



HAL
open science

Towards a Framework for Multiscale Social Event Extraction and Visualization

Faizan Ur Rehman

► **To cite this version:**

Faizan Ur Rehman. Towards a Framework for Multiscale Social Event Extraction and Visualization. Computers and Society [cs.CY]. Université Grenoble Alpes, 2018. English. NNT : 2018GREAM073 . tel-02091712

HAL Id: tel-02091712

<https://theses.hal.science/tel-02091712>

Submitted on 6 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA

COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Faizan Ur REHMAN

Thèse dirigée par **Ahmed LBATH**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG)**
dans l'**École Doctorale Mathématiques, Sciences et Technologies de**
l'Information, Informatique (EDMSTII)

Vers une plateforme pour l'extraction et la
visualisation multi-échelle d'événements sociaux

Towards a Framework for Multiscale Social Event
Extraction and Visualization

Thèse soutenue publiquement le **7 December 2018**,
devant le jury composé de:

M. Arif GHAFOR

Professor, Purdue University, Rapporteur

M. Kokou YETONGNON

Professeur, Université de Bourgogne, Rapporteur

M. Abd-El-Kader SAHRAOUI

Professeur, Université de Toulouse, Président

M. AHMED LBATH

Professeur, Université Grenoble Alpes, Directeur de thèse

M. Saleh BASALAMAH

Associate Professor, Umm Al-Qura University, Examineur (co-encadrant)

M. Imad AFYOUNI

Assistant Professor, University of Sharjah, Examineur (co-encadrant)





Résumé

La population des villes devrait doubler d'ici le milieu du siècle, selon les estimations de l'Organisation mondiale de la santé. Cette augmentation rapide de la population aura un impact sur les transports et la croissance économique, et accroîtra les responsabilités des autorités de gestion locales et des différentes parties prenantes. Nous vivons une transformation des villes en villes intelligentes permettant d'offrir de nouveaux services au public, en optimisant l'utilisation des ressources disponibles. Qu'il s'agisse de données provenant des citoyens (utilisateurs finaux), de données gouvernementales ouvertes ou d'autres sources en ligne, une pluralité de sources de données peut permettre la fourniture, à tous les utilisateurs, des outils intelligents pour gérer efficacement leurs activités quotidiennes. De plus, avec les progrès de l'Internet et des technologies mobiles, les plateformes des réseaux sociaux tels que Facebook et Twitter sont devenues des modes de communication populaires. Ils permettent aux utilisateurs de partager en temps réel un large éventail d'informations, y compris des données spatio-temporelles, à la fois publiquement et au sein de leur communauté d'intérêts. Il est devenu plus facile d'examiner les connaissances provenant de différents types de données disponibles, riches, géo-référencées et provenant de sources multiples et de les intégrer sur une carte. Il s'agit d'une réelle opportunité d'enrichir les cartes traditionnelles et d'enrichir les interrogations spatio-temporelles conventionnelles à l'aide de différents types de données extraites d'une pluralité de sources de données disponibles. Dans cette thèse, nous proposons d'abord un système de recommandation d'itinéraires tenant compte des contraintes en l'absence d'infrastructure physique qui exploite les données géolocalisées issues de réseaux sociaux (comme twitter) et le contenu généré par les utilisateurs pour identifier les contraintes de trafic à venir et, par conséquent, recommander un chemin optimisé. Nous avons mis en œuvre un nouveau système à l'aide d'indexation à base de grille spatiale afin d'informer les utilisateurs des contraintes à venir et calculer un nouveau chemin optimisé en un temps de réponse minimal. Ensuite, nous avons introduit le concept de "cartes intelligentes" intégrant la représentation visuelle de couches de « connaissances pertinentes » par le biais de la collecte, la gestion et l'intégration de sources de données hétérogènes. Contrairement aux cartes conventionnelles, les cartes intelligentes extraient des informations à partir des événements annoncés et découverts en temps réel (p. ex., concerts, concours, incidents, etc.), les offres en ligne et les analyses statistiques (p. ex., zones dangereuses) en encapsulant les données entrantes semi-structurées et non structurées dans des paquets génériques structurés.

Cette méthodologie ouvre la voie à la fourniture de différents services et applications intelligents. De plus, le développement de « cartes intelligentes » nécessite un traitement efficace et évolutif et la visualisation de couches basées sur les connaissances à plusieurs échelles cartographiques, permettant ainsi une navigation fluide et sans encombre. Enfin, nous présentons Hadath, un système évolutif et efficace qui extrait les événements sociaux d'une multitude de flux de données non structurés. Hadath applique le traitement du langage naturel et les techniques de regroupement multidimensionnel pour extraire les « événements pertinents » à différentes échelles cartographiques et pour déduire l'étendue spatio-temporelle des événements détectés.

Le système comprend un composant de gestion des données qui effectue un prétraitement des différents types de sources de données et génère des paquets de données structurés à partir de flux non structurés. Notre système, nommé Hadath comprend également un schéma d'indexation spatio-temporelle hiérarchique en mémoire pour permettre un accès efficace et

évolutif aux données brutes, ainsi qu'aux groupes d'événements extraits. Dans un premier temps, les paquets de données sont traités pour la découverte des événements à l'échelle locale, puis l'étendue spatio-temporelle appropriée. Par conséquent, les événements détectés peuvent être affichés à différentes résolutions spatio-temporelles, ce qui permet une navigation fluide et unique. Enfin, pour valider le système proposé, nous avons mené des expériences sur des flux de données réelles. Le résultat final du système proposé, nommé Hadath crée une expérience unique et dynamique de navigation cartographique.

