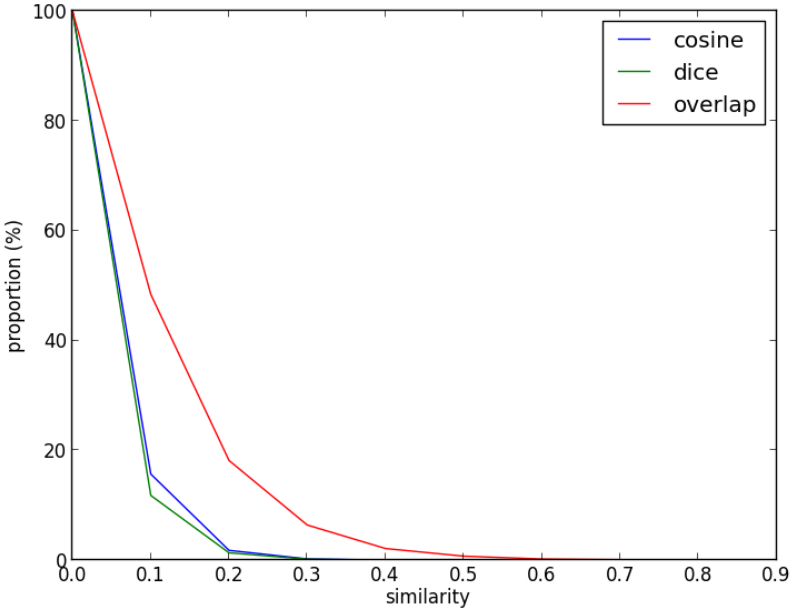
Figure 5.5 (a) shows the distributions of the similarity values derived from the web page titles, and Figure 5.5 (b) shows the distributions derived from the tweets. As shown in both figures, the distributions for the three similarity measures have similar trends. Mostly the similarity values are less than 0.1 in Figure 5.5 (a), and the similarity values are less than 0.2 in Figure 5.5 (b). Consequently, this experiment used 0.05 and 0.1 as the T values when running the enhanced FGM algorithm with the web page titles, while used 0.1 and 0.2 when running the algorithm with the tweets.

Figure 5.6 to Figure 5.8 show the results for the enhanced FGM algorithm. These figures demonstrate the numbers of CoIs discovered under different purities [1]. Since the research aims to examine whether the enhanced FGM enables to discover more CoIs, this experiment examined the fraction of the discovered communities that are relevant to a list of authoritative communities under different purities. The more the numbers of CoIs are discovered under higher purities, the more efficient the algorithm is in discovering CoIs.

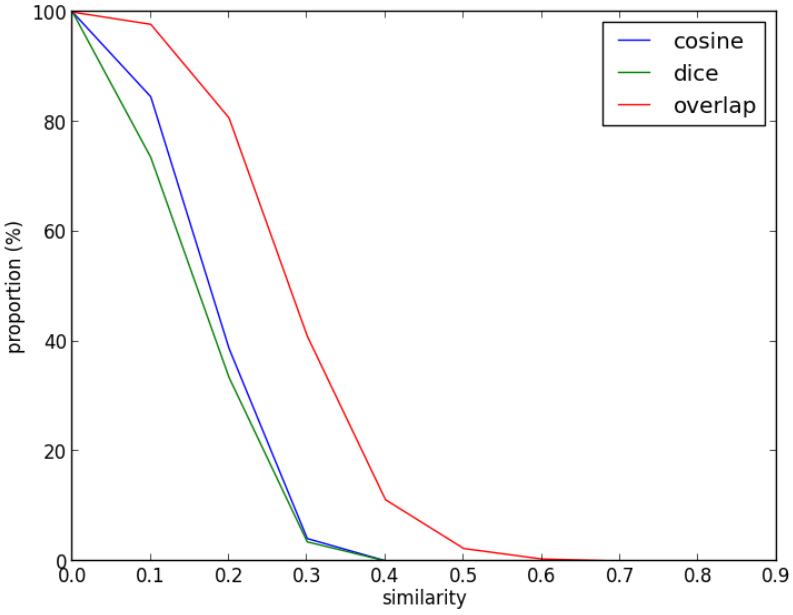
Figure 5.6 demonstrates the results for the first scenario (Mention graph with edge weights larger than 2). Results show that the enhanced FGM algorithm always outperforms the generic FGM, indicating both the tweets and the web page titles help in discovering CoIs. However, the results for the three similarity measures are different. This is because different similarity measures score similarities differently, and the orders for merging pairs of user are not the same (see Algorithm 4.5). Nevertheless, the overall numbers of CoIs identified and the numbers of CoIs identified under higher purities are larger than that identified by FGM algorithm.

Figure 5.7 demonstrates the results for the second scenarios (Mention graph with edge weights larger than 10). It is not surprising that the results show that the enhanced FGM

[1]Purity measures the fraction of the largest class to the overall cluster size



(a) Web page titles



(b) Tweets

Figure 5.5: Distribution of Text Similarities

This graph demonstrates the distribution of text similarities between users that belong to the same authoritative clusters. The first graph is the result based on web page titles, and the second graph is the result based on tweets

algorithm performs similarly to the FGM algorithm. This is due to the fact that when the graph is constructed based on the first scenario, the result is a very connected graph, with non-obvious community structures. The FGM algorithm failed to capture the communities accurately. On the contrary, when the graph was built based on the second scenario, the community structures became obvious, so the FGM algorithm performed better.

The results also show that the text similarity measures are more powerful for complex graphs. Nevertheless, for most cases, the overall numbers of CoIs identified and the numbers of CoIs identified under higher purities are larger than that identified by the FGM algorithm.

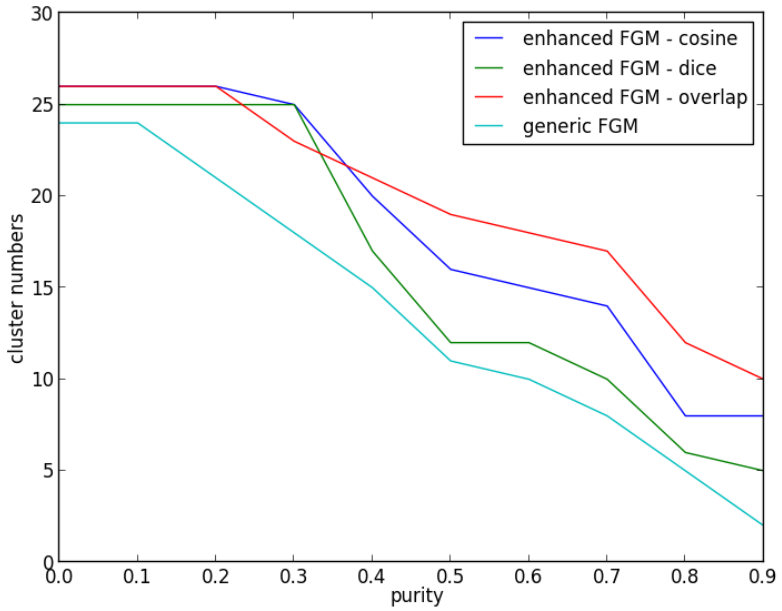
Figure 5.8 demonstrates the results for the third scenario (whole URL prefix graph). The results are similar to the results for the second scenario. This is because the community structures in the URL prefix graph is obvious, indicating that URL prefixes are good resources to identify CoIs.