Mots-clés

Villes intelligentes, Crowdsourcing, Cartes enrichies d'événements, Portée spatio-temporelle, Visualisation d'événements, réseau sociaux (capteurs)

Abstract

The population in cities is slated to double by mid-century according to estimates prepared by the World Health Organization. This rapid increase in population will impact transportation and economic growth, and will increase responsibilities of local managing authorities and different stakeholders. It is a need of the hour to convert cities into smart cities in order to provide new service to the public, by using available resources in an optimum manner. From crowd-sourced data and open governmental data to other online sources, a variety of data sources can provide users with smart tools to efficiently manage their daily activities. Moreover, with the advancement in Internet and mobile technologies, social networking platforms such as Facebook and Twitter have become popular modes of communication. They allow users to share a spectrum of information, including spatio-temporal data, both publicly and within their community of interest in real-time. Scrutinizing knowledge from different types of available, rich, geo-tagged, and crowd-sourced data and incorporating it on a map has become more feasible. This presents a real opportunity to enrich traditional maps and enhance conventional spatio-temporal queries with the help of different types of data extracted from a variety of available data sources. In this thesis, we first propose a constraint-aware route recommendation system in lack of physical infrastructure environment that leverages geo-tagged data in social media and user-generated content to identify upcoming traffic constraints and, thus, recommend an optimized path. We have designed and developed a system using a spatial grid index to inform users about upcoming constraints and calculate a new, optimized path in minimal response time. Later, the concept of “smart maps” will be introduced by collecting, managing, and integrating heterogeneous data sources in order to infer relevant knowledge-based layers. Unlike conventional maps, smart maps extract information about live events (e.g., concert, competition, incidents, etc.), online offers, and statistical analysis (e.g., dangerous areas) by encapsulating incoming semi- and un-structured data into structured generic packets. This methodology sets the ground for providing different intelligent services and applications: 1) city explorer that provides the latest information collected from multiple sources about places and events; 2) route and trip planning to leverage the smart map framework to recommend safe and optimized routes; and 3) multimedia routing mobile applications that enhance the routing framework by leveraging geo-tagged multimedia data such as images, audio, video, and text in order to add semantics to conventional spatial queries. Moreover, developing smart maps requires an efficient and scalable processing and the visualization of knowledge-based layers at multiple map scales, thus allowing a smooth and clutter-free browsing experience. Finally, we introduce Hadath, a scalable and efficient system that extracts social events from a variety of unstructured data streams. Hadath applies natural language processing and multi-dimensional clustering techniques to extract relevant events of interest at different map scales, and to infer the spatio-temporal extent of detected events. The system comprises a data wrapping component which digests different types of data sources, and preprocesses data to generate structured data packets out of unstructured streams. Hadath also implements a hierarchical in-memory spatio-temporal indexing scheme to allow efficient and scalable access to raw data, as well as to extracted clusters of events. Initially, data packets are processed to discover events at a local scale, then, the proper spatio-temporal extent and the significance of detected events at a global scale is determined. As a result, live events can be displayed at different spatio-temporal resolutions, thus allowing a smooth and unique browsing experience. Finally, to validate our proposed system, we conducted experiments on real-world data streams.

The final output of our system named Hadath creates a unique and dynamic map browsing experience.

Keywords

Smart Cities, Crowdsourcing, Event-Enriched Maps, Spatio-Temporal Scope, Visualization, Social Sensors

.

Acknowledgments

My happiness knows no bound in expressing my cordial gratitude to all the kind people with support and help of whom this thesis has become possible.

Foremost, I want to offer this endeavour to the Almighty, for the wisdom, strength, peace of mind and health he bestowed upon me in order to finish this research.

I am highly indebted to the University of Grenoble and University of Umm Al Qura, for their constant support in providing necessary information regarding this research and for completing this endeavour.

I would like to express my deepest appreciation to my supervisor, *Prof Ahmad Lbath*, for imparting his knowledge and expertise in this study. He has an attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in this research. Without his guidance and support this research would not have been possible.

I would like to express my sincere thanks to my co-supervisor, *Prof Salah Basalamah*. His good advices, supports and friendship has been invaluable on both academic and personal levels, for which I will be forever grateful.

My sincere thanks to *Dr Imad Afyouni*, who is like a brother to me. I am highly indebted that I met him and got a chance to work with him. My skills and efficiencies as a researcher has developed tremendously under his company.

My special thanks to *Dr Mohammed Abdur Rahman*, *Prof Mohammad Mokbel*, *Dr Sohaib Khan*, *Prof Walid Aref*, for providing me the opportunity to work with them. I am very thankful for their trust in me and the opportunities they gave me to grow not just as researcher but also as a good person.

In addition to my supervisors, I humbly extend my thanks for the support of my colleagues and friends *Ahmad Muaz Qamar*, *Akhlaq Ahmad*, and *Bilal Sadiq*. We worked as a team in research field and were helping hands to each other whenever required.

I would also like to thank the members of my Ph.D. committee, Professor *Abd-El-Kader Sahraoui*, Professor *Arif Ghafoor*, Professor *Kokou Yetognon* for their productive feedback. They have put tremendous efforts to review the thesis which is without any doubt not an easy task. Thanks to committee members for all the efforts.