Overall, the experiment results show that there is no best similarity measure. This is due to the fact that different similarity measures score similarities differently. For example, Dice similarity measure multiplies the frequencies of use of common words between users with two (eq. 4.7), so that the similarities between users who have more common words with higher frequency of use are higher than that between users who have more common words with lower frequency of use.

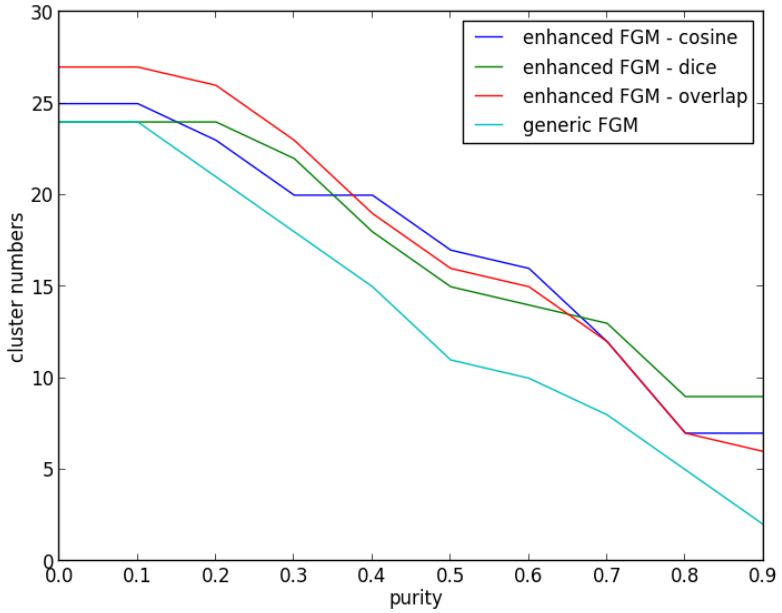
On the contrary, the Overlap and Cosine similarity measures (eq. 4.8 and 4.9) lower the denominator in order to smooth out the effects of large bag sizes. Since each of them determines the similarities differently, there are always ambiguities in the results, especially when the text inputs are not perfectly reflecting users' interests. For example, two bags of words may have higher similarity because there are a lot of common words with frequency equal to one. However, not all the frequency-one words indicate users' interests. Examples of noun are "library", "price", and "view".

Nevertheless, it may not be appropriate to remove words with lower frequencies because

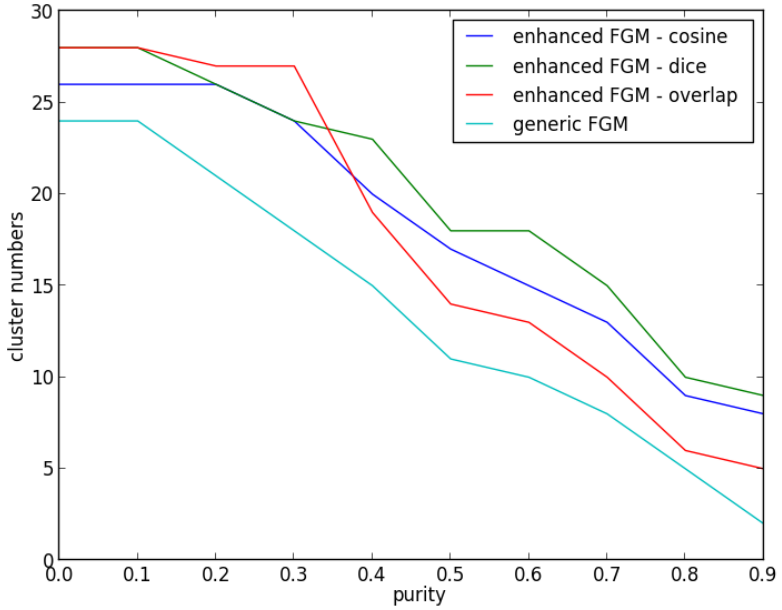
the words also help in indicating the similarities between users' interests. Examples of word are "bbq", "beer", "gallery", "cfl", and "dino". Usually, the similarities between these words are not recorded appropriately in word dictionary.



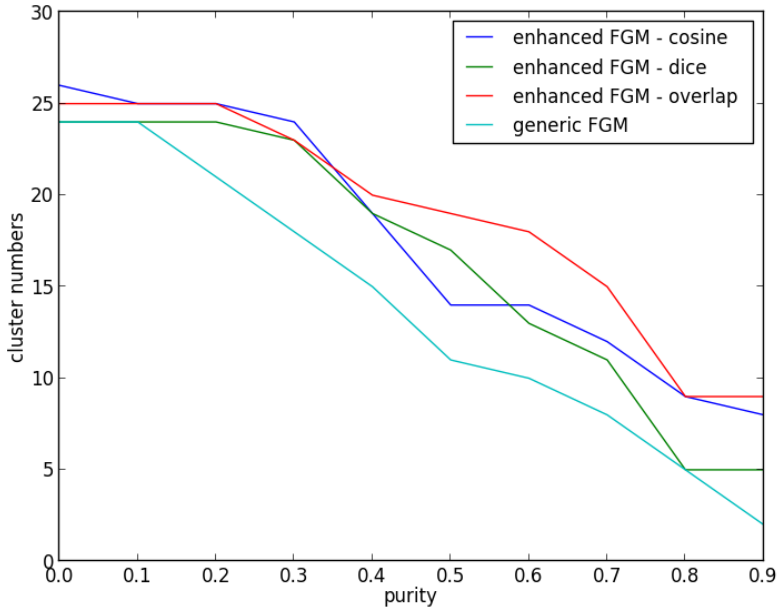
(a) Numbers of communities discovered by using 0.1 as the threshold (T). The text inputs are tweets



(b) Numbers of communities discovered by using 0.2 as the threshold (T). The text inputs are tweets



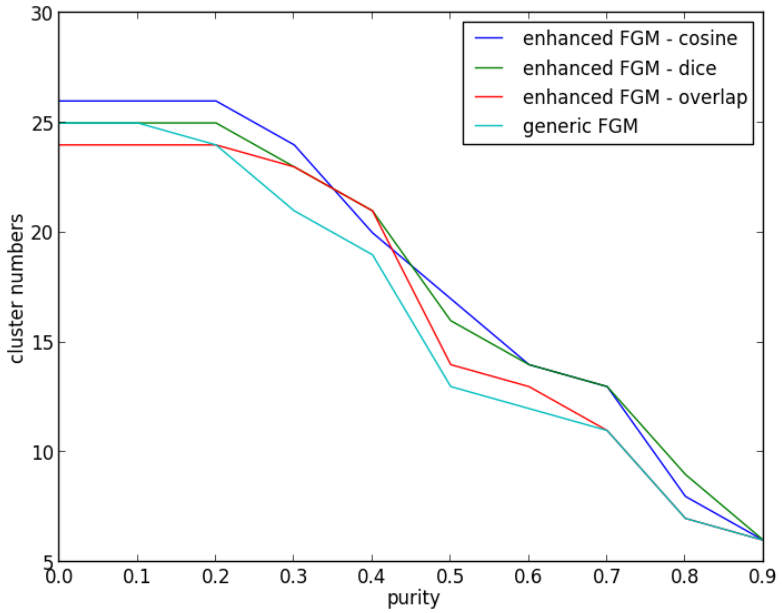
(c) Numbers of communities discovered by using 0.05 as the threshold (T). The text inputs are web page titles