Finally, my sincerest and deepest gratitude is towards my parents, and family members, who has always seen the best in me and has been there besides me through thick and thin. My mere expression of thanks for their unequivocal personal support, as always, does not suffice. My parents are my pillar since my childhood. They always asked me to gain knowledge not only for my betterment but for the betterment of the society. My wife *Uzra Khan* is a constant support for me on personal and professional end and was always there during the challenges that I faced

during this time span. Last but not least special thanks to my children *Yahya Rehman* and *Yaqub Rehman*.

Contents

Abstract v

Acknowledgments..... vii

Contents..... ix

List of Tables..... xii

List of Figures xiii

List of Algorithms..... xvi

Abbreviations..... xvii

Chapter 1 Introduction 1

1.1 Motivation and Challenges..... 2

1.2 Problem Statements 5

1.3 Contributions 6

1.4 Outline 8

Chapter 2 Literature Review 10

2.1 Introduction 10

2.2 Existing Map System and Mapping Technologies..... 11

2.3 Data Collection towards Smart Cities 12

2.4 Event Detection along with spatio-temporal scope..... 13

2.5 Spatial Clustering techniques 15

2.6 Existing System-Performance and Scalability Perspectives 17

2.7 Shortest Path and routing 18

 2.7.1 Single Source Shortest Path 19

 2.7.2 Multimedia Routing 20

 2.7.3 Safe Routing..... 21

2.8 Conclusion..... 22

Chapter 3 Constraint-Aware Optimized Path in Lack of Physical Infrastructure Environment 23

3.1 Introduction 23

3.2 Services and Routing Challenges 26

 3.2.1 Location-based Services..... 27

 3.2.2 Routing based services 28

 3.2.3 Traffic Updates Data Collection Service and Geotagged Crawler..... 28

3.2.4	Routing Challenges in large crowd	29
3.3	Constraint-Aware Optimized Path Algorithm.....	30
3.3.1	Modelling.....	30
3.3.2	Algorithm.....	31
3.4	High-Level Architecture	33
3.5	Implementation and Test Results	34
3.5.1	Implementation.....	34
3.5.2	Evaluations and Routing Algorithm Output	37
3.6	Conclusion and Future Work	39
Chapter 4 Building Smart Maps from Heterogeneous Data Sources		40
4.1	Introduction	40
4.2	High-level architecture	42
4.2.1	Data Collection and Management.....	43
4.2.2	Preprocessing	44
4.2.3	Indexing	45
4.2.4	Knowledge-based Layers.....	45
4.2.5	Knowledge Extraction	45
4.2.6	Smart Map Layer Building	46
4.2.7	Generating and Visualizing Smart Map Layers.....	47
4.3	Implementation.....	48
4.4	Application on smart maps.....	49
4.4.1	City explorer	49
4.4.2	Safe Routing.....	50
4.4.3	Multimedia Routing - Mobile App	56
4.4.3.1	Architecture.....	57
4.4.3.2	Implementation	59
4.4.3.3	Application Functionality	59
4.5	Conclusion.....	63
Chapter 5 Towards a Framework for Scalable Multi-Resolution Event Enriched Maps		64
5.1	Introduction	64
5.2	Motivation and Challenges.....	66
5.3	Event-Enriched Maps	68
5.4	System Design.....	70
5.5	Data Cleaner and Wrapper	72
5.6	Data Manager	77

5.7	Event of Interest Detection.....	80
5.8	Spatial Scope Finder.....	84
5.9	Graphical Interface for Exploring Maps	91
5.9.1	Overview of the System.....	91
5.9.2	Database Schemna	93
5.9.3	Use Cases	95
5.10	Conclusion.....	100
Chapter 6 HADATH: System Implementation and Results		101
6.1	Introduction	101
6.2	Implementation.....	101
6.3	Results	108
6.4	Evaluation.....	113
6.5	Conclusion.....	116
Chapter 7 Conclusion and Future Research.....		117
7.1	Conclusion.....	117
7.2	Future Research.....	118
Appendix A Multimedia Routing.....		120
Appendix B Hadath.....		124
List of Publications.....		144
	Journal Papers	144
	Conference Papers.....	144
Bibliography		147

List of Tables

Table 3-1: Number of Pilgrims, Vehicles used and pilgrims per vehicle for the last three years of Hajj.....	24
Table 3-2: Server Implementation Details	35
Table 4-1: Relation between Keyword of User’s Query, EoIs and Sources of Data Collection	51
Table 5-1: NLP for a given text with type and confidence score.....	76
Table 6-1: Some potential EoIs that are detected from window-based bulk of tweets of 19-June-2017 (11:48 am to 12:14 pm)	109
Table 6-2: Indexing time for 15,000 packets	114

List of Figures

Figure 3-1: Percentage of different services used by pilgrims in Hajj 2014 (cf. ‘Perform Hajj and Umrah Application’ ¹⁵).	25
Figure 3-2: Constraint collection framework.	26
Figure 3-3: Main landing page of the new Hajj and Umrah Application (2015) with the list of Services	26
Figure 3-4: Incentive based traffic update module (a) Accident with image and comments (b) List of other options in the drop down	27
Figure 3-5: Visual Interface of the tweets related to traffic constraints	28
Figure 3-6 Comparison of Navigation issue based on A) Open Street Map Data in our Hajj and Umrah Application (2014) and B) Google Maps	29
Figure 3-7: Show cells and boundary points after the initial step of the algorithm	31
Figure 3-8: High level system architecture	33
Figure 3-9: JSON/XML based serialization /de-serialization	35
Figure 3-10: Response time graph for load testing.	36
Figure 3-11: Algorithm output for a) finding points of interest and b) nearby mosques	39
Figure 4-1: Dynamic layers of smart maps	41
Figure 4-2: Architecture of the smart maps framework	43
Figure 4-3: Sample Data Packet from Twitter Streams	44
Figure 4-4: PostGIS screenshot to demonstrate data integration of POIs from Yelp ID and Twitter Screen Name	44
Figure 4-5: Overview of Implemented Smart Maps with Crime and Accident Roads	47
Figure 4-6: Overview of Implemented Smart Maps with POI Semantics	48
Figure 4-7: Enhanced city explorer on smart maps: Different type of EoIs on 3rd June: sports, graduation party, road incident, jobs hiring (from top left, top right, bottom left and bottom right)	50
Figure 4-8: A Map with Crime (pink color) and Accident Prone (blue color) Edges along with two Red Markers to Denote Starting and End Point	52
Figure 4-9: An optimized path with respect to time between two markers (showing with red color) without considering crime and accident prone edges	53
Figure 4-10: A crime free path by avoiding crime prone edges (light green color) without considering accident prone edge.	54
Figure 4-11: Safest path calculated by avoiding both accident prone and crime prone edges (showing with blue color).	55

Figure 4-12: Multimedia geotagged data collection through our developed mobile application. A) Multimedia-enhanced traffic update; and B) Adding multimedia-enhanced points of interest	57
Figure 4-13: High-level architecture	58
Figure 4-14: A) Select multimedia sources, POIs and date, B) Multimedia-enhanced routing	60
Figure 4-15: A) Conventional shortest path query and B) Multimedia-enhanced routing query with images and text related to supermarket and restaurant fetched from twitter and our system from 25-May-2015 onwards.	60
Figure 4-16: K-nearest multimedia-enhanced neighbor query	61
Figure 4-17: Multimedia finding lost individuals	61
Figure 4-18: A) Conventional K_{nn} Query, B) Live traffic constraints for nearby area query and C) K-nearest multimedia-enhanced neighbor query.	62
Figure 4-19: A) Conventional range query, B) Live traffic constraints for nearby area query and C) Multimedia enhanced range query.	62
Figure 4-20: Pseudocode of the algorithm for extracting location from geo-tagged multimedia messages and generating multimedia-enhanced route.	63
Figure 5-1: Conceptual illustration of Event-Enriched Maps: Findings automatically discovered from live streams, such as a restaurant opening, a neighborhood party, an accident prone road segment, warnings on demonstrations and emergency cases	65
Figure 5-2: Hadath System Architecture.....	71
Figure 5-3: Wrapping multiple sources of data into a generic composite event packet	73
Figure 5-4: Sample Data Packet from Twitter and Flickr Streams	77
Figure 5-5: A snapshot of indexed data packets at a fine level of the hierarchical tree.....	78
Figure 5-6: Multi-level spatio-temporal index	79
Figure 5-7: Local EOI Detection within the leaf cells A) List of existing clusters and packets, B) shows arrival of new indexed data packets, C) Merging new packets in existing cluster, forming new cluster, and cleaning old packets and clusters based on threshold value, D) resultant leaf cells after all operation mentioned in Algorithm 2.....	83
Figure 5-8: Visualization of EoI without scope	84
Figure 5-9: Spatial Scope. A) Horizontal and B) Vertical	85
Figure 5-10: Sample horizontal spatial scope of events from Depth 'X' to 'X-4'.....	90
Figure 5-11: Overview of the Query Interface System	92
Figure 5-12: Database Schema.....	94
Figure 5-13: Query Interface Overview	95
Figure 5-14: List of some EoIs with an option to write in inbox in case of other type of EoIs	96
Figure 5-15: Event scope to select importance of different level with an option to select “Less than”, “More than”, and “Equal”	97
Figure 5-16: Smart Routing Query Interface	98

Figure 5-17: Smart City Explorer Query Interface	98
Figure 5-18: Emergency Evacuation Query Interface.....	99
Figure 5-19: Smart Trip Planner Query Interface	99
Figure 6-1: Shows view of aggregated geo-clusters with 3 character ‘DR5’ and 5 character ‘DR5R’ of indexed tree	102
Figure 6-2: Map view of aggregated geo-cluster with location string of 2 character and precise location of 1,250km * 625km.	103
Figure 6-3: Map view of aggregated geo-cluster with location string of 4 character and precise location of 39.1 km * 19.5 km.	104
Figure 6-4: Map view of aggregated geo-cluster with location string of 6 character and precise location of 1.22km * 0.61km.	105
Figure 6-5: Size of Cells at Different Level of Spatially Indexed Tree	106
Figure 6-6: List of Unspecified Words with Number of Occurrence in One Batch of Tweets	110
Figure 6-7: Query and a sample EoI of zoom level 17	110
Figure 6-8: Interactive Tag/Word Cloud of EoIs	111
Figure 6-9: High Abstraction Map View with Country Level Social Events	112
Figure 6-10: Detailed Map View for the City of New York	113
Figure 6-11: Number of potential indexed packet at precision 5 level	113
Figure 6-12: Processing time for generating data packets from 100,000 tweets	114
Figure 6-13: Processing time for event discovery at the different map zoom levels	116

List of Algorithms

Algorithm 3-1: Recommending optimize path to the affected User.....	31
Algorithm 5-1: Data Wrapper and Cleaner.....	75
Algorithm 5-2: Local EoI detection by clustering of event packets within leaf cells.....	82
Algorithm 5-3: Efficient hierarchical clustering to determine spatial scope.....	88