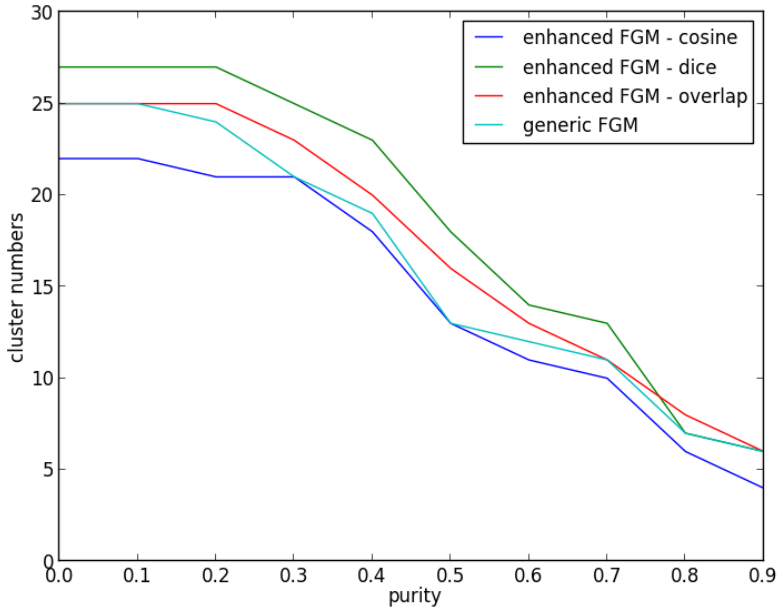
(d) Numbers of communities discovered by using 0.1 as the threshold (T). The text inputs are web page titles

Figure 5.6: Results for Scenarios One: Mention Graph with Edge Weights Larger Than 2

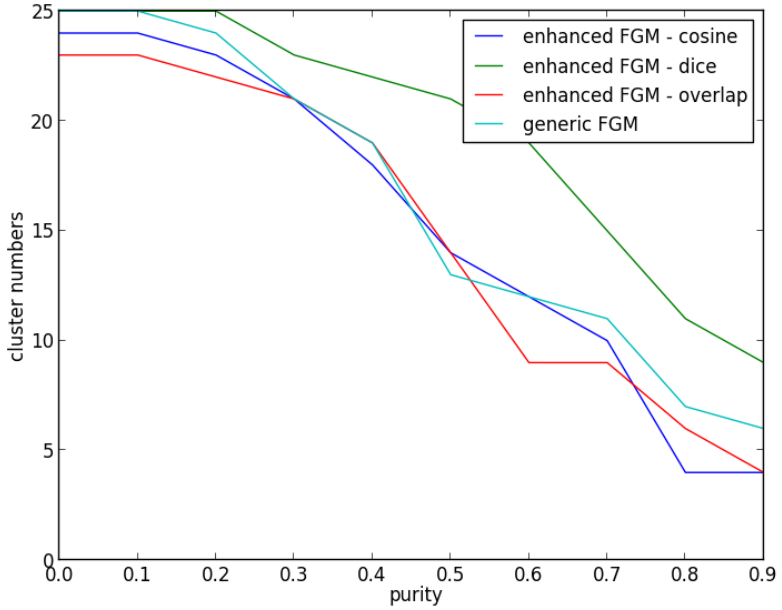
Sub-graphs demonstrate the numbers of communities discovered under different T values and text resources. Results show that the proposed algorithm outperforms the FGM.



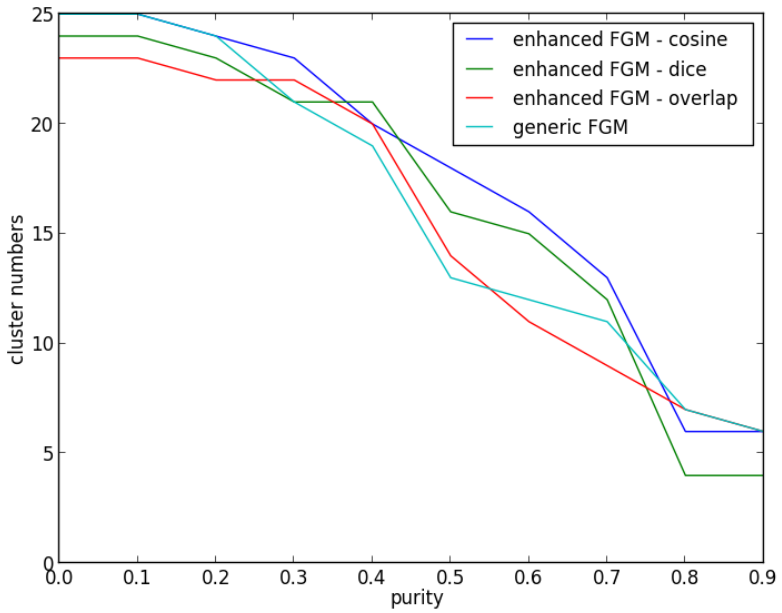
(a) Numbers of communities discovered by using 0.1 as the threshold (T). The text inputs are tweets



(b) Numbers of communities discovered by using 0.2 as the threshold (T). The text inputs are tweets



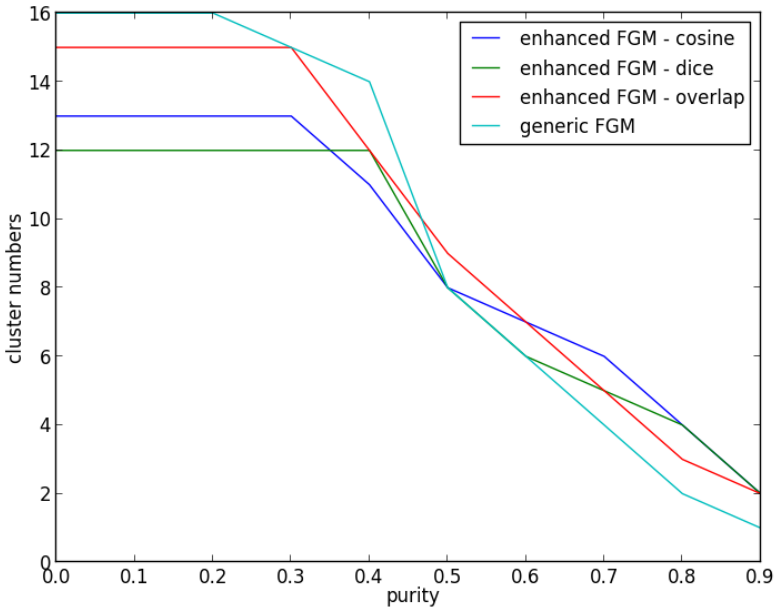
(c) Numbers of communities discovered by using 0.05 as the threshold (T). The text inputs are web page titles