Abbreviations

AF	A ccident F ree
API	A pplication P rogramming I nterface
APSP	A ll P air S hortest P ath
CF	C rime F ree
COI	C ommunities o f I nterest
DBSCAN	D ensity-based S patial C lustering of A pplications with N oise
EoI	E vents o f I nterest
IDF	I nverted D ocument F requency
KDE	K ernel D ensity E stimation
LSH	L ocation S ensitivity H ashing
MM Routing	M ultimedia R outing
NLP	N atural L anguage P rocessing
PoI	P oints o f I nterest
SP	S hortest P ath
SSSP	S ingle S ource S hortest P ath
ST Scope	S patio-temporal S cope
TF	T erm F requency
UGC	U sers G enerated C ontents

Chapter 1

Introduction

Social networking has allowed users to write, read, and comment on social posts. Many users prefer using social networking platforms when in transit. Social networking sites such as Facebook and Twitter are the world's most visited sites. End users, industry experts, and researchers have shown considerable interest in social sensors because they provide a rich source of data. Thanks to smartphones equipped with high precision GPS sensors and high speed internet access, most content shared on social networking platforms is geo-tagged. For example, a large number of the millions of tweets posted every second are geo-tagged. This geo-tagged data gives the location and time from where people are sharing their content: this provides additional knowledge to enhance existing maps and traditional spatial queries.

Mapmaking has been essential to human civilisation since its beginning. The Ptolemaic maps were the first to refer to locations on the earth with latitude and longitude: since then, many maps have been introduced with improved cartography, including Fra Mauro Map (around 1450 AD), Mercator map (1569 AD), and Ricci Map (1602 AD). They have been used mostly for administrative and military necessities [1]. During late 19th and 20th centuries, growth in railroads made travelling quicker and inexpensive and encouraged cartographers to give greater attention to paper maps. Subsequently, technological innovations such as plotters, scanners, databases, image processing, and spatial analysis made it possible for maps to become more detailed. The advent of the twenty-first century has seen a tremendous revolution in the age old practice of cartography. With widespread access to tablets and smart phones, and ease of availability of positioning technologies like GPS, the frequency of usage of maps—particularly for navigation—is unprecedented in today's world. Maps today include themes: for example, road networks along with some meta-data on prominent places and points of interest at different scales have been incrementally added to maps. This new generation of maps includes digital maps, semantically-annotated maps, and, more recently, maps with live streams that are dynamically disseminated in space and time. The use of digital maps has increased with the overall aim of sharing preeminent information about current locations and spatial characteristics. Today, researchers, authorities, and industries generate thousands of map-based analytics to satisfy their social and economic needs [2]. Moreover, big giants—including

Google, Here, Yahoo, and Bing—provide dynamic layers about traffic updates such as jams, accidents, and congestions.

A variety of data are available in smart cities and big data era namely online social media, crowd-sourced data, open governmental data, and other online news sources. These huge heterogeneously available data can be utilized for extracting knowledge beneficial for end users. Nowadays, it is a general practice to use digital mapping applications which are limited to live data in finding directions, the status of traffic, or places of interest. There is, thus, a tremendous opportunity to enrich current maps with knowledge extracted from available heterogeneous sources. The goal of this thesis is to propose an efficient and scalable framework that is capable of information retrieval, data management, and sentiment analysis techniques to extract knowledge from available sources and disseminate with multi-resolution visualization. For example, a spike in tweets talking about an accident in a particular locality followed by rapid dissemination of related tweets or other sources may well indicate that a celebrity or a public figure is involved. We, therefore, design new models and algorithms by handling crowdsourced data for efficient extraction, clustering, spatial scope and mapping of live events by using an in-memory spatial indexing scheme. To validate our proposed framework, we developed a system with real-world data. Moreover this thesis also provides a framework for delivering different intelligent services and applications, such as city explorer, trip planner and optimized routing with safest, scenic, multimedia and constraint-aware parameters.

In this chapter, the main issues of this thesis are introduced. The description of the motivation and challenges of this study are illustrated in Section 1.1. Section 1.2 presents the problem statement and then the objectives and contributions are detailed in Section 1.3. Finally, the outline of this thesis is listed in Section 1.4.

1.1 Motivation and Challenges

Analyzing crowdsourced data can provide deep insights about live surrounding events or any unusual happenings, given plenty of relevant spatio-temporal information is embedded in social media streams. Social events of interest usually include gatherings, concerts, incidents, job announcements, or natural disasters, among others. Detecting or predicting such events in real-time can leverage new ways for exploring cities with dynamic content generated by the live communities in the surroundings, thus helping decision makers and authorities in providing context-aware intelligent services to their audience.

Existing mapping systems are limited in the sense that they are able to offer navigational aids based only on live traffic conditions or by visualizing streams from social media sources. Social media streams, however, contain considerable noisy and irrelevant data. It is necessary to filter these streams so as to segregate irrelevant content in an aggregated and non-cluttered manner. Our vision for the next generation of maps is premised on the richness of relevant spatio-temporal information embedded in social media streams. Users' map browsing experience can be enriched significantly if intelligent maps can discover relevant information from these unstructured data streams.

Let us illustrate this with a hypothetical case study, of a small family visiting Paris for the first time. This family is interested in planning the best weekend trip, to visit all the top attractions and eat the best food. They are interested in historic places with entry discounts, museums,

musical concerts, good Italian and/or Japanese food with family promotions, and other kid- and family- friendly activities. An optimum trip to reach all these attractions in the least possible time will require gathering live events and announcements for that particular weekend. The most convenient and time effective routes they must take will have to take into account the opening hours of their various POIs and all available transportation facilities along with their reviews and ratings. Current mapping systems struggle to accomplish this task. A native approach, for instance, will need an expert user to look into traditional search engines to find relevant events and reviews, and then manually best-match all the many keeping in mind the strict time constraint.

Fundamentally, *enriched maps* are about answering the *what-is-happening-around?* question. Unlike traditional systems where spatio-temporal information is inferred on an on-demand basis, a mapping system that can extract and update dynamic events on the move and define their spatio-temporal scope appropriately is better suited as a search engine for cities. Events of interest can be identified at different scales in urban areas, such as musical events, concerts, jobs, weather updates, traffic conditions, marches and protests, crimes, robberies, disasters, etc. Furthermore, new maps can be used to enhance existing routing and spatio-temporal queries. However, collecting and integrating this mass of overlapping, complementary, and sometimes contradictory data in order to extract relevant events and inferring their spatial and temporal scope so that they can be displayed in a clear, non-cluttered manner remains a challenge.

There are several challenges related to data collection, data integration and knowledge extraction, and others with respect to the efficiency and scalability to build a real-time system. We present the main challenges of the thesis as follows:

Challenge 1: Collecting Data and Recommending Optimized Path in a poorly Infrastructured Environment

Recommending a constraint-aware optimized path for a large crowd poses a unique challenge to existing routing algorithms due to the interactions between users and the dynamic changes over road networks. In cases of lack of physical infrastructure such as inductive loops, cameras, and drones, this approach has become more difficult. In this regard, we need a framework to collect data through a) available geotagged social media data and b) by motivating users to use location-based services. From this data, dynamic road constraints can be extracted such as accidents, constructions, road closed, which can be utilized to inform users, which might get affected, in a minimal response time. Understanding the users need and recommending them a constraint-aware routing service to enhance their daily activities in a smart manner is the goal of this framework.

Challenge 2: Dealing with Heterogeneous Data Sources

Information about potential events is scattered among the different data sources, such as Twitter, Instagram or Flickr. One key challenge is how to engineer a system to exploit disparate sources with diverse unstructured content, varying input rates and volumes. Extracting *live events* dynamically from a variety of data sources requires converting unstructured data streams into a common structured format in real-time. Hence, there is a need to have a module that can package unstructured content into a single data composite, which can be mined for inferring relevant event content by higher layers of the system. Adding a new data source to the system

requires defining the different package items and its meta-data context, without affecting any other component of the system.

Challenge 3: Discovering Events of Interest

Detecting an event requires building a classification model that indicates whether a given packet represents a potential unusual happening, and if so, labeling that packet with the type or category as provided in the trained data sets. The input raw features that should be provided to the classification model are obtained from the training data on existing built corpora. Feature engineering includes n-gram, TF-IDF, or vector embedding in order to represent the textual data. Moreover, events can be discovered more efficiently on small-scale regions based on nearby locations and text similarity, but it becomes more challenging to efficiently merge similar events as we zoom out of the map. For instance, events of someone's birthday cannot be displayed at a national level, except if this person is a celebrity, and that particular happening had spread throughout the country.

Challenge 4: Understanding Spatio-Temporal Scope

It is necessary to extract not only the event information and location, but also its spatio-temporal extent. In mapping applications, the context of what to display is set by the spatial extents of the visualization. When viewing the entire city, events that have a city-level interest should be displayed. As the user zooms in, events of progressively narrower scope must be displayed. For example, a soccer match can be of interest for the city scale, whereas a wedding may be of interest only at the local scale. However, even in the case of a wedding, that event may need to be displayed on higher levels of abstraction if it involves lots of streaming input from other neighborhoods or cities, as may be the case, for example, of a celebrity wedding. A framework to extract spatial extent from live data streams is essential if a reasonable map browsing experience needs to be generated. Furthermore, the temporal scope needs also to be defined to clean old or unnecessary events after a certain time period.

Challenge 5: Efficiency and Scalability

Browsing *event-enriched maps* requires a smooth and fast panning and zooming capabilities. Events of different levels of abstraction are shown on the fly depending on the user navigation behavior. An important challenge here consists in managing input data streams as well as extracted events for efficient processing and retrieval. With the large volume of incoming streams, data indexing and the distributed processing of data represent an essential part of this system. Both real-time and historical data need to be managed and processed for extracting the different categories of events. Consequently, such a system should provide support for both main-memory and disk-resident indexes.

Challenge 6: Provide User Friendly Environment to Explore Event-Enriched Maps

Let's take an example of someone who is planning a foreign trip in coming vacation. He has multiple options for the places to visit and he want to finalize the place on the basis of decision regarding different EoIs in that area. In current scenario, he has a search engine that requires city name, time and name of the event and later, on the basis of results he looks for a hotel nearby. In this respect, there is a need of an interface that allow users to select area on the map

with time and see the list of EoIs with different granularity including country level, city level, and neighborhood level. User can also search specific event and see its video or images to understand it in a better way. Unlike existing query languages, this interface allows to pass spatio-temporal scope or multimedia information of the EoIs along with spatial and temporal parameters.