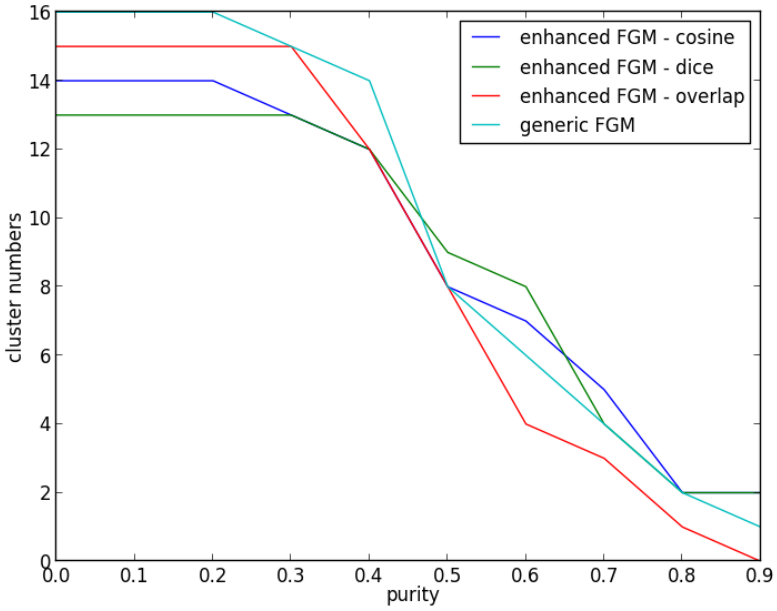
(d) Numbers of communities discovered by using 0.1 as the threshold (T). The text inputs are web page titles

Figure 5.7: Results for Scenarios Two: Mention Graph with Edge Weights Larger Than 10

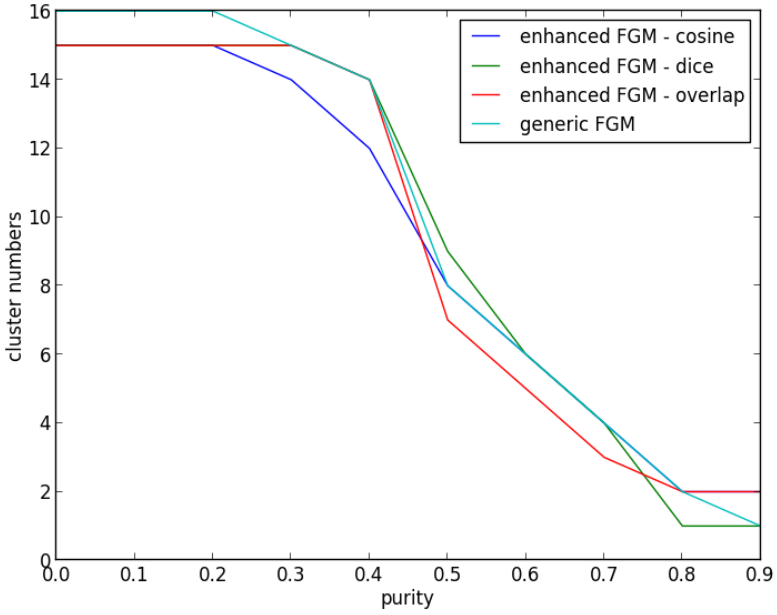
Sub-graphs demonstrate the numbers of communities discovered under different T values and text resources. Results show that the proposed algorithm performs similarly to the FGM.



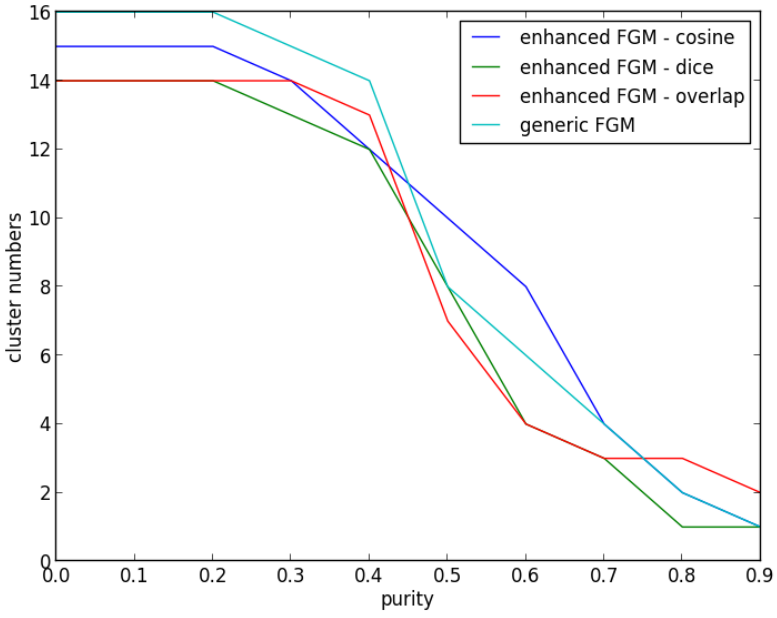
(a) Numbers of communities discovered by using 0.1 as the threshold (T). The text inputs are tweets



(b) Numbers of communities discovered by using 0.2 as the threshold (T). The text inputs are tweets



(c) Numbers of communities discovered by using 0.05 as the threshold (T). The text inputs are web page titles



(d) Numbers of communities discovered by using 0.1 as the threshold (T). The text inputs are web page titles

Figure 5.8: Results for Scenarios Three: Whole URL Prefix Graph

Sub-graphs demonstrate the numbers of communities discovered under different T values and text resources. Results show that the proposed algorithm performs similarly to the FGM.

Another experiment was further executed to combine the tweets and the web page titles to provide new text similarity measures. The new similarity measures are the summations of the text similarity measures of the tweets and of the web page titles (eq. 5.1).

The purpose of this experiment is to evaluate whether the combination of tweets and web page titles improves the clustering results. This experiment applied Dice, Cosine, and Overlap similarity measure to eq. 5.1 and chose 0.15 and 0.25 as the T values (between 0.15 (0.05+0.1) and 0.3 (0.1+0.2)).

$$New\ Similarity\ = Text\ Similarity(tweets) + Text\ Similarity(web\ page\ titles) \quad (5.1)$$

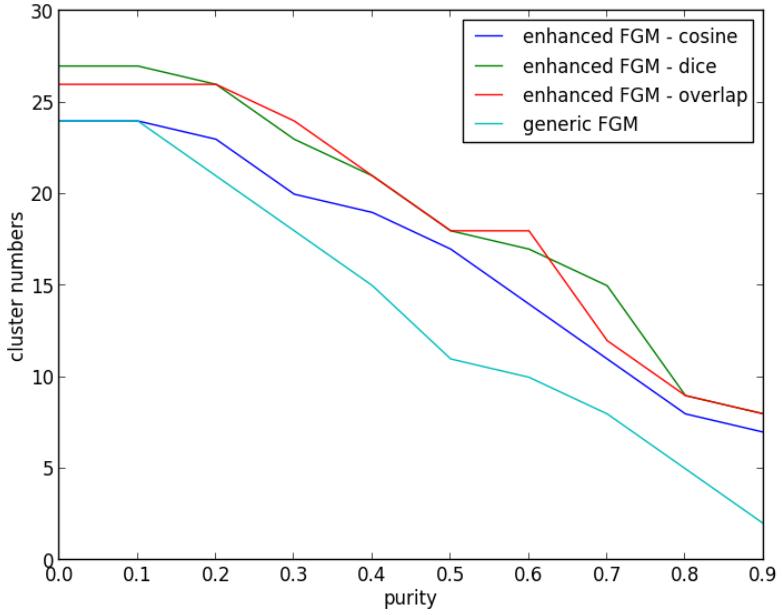
Where Text Similarity $\in$ Dice, Cosine, Overlap

To evaluate the performances, the clustering results for the first scenario and the third scenario were examined and compared (i.e., compare the results in Figure 5.9 and Figure 5.10 with the results in Figure 5.6 and Figure 5.8). The results demonstrate that the combination of the tweets and the web page titles improves the clustering results. Comparing to the results in Figure 5.6 and Figure 5.8, the new clustering results for the three similarity measures are more consistent, demonstrating the combination of tweets and web page creates a more reliable text resource. Nevertheless, the clustering results are still not identical due to the different clustering orders.

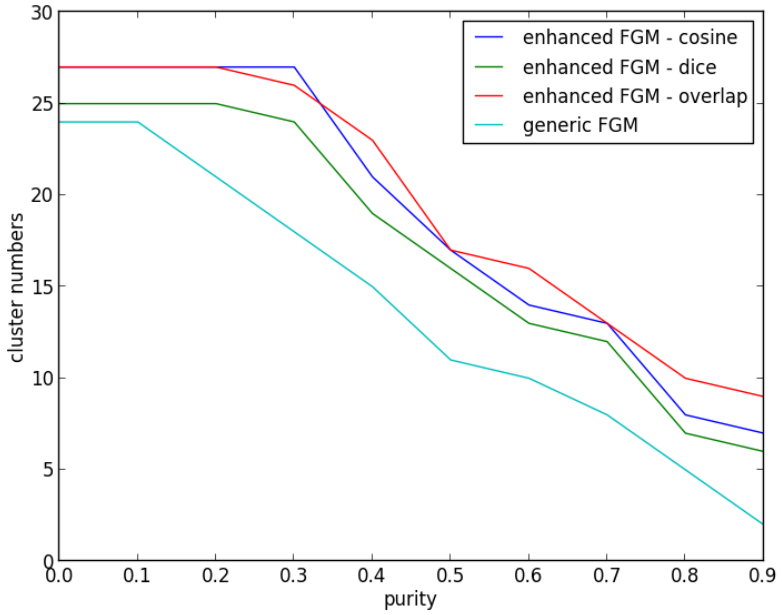
The results in Figure 5.10 show that the enhanced FGM algorithm does not identify more clusters comparing to the FGM algorithm. Again, this is because the community structures in the URL prefix graph are strong, and the strong community structures reduce the power of the text similarity measure.

It is not surprising that the new similarity measure performances better. To meet the new T values (0.15 and 0.25), users' shared contents have to be more similar. An example case for a pair of users to meet the new T values is, the similarity of the tweets between the users is larger than 0.5, and the similarity of the web page titles between them is higher than 0.1. Consequently, the similarities measured by the new methods are more reliable than that

measured by the methods relying on any of them.



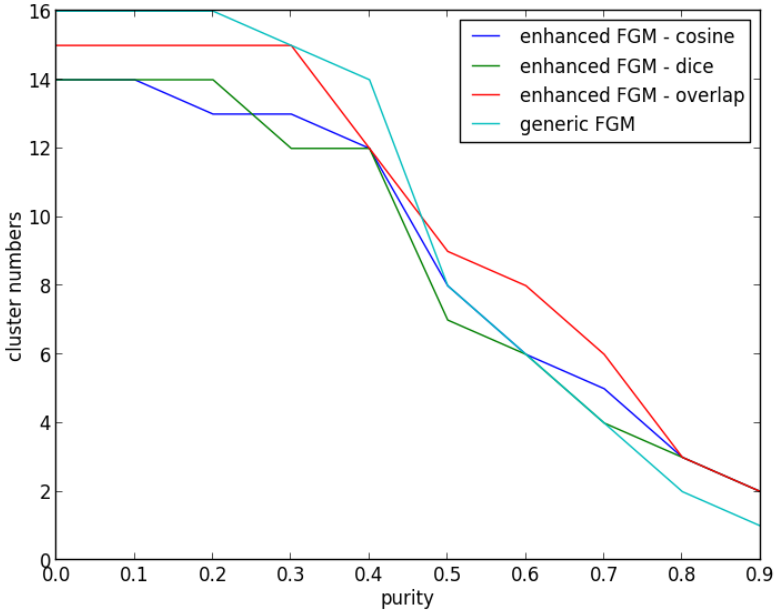
(a) Numbers of communities discovered by using 0.15 as the threshold (T). The text inputs are both web page titles and tweets



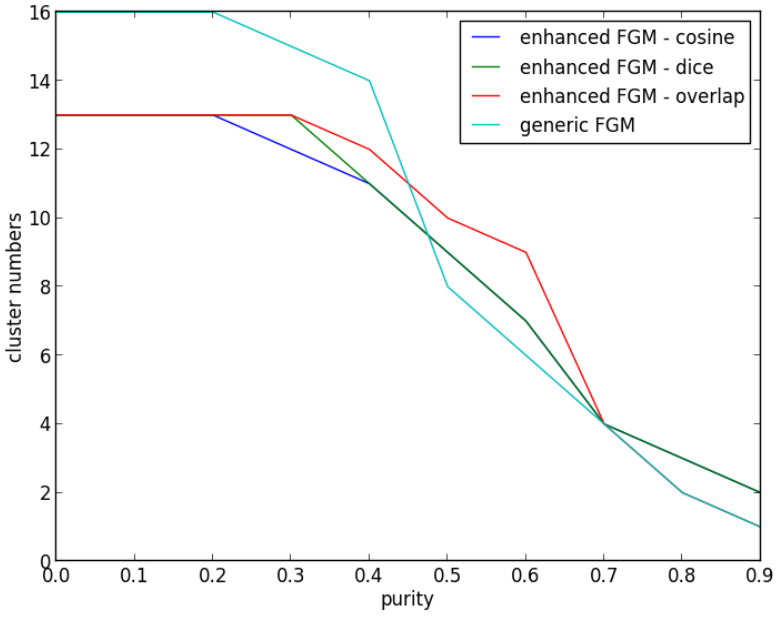
(b) Numbers of communities discovered by using 0.25 as the threshold (T). The text inputs are both web page titles and tweets

Figure 5.9: Results of New Text Similarity Measure - Scenario One

Sub-graphs demonstrate the numbers of communities discovered by using the new text similarity measure. Clustering results for the three similarity measures are more consistent.