1.2 Problem Statements

Maps are useful tools for a vast majority of human society. Pre-existing maps are often stylized and static, reflecting only one section of time. Such conventional maps are problematic in compact and unorganized areas: it is difficult to locate POIs or individuals using them. Maps should be full-fledged systems with substantive data sources to increase their ability to detect events correctly and concisely. A different generation of maps has emerged, including digital maps, semantically-annotated maps, and latest maps with live streams that are dynamically disseminated in space and time. Simultaneously, social networking has become an important part of our daily lives. Being heavily equipped with geotagged data, it is explicitly scrutinised by industries and researchers. Consequently, the next generation of maps takes into account the considerable spatio-temporal information emergent from social streams. Hence, map surfing can be augmented significantly by using these unstructured data streams: this can give a new dimension to existing spatial queries.

Given this context, the following constitutes the problem statement of this thesis:

- **Knowledge Extraction:** It is increasingly important in our busy lives today to use all available resources smartly, in an optimized manner, in the execution of daily activities and tasks. Knowledge extraction from various data sources—such as crowd-sourced data, open governmental data, and other online sources—can enable users to manage their daily activities better with the assistance of smart tools. For example, knowledge extraction for dynamic road conditions for a specific city can be obtained by generating, collecting, and analysing rich, crowd-sourced, geo-tagged social media data and, thereafter, validating the content so emergent. This type of information can also be used to improve routing services for large crowds, including tracking users smartly and identifying the ones who are most likely to be affected by upcoming constraints. This should be done in the least possible time along a constraint-aware, optimized path.
- **Knowledge Dissemination:** Current maps are mostly embedded with static information or with some dynamic traffic knowledge: at most, they visualize geo-tagged social media data. In this context, there is a real need to enhance static maps with new, dynamic, knowledge-based layers by extracting knowledge from heterogeneous data sources. This information can also be useful in enhancing existing spatial queries, including different routing approaches or various ways of exploring cities. Following are the two issues that needs to be handled when knowledge disseminates to end-users'.
 - **Spatio-temporal Scope:** It is important to determine the proper spatio-temporal scope and the level of abstraction for detected events at a global scale. This will

allow a system to show live events as per the scale of view: for example, when viewed at city scale, events of higher significance are displayed, while zooming into a neighbourhood highlights events of more local interest.

- **Visualization:** Maps allow interactive browsing. For example, one can see only country names from low zoom level: as one zooms in, one can find detailed views. Visualizing different information on top of a map will affect its usability by displaying clutter or overlapping EoIs (Events of Interest). Therefore, extracting and updating dynamic events on the move and defining their spatio-temporal scope for appropriate display is an urgent need in cartography. The final output of such a system will create a unique and dynamic map browsing experience
- **Efficiency and Scalability:** Additionally, there is still a need to build a complete system that extracts EoIs dynamically from social media streams or other crowd-sourced data. Building such maps requires developing a scalable and efficient system to deal with a variety of unstructured data streams. These include applying sentiment analysis and multi-dimensional clustering techniques so as to better extract relevant information from these streams, and thus inferring the spatio-temporal scope of detected events. The system can handle incoming unstructured data, such as cleaning, filtering, and removing noise on the move, implementing indexing schemes to support efficient access, and memory flushing purposes. Moreover, data from multiple sources must be in a generic format so that it can be efficiently processed to extract (EoI) at a local scale.

1.3 Contributions

The main purpose of this work is to take maps and existing spatial queries one step further by introducing a new scalable and efficient next generation map framework. This framework will be capable of extracting knowledge of geo-tagged, rich, crowd-sourced information. Accordingly, our main objectives are:

- Develop a data collection framework by providing location-based services to identify traffic constraints such as accidents and road closures along with geo-tagged social media data. It also identify users likely to be affected in the least possible time and recommend the best path based on constraints so identified.
- Provide a framework to enrich existing maps with new dynamic layers by extracting knowledge from heterogeneous sources, including unstructured, semi-structured, and structured data.
- Design and develop tools to enhance conventional spatial queries by exploring and visualizing on this enhance maps such as routing, city exploration.
- Design and develop scalable architecture to handle streaming of data from multiple sources with efficient indexing and cleaning.
- Propose models and tools for event detection and discovery in order to extract EoIs and determining their spatial scope concisely, so that they can be displayed appropriately within the right period and in a clutter-free manner.

-
- Validate a proposed approach by designing and developing prototype of the complete system with real-world datasets.

Our main contributions are summarised as the following:

1. **Use of crowd-sourcing and social media to recommend optimized paths:** To being with, we developed an application that collects data from large crowds by offering location-based services, incentive based traffic update modules, and geo-tagged social networks. The system extracts information from data on social networks and mobile applications in terms of traffic constraints such as accidents, congestions, and roadblocks, and recommends the best-optimized path over dynamic road networks. We proposed a spatial grid index to compute the optimized path and to identify affected users within impact zones in the least possible response time. Excepting traffic constraints, we found a lot of relevant spatio-temporal information to be embedded in social media streams. Given that many governments have launched open governmental data platforms, it is possible now to enrich traditional maps with different knowledge-based layers extracted from a plenitude of available data sources. Therefore, the map browsing experience can be enriched noticeably if intelligent maps can discover relevant information from these unstructured data streams.
2. **Concept of enriched maps:** Next, we introduce the concept of enriched maps (called as smart maps) by collecting, managing, and integrating heterogeneous data sources in order to infer relevant knowledge-based layers. Unlike conventional maps, smart maps extract relevant knowledge by applying state-of-the-art text mining techniques to find EoIs (e.g., concerts, competitions, incidents) and online offers and perform statistical analyses (e.g., dangerous areas) by cleaning and encapsulating incoming semi- and unstructured data into structured generic packets. Moreover, we visualize smart map layers in an interactive way and leverage knowledge-based layers to enhance traditional spatial queries. To validate our framework, we developed a prototype of smart maps based on more than 30 million geo-tagged tweets around the world, as well as Yelp, Booking.com, Open Statistical Data from US government websites, OSM road network, and OSM POI list of New York, USA. We also demonstrated a safe and optimized routing, city explorer, and multimedia routing mobile based application on top of the smart map framework. Smart maps with dynamic layers require considerable interaction with the user while browsing, and are confined to a particular city or state. Visualizing multiple knowledge-based layers concurrently makes maps inaccessible and worthless due to cluttered knowledge views.
3. **Scalable and Efficient System:** To overcome this, we presented *Hadath*, a scalable system that extracts dynamic events from social data. This system encapsulates incoming unstructured data into generic data packets even as it preserves accuracy and conciseness. The system implements a hierarchical, in-memory, spatio-temporal

indexing scheme to support efficient access to data packets, as well as for memory flushing. Data packets are then processed to extract EoI at a local scale. The spatial and temporal scope of events must be established to display them on maps: i.e., when a user changes the zoom level, only events of the appropriate scope get displayed. For example, a soccer match may be displayed at the city scale, the opening of a new restaurant at the sub-urban scale, and a house-warming party at the neighbourhood scale. In addition, an accident or soccer match which occurred last week is most likely considered as non-relevant in the current moment, except if a sizeable number of people are still discussing that event. Thus, not only is it necessary to extract events themselves, but it is also important to establish their spatial and temporal scope concisely so that they can be displayed appropriately within the right period and in a clutter-free manner. Finally, all of this has to be done in (near) real-time so that live streams can be created and up-to-date events at multiple resolutions extracted. The final output of our developed system creates a unique and dynamic map browsing experience.

1.4 Outline

This work will be presented through seven chapters:

- In the second chapter, we present a brief literature review of existing mapping technologies and shortest path finding techniques, including multimedia and safe routing, data collection, and event detection. We also introduce existing clustering techniques and existing map based systems.
- The third chapter presents a data collection framework in lack of physical infrastructure environment using location-based services in smartphones and other crowdsourcing data. The knowledge extracted from it can help users to find optimum routes in dynamic environments. For example, we developed a prototype to collect dynamic road conditions via a set of location-based services from a large Hajj crowd by capturing their locations using smartphones. We also collect geo-tagged social network data: this provided precise details about road conditions. The system leverages geo-tagged, crowd-sourced information to identify constraints such as accidents, congestions, and roadblocks. Moreover, by continuously collecting real time, geo-tagged data of constantly mobile users, the system can also find the flow of traffic and road conditions. We propose a spatial grid index to compute an optimum path and to identify affected users within impact zones.
- The fourth chapter introduces the concept of smart maps by collecting, managing, and integrating heterogeneous data sources so as to infer relevant knowledge-based layers. Unlike their conventional counterpart, smart maps extract live events (e.g., concerts, competitions, incidents) and online offers and also perform statistical analyses (e.g., dangerous areas) by encapsulating incoming semi- and unstructured data into structured generic packets. These packets are then processed to extract statistical knowledge on accident-prone and safe areas, and then detect EoIs based on a multi-dimensional clustering technique. This approach provides the framework for delivering different intelligent services and applications, such as: 1) a city explorer which provides the latest information about places and events collected from multiple sources; 2) a route and trip planner; and 3) a multimedia routing mobile app.

-
- In the fifth chapter, we introduce, a scalable multi-resolution event enriched framework that extracts social EoIs from a mass of unstructured data streams. It applies natural language processing (NLP) and multi-dimensional clustering techniques to extract relevant EoI at different map scales. It then infers the spatio-temporal extent of detected events. The framework comprises a data wrapping component: this digests different types of data sources and preprocesses data to generate structured data packets out of unstructured streams. The framework also implements a hierarchical, in-memory, spatio-temporal indexing system to allow efficient and scalable access to raw data. Additionally, it allows extraction of clusters of events from data streams which can be processed to extract events at a local scale. It also introduces a novel mechanism which extracts events from social networking data at different spatio-temporal granularities. This allows users to show live multi-resolution events in keeping with the scale of view: for example, we see events of higher significance when viewing at city scale, and see events of a more local interest when zooming for neighborhood highlights. Later, we explored knowledge extracted from framework and other sources using graphical interface for exploring maps.
 - The sixth chapter demonstrates the feasibility and adaptability of Hadath by displaying events. We also show multi-resolution events at different scales, and simultaneously evaluate the system.
 - Chapter seven gives the general conclusion and presents suggestions for further research.

Chapter 2

Literature Review

2.1 Introduction

Enriching maps with high-level information extracted real-time is vital to many areas of research, including real-time recommendation systems, requiring updated information about surroundings. Leveraging publicly available data makes spatio-temporal EoIs more efficient and beneficial: such data can be used to enhance spatio-temporal conventional queries, including routing and city explorers. Routing is one of the most popular queries that are used on maps. Information on EoIs can give a new dimension to existing routing queries, such as multimedia routing, safe routing. Users can also provide customized queries with the help of EoIs information.

This chapter is organized as follows: we start with a brief overview of existing maps