(a) Numbers of communities discovered by using 0.15 as the threshold (T). The text inputs are both web page titles and tweets



(b) Numbers of communities discovered by using 0.25 as the threshold (T). The text inputs are both web page titles and tweets

Figure 5.10: Results of New Text Similarity Measure - Scenario Three

Sub-graphs demonstrate the numbers of communities discovered by using the new text similarity measure. Clustering results for the three similarity measures are more consistent.

Chapter 6

Conclusions and Future Work

This thesis has investigated three research topics related to two types of crowdsourcing data: VGI and social media. The first topic focuses on the design and implementation of a crowd-based GIS for transportation. The second topic focuses on the discovery of transient local communities. The third topic focuses on the discovery of communities of interest in LSNs.

The first part of this thesis started investigating the contribution of VGI to transportation systems in 2009, when several researches started exploring the use of VGI in transportation [93]. TrafficPulse, a completed front-end and back-end GIS, was designed and implemented to overcome the limitation of existing systems for transportation at the moment.

In order to explore the capability of Smartphones for traffic data collection, this research has examined the usefulness of each Smartphones element and implemented a mobile application for transportation. For example, TrafficPulse utilizes different sensors to enable different features that the previous systems did not support. TrafficPulse also provides different visualization modules that the previous systems did not enable. Different from Waze, TrafficPulse collects wireless and offline GPS positioning locations. Moreover, the traffic data collected by TrafficPulse are available for research purposes.

Besides the mobile application, this research has also proposed a unique architecture for server to leverage the benefit of Cloud storage and to ensure high query performances. This part was investigated because there were no architectures for transportation systems available at the moment. TrafficPulse's font-end and backend system and the lessons learned can provide references to readers who want to make a similar GIS for transportation.

To examine the usefulness of the TrafficPulse system, a data collection campaign was held to recruit fifty students on University of Calgary campus to be TrafficPulse's users to collect

VGI. The results demonstrate that TrafficPulse is able to collect real-time traffic data, and the trace data contributed by the TrafficPulse users reflect most of the city's road network (Figure 2.8). After the campaign, we have drawn some conclusions, as well as, collected feedback from the users, and they are listed as followings.

First, it is necessary to collect offline GPS and wireless positioning locations because users who have Smartphones do not always have access to the Internet. However, these types of data provide information that cannot be captured by the "online" Smartphone users.

Second, it is recommended to include both an opportunistic sensing and a participatory sensing module when designing a mobile transportation system, because these two types of sensing data allow us to derive different kinds of information.

Third, it is also recommended to separate stale and raw data when designing a storage service for transportation applications. Shared traffic data become stale at some point, for example, most user requests are focused on near real-time traffic data. Since users are only interested in recent information, it is better to maintain a light weighted local data storage that stores fresh data and a key-value storage for stale data.

Lastly, it is important to design a system that is continuously improved based on user feedback. For example, we included a photo up-loader because the users found that photographic can be an important resource. Figure 6.1 shows a picture taken by a student who witnessed a car accident and wanted to provide a visual aid of the scene to other users in the system.

In the future, more privacy-preserving methods should be considered in order to have more users use the proposed GIS. For example, privacy issue for offline data should be carefully managed. In addition, it is necessary to provide animations to display the dynamics of traffic flow on clients, so that users can view not only near-by users but also their movements (e.g., directions).

In the second part of the thesis, in order to explore and extract local information from

Figure 6.1: Photo of a Car Accident Taken by a Student

social media, the GLC algorithm has been proposed to discover TLCs in social networks. Previous research focused on the discovery of emerging active users on the social networks and did not consider users' geo-information, such as geo-locations when designing their algorithms. In this way, communities discovered by the previous algorithms are not geographically bounded and users within groups are distributed sparsely. To discover geographically close communities, this research has enhanced the LC algorithm with the geo-location proximity.

Experimental results demonstrate that the GLC algorithm allows us to discover TLCs, and the TLCs enable us to identify more information that is relevant to local, (e.g., place abbreviations, local issues, and local influential users). However, limited by the data collected, this research could only study the city level TLCs. In fact, the GLC algorithm can be applied to different scales, for example, district level.

In the future, this algorithm can be applied to different data sets (e.g., mobile phone networks). The TLCs discovered at different scales (i.e., district level) can be studied. In addition, analyzing the lifetimes of the communities and the evolutions of topics in the com-

munities allow us to retrieve more information from the TLCs. It can also be an interesting research question.

In the third part of the thesis, in order to discover communities of interest (CoIs) in local social media, a framework has been proposed to discover local CoIs in the collected Twitter data. The proposed framework has examined the most suitable properties for detecting CoIs in the Twitter data and found that there are more than one property enabling the discovery of CoIs (i.e., the Mention and the URL prefix graph).

In addition, this research has proposed an algorithm that enhances the FGM algorithm to overcome the weakness of the FGM algorithm. The FGM algorithm is found to be too sensitive to graph structures; any change on the graph weights affects the clustering results. The proposed algorithm enhances the FGM algorithm with text similarity measures. The results show that the proposed algorithm is not as sensitive to graph structures as the generic algorithm. The text similarly measures act as a weak force to balance the importance of user-shared contents (e.g., short texts) and the graph structures when clustering graphs. As a result, the proposed algorithm is able to discover more CoIs than the generic algorithm. Also the proposed algorithm is reliable because the experiments provided compatible results under different scenarios.

Since the text similarly measure can be sensitive to the text inputs, this framework has employed NLP techniques to clean the short texts, normalize intentional and unintentional errors, and extract nouns, so that the extracted information will be more relevant to users' interests. Overall, the proposed framework successfully discovered a greater number of CoIs in local social media.

In the future, this framework can be applied to different social media, so that different sociograms and CoIs can be studied. In addition, different NLP techniques can be studied to help extract more relevant information from short texts. Moreover, the members in CoIs discovered from different sociograms might be different, and the integration of the sociograms

is also an interesting research question.

Bibliography

[1] Cycle watch – gps cycling computer for outdoor biking. https://itunes.apple.com/us/app/cycle-watch-gps-cycling-computer/id383531359?mt=8&ign-mpt=uo%3D4.

[2] Flickr. http://www.flickr.com/.

[3] Fluview. http://www.cdc.gov/flu/weekly/.

[4] Geonames. http://www.geonames.org/.

[5] Google earth. http://www.google.com/earth/index.html.

[6] Google maps for android. http://www.google.com/mobile/products/maps.html#p=default.

[7] Naheed nenshi and social media. http://www.thetelecomblog.com/2010/10/22/naheed-nenshi-and-social-media/.

[8] Open street map. http://www.openstreetmap.org/.

[9] The penn treebank project. http://www.cis.upenn.edu/~treebank/.

[10] Report: More than ever, traffic jams waste time. http://articles.cnn.com/2002-06-20/travel/traffic.jam_1_tim-lomax-rush-hour-driver-urban-mobility-report?_s=PM:TRAVEL.

[11] Social media: the right tool in right hands. http://bit.ly/fWCjgy.

[12] Ushahidi. http://www.ushahidi.com/.

[13] Waze. http://www.waze.com/.

[14] Wikipedia. http://www.wikipedia.org/.

[15] Wiktionary, the free dictionary. http://en.wiktionary.org/wiki/Main_page.

[16] Adamic, L. A., and Adar, E. How to search a social network. *Social networks 27*, 3 (2010).

[17] Alexander, W., Marion, C., Niina, H., Kevin, A., Sandi, L., and Der, Y. Towards a c2 poly-visualization tool: Leveraging the power of social-network analysis and gis. International Command and Control Research and Technology Symposium.

[18] Badrul, S., George, K., Joseph, K., and John, R. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (New York, NY, USA, 2001), WWW '01, ACM, pp. 285–295.

[19] Barna, S., and Pabitra, M. Dynamic algorithm for graph clustering using minimum cut tree. In *Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops* (Washington, DC, USA, 2006), ICDMW '06, IEEE Computer Society, pp. 667–671.

[20] Bell, M. The real time estimation of origin-destination flows in the presence of platoon dispersion. *Transportation Research Part B 25* (1991), 115–125.

[21] Blondel, V., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. 2008*, 1 (October 2008), 10008.

[22] Blondel, V. D., Krings, G. M., and Thomas, I. Regions and borders of mobile telephony in belgium and around brussels. *Brussels Studies 42* (2010).

[23] Boettcher, S., and Percus, A. G. Optimization with extremal dynamics. *Physical Review Letters 86* (2001), 5211–5214.

[24] Bollegala, D., Matsuo, Y., and Ishizuka, M. Measuring semantic similarity between words using web search engines. In *Proceedings of the 16th international conference on World Wide Web* (New York, NY, USA, 2007), WWW '07, ACM, pp. 757–766.

[25] Bonnington, C. Global smartphone adoption approaches 30 percent. `http://www.wired.com/gadgetlab/2011/11/smartphones-feature-phones/`.

[26] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst. 30*, 1-7 (1998), 107–117.

[27] Bruce, C., Donald, M., and Trevor, S. *Search Engines: Information Retrieval in Practice*, 1st ed. Addison-Wesley Publishing Company, USA, 2009.

[28] Campbell, J., and Shin, M. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[29] Cano, A., Varga, A., and Ciravegna, F. Volatile classification of point of interests based on social activity streams. In *Procedings of the 10th International Semantic Web Conference* (2011), vol. 27, pp. 23–27.

[30] Carpineto, C., Osiński, S., Romano, G., and Weiss, D. A survey of web clustering engines. *ACM Comput. Surv. 41*, 3 (2009), 1–38.

[31] Carter, H. Strategic planning reborn. *Work Study 48*, 2 (2004), 385–408.

[32] Chiti, F., and Fantacci, R. Urban microclimate and trafc monitoring with mobile wireless sensor networks. *InTechOpen* (2010).

[33] Christof, D., Tanja, H., and Dorothea, W. Fully-dynamic hierarchical graph clustering using cut trees. In *Proceedings of the 12th international conference on Algorithms and data structures* (Berlin, Heidelberg, 2011), WADS'11, Springer-Verlag, pp. 338–349.

[34] Clauset, A., and Newman, M. E. J. Finding community structure in very large networks. *Physical Review E 70*, 6 (2004).

[35] Cortes, C., Pregibon, D., and Volinsky, C. Communities of interest. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis* (London, UK, UK, 2001), IDA '01, Springer-Verlag, pp. 105–114.

[36] Cryans, J., April, A., and Abran, A. Criteria to compare cloud computing with current database technology. In *Proceedings of the International Conferences on Software Process and Product Measurement* (Munich, Germany, 2008), pp. 114–126.

[37] Damerau, F. J. A technique for computer detection and correction of spelling errors. *Commun. ACM 7*, 3 (1964), 171–176.

[38] Deterding, S., Sicart, M., Nacke, L., O'Hara, K., and Dixon, D. Gamification. using game-design elements in non-gaming contexts. In *Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems* (2011), CHIEA'11, pp. 2425–2428.

[39] Diestel, R. *Graph Theory*, 3st ed. Springer-Verlag Heidelberg, New York, USA, 2005.

[40] Donath, W. E., and Hoffman, A. J. Lower bounds for the partitioning of graphs. *IBM J. Res. Dev. 17*, 5 (1973).

[41] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern classification*, 2nd ed. New York: Wiley Interscience, 2000.

[42] Erkan, G., and Radev, D. R. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res. 22*, 1 (2004), 457–479.

[43] Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal 23*, 98 (1973), 298–305.

[44] Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. Graph clustering and minimum cut trees. *Internet Mathematics 1* (2004), 385–408.

[45] Fortunato, S. Community detection in graphs. *Physics Reports 486* (2010), 75–174.

[46] Freeman, L. C. A set of measures of centrality based on betweenness. *Sociometry 40* (1997), 35–41.

[47] Ganapati, S. Using geographic information systems to increase citizen engagement.

[48] G'omez, S., and Fern'andez, A. Multidendrograms: Variable-group agglomerative hierarchical clustering. *In Computer Research Repository (CoRR)* (2012).

[49] Gomory, R., and Hu, T. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics 9*, 4 (1961), 551–570.

[50] Goodchild, M. F. Citizens as sensors: The world of volunteered geography. 211–221.

[51] Group, I. C. U. E. R.

[52] Haewoon, K., Changhyun, L., Hosung, P., and Sue, M. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web* (New York, NY, USA, 2010), WWW'10, ACM, pp. 591–600.

[53] Hanneman, R. A., and Riddle, M. *Introduction to social network methods.* Riverside, CA: University of California, 2005.

[54] Hannon, J., Bennett, M., and Smyth, B. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems* (2010), RecSys '10, pp. 199–206.

[55] Haythornthwaite, C., and Gruzd, A. A noun phrase analysis tool for mining online community conversations. In *Proceedings of the Third Communities and Technologies Conference* (Michigan State University, 2007).

[56] Huang, A. Similarity measures for text document clustering. In *Proceedings of the New Zealand Computer Science Research Student Conference* (2008), pp. 49–56.

[57] Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, K., Miu, A., Shih, E., Balakrishnan, H., and Madden, S. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems* (2006), pp. 125–138.

[58] Jeh, G., and Widom, J. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2002), KDD '02, ACM, pp. 538–543.

[59] Kaisar, S. Smartphone traffic characteristics and context dependencies. Master's thesis, University of Saskatchewan, 2012.

[60] Kamath, K. Y., and Caverlee, J. Transient crowd discovery on the real-time social web. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining* (2011), WSDM'11, ACM, pp. 585–594.

[61] Katz, J. S. Co-link web indicators of the european research area. Tech. rep., Web Indicators For Scientific, Technological and Innovation, 2005.

[62] Kaufmann, M., and Kalita, J. Syntactic normalization of twitter messages. In *In Proceedings of the International Conference on Natural Language Processing* (2010), ICON'10.

[63] Kernighan, B. W., and Lin, S. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal 49*, 1 (1970), 291–307.

[64] Kernighan, M. D., Church, K. W., and Gale, W. A. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics* (1990), COLING '90, Association for Computational Linguistics, pp. 205–210.

[65] Kirkpatrick, S., Gelattand, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.

[66] Kwak, H., Lee, C., Park, H., and Moon, S. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web* (2010), WWW '10, pp. 591–600.

[67] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (San Francisco, CA, USA, 2001), ICML '01, Morgan Kaufmann Publishers Inc., pp. 282–289.

[68] Lane, N., Eisenman, S. B., Musolesi, M., Miluzzo, E., and Campbell, A. T. Urban sensing systems: opportunistic or participatory? In *Proceedings of the 9th workshop on Mobile computing systems and applications* (New York, NY, USA, 2008), HotMobile '08, ACM.

[69] Lee, D. L., Chuang, H., and Seamons, K. Document ranking and the vector-space model. *IEEE Software 14*, 2 (Mar. 1997), 67–75.

[70] Lee, D. W., and Liang, S. H. L. Geopot: a cloud-based geolocation data service for mobile applications. *International Journal of Geographical Information Science 25*, 8 (2011), 1283–1301.

[71] Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady 10* (1966), 707–710.

[72] Li, X., Hu, W., and Hu, W. A coarse-to-fine strategy for vehicle motion trajectory clustering. In *Proceedings of the International Conference on Pattern Recognition (ICPR)* (2006), pp. 591–594.

[73] Lieberman, M. D. You are where you edit: locating wikipedia users through edit histories. In *Proceedings of the Third International Conference on Weblogs and Social Media* (New York, NY, USA, 2005), ICWSM'09, ACM, pp. 187–203.

[74] MacQueen, J. B. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), vol. 1, University of California Press, pp. 281–297.

[75] Marlow, C., Byron, L., Lento, T., and Rosenn, I. Maintained relationships on facebook.

[76] Martin, E., Hans-Peter, K., Jörg, S., and Xiaowei, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Information Retrieval* (1996), pp. 226–231.

[77] Mays, E., Damerau, F. J., and Mercer, R. L. Context based spelling correction. *Inf. Process. Manage. 27*, 5 (1991), 517–522.

[78] Mengen, C., Xiaoming, J., and Dou, S. Short text classification improved by learning multi-granularity topics. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three* (2011), IJCAI'11, AAAI Press, pp. 1776–1781.

[79] Mihalcea, R., Corley, C., and Strapparava, C. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence* (2006), vol. 1 of *AAAI'06*, AAAI Press, pp. 775–780.

[80] Mohan, P., Padmanabhan, V. N., and Ramjee, R. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems* (2008), pp. 323–336.

[81] Newman, M. E. J. Detecting community structure in networks. *The European Physical Journal B (EPJ B) 38* (2004), 321–330.

[82] Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E 74* (2006), 1–22.

[83] Newman, M. E. J., and Girvan, M. Finding and evaluating community structure in networks. *Physical Review E 69* (2004).

[84] Okazaki, N., Matsuo, Y., and Matsumura, N. Sentence extraction by spreading activation through sentence similarity. *IEICE Transactions on Information and Systems E86-D* (2003), 1686–1694.

[85] Oreilly, T. What is web 2.0: Design patterns and business models for the next generation of software. *Communications and Strategies 1* (2007), 17–37.

[86] Paolo, B., Massimo, S., and Sebastiano, V. Pagerank as a function of the damping factor. In *Proceedings of the 14th international conference on World Wide Web* (New York, NY, USA, 2005), WWW '05, ACM, pp. 557–566.

[87] Parker, C., May, A., and Mitchell, V. Relevance of volunteered geographic information in a real world context. In *Proceedings of GIS Research UK 19th Annual Conference* (Portsmouth, UK, 2011).

[88] Porter, M. E. Location, competition, and economic development: Local clusters in a global economy. *Economic Development Quarterly* (2000), 15–34.

[89] Qian, X., Di, L., Li, D., Li, P., Shi, L., and Cai, L. Data cleaning approaches in web2.0 vgi application. In *Proceedings of the 17th International Conference on Geomatics* (Fairfax, USA, 2009), pp. 12–14.

[90] Ravi, K., Jasmine, N., and Andrew, T. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2006), KDD '06, ACM, pp. 611–617.

[91] Sahami, M., and Heilman, T. D. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web* (New York, NY, USA, 2006), WWW '06, ACM, pp. 377–386.

[92] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing And Management 24*, 5 (1988), 513—523.

[93] Shaw, S. L. Geographic information systems for transportation: from a static past to a dynamic future. *Annals of GIS 16*, 3 (2010), 129–140.

[94] Stefanidis, A., Crooks, A., and Radzikowski, J. Harvesting ambient geospatial information from social media feeds. *GeoJournal* (2011), 1737–1748.

[95] Suia, D., and Goodchild, M. The convergence of gis and social media: challenges for giscience. *International Journal of Geographical Information Science 25*, 11 (2011), 1737–1748.

[96] Takeru, M., Saeko, N., and Toru, I. Semantic web link analysis to discover social relationships in academic communities. In *Proceedings of the The 2005 Symposium on Applications and the Internet* (Washington, DC, USA, 2005), SAINT '05, IEEE Computer Society, pp. 38–45.

[97] Takeshi, S., Makoto, O., and Yutaka, M. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 851–860.

[98] Tobler, W. A computer movie simulating urban growth in the detroit region. *Economic Geography 46*, 2 (1970), 234–240.

[99] Turner, A. *Introduction to Neogeography*. O'Reilly Media, December 2006.

[100] White, H. D., and McCain, K. W. Visualizing a discipline: an author co-citation analysis of information science, 1972-1995. *J. Am. Soc. Inf. Sci. 49*, 4 (Apr. 1998), 327–355.

[101] Wu, S., Hofman, J. M., Mason, W. A., and Watts, D. J. Who says what to whom on twitter. In *Proceedings of the 20th international conference on World wide web* (2011), WWW '11, pp. 705–714.

[102] Yang, B., and Liu, J. Discovering global network communities based on local centralities. *ACM Trans. Web 2*, 1 (2008).

[103] Yates, R. B., Boldi, P., and Castillo, C. The choice of a damping function for propagating importance in link-based ranking. Tech. rep., Dipartimento di Scienze dell'Informazione, Universita degli Studi di Milano., 2005.

[104] Zamir, O., and Etzioni, O. Web document clustering: a feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 46–54.

[105] Zhang, L., Li, C., Liu, J., and Wang, H. Graph-based text similarity measurement by exploiting wikipedia as background knowledge. *World Academy of Science,*

Engineering and Technology 59 (2010), 1548–1553.

[106] Zhi-Dan, Z., and Ming-sheng, S. User-based collaborative-filtering recommendation algorithms on hadoop. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining* (Washington, DC, USA, 2010), WKDD '10, pp. 478–481.

[107] Zhiyuan, C., James, C., and Kyumin, L. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (New York, NY, USA, 2010), CIKM '10, ACM, pp. 759–768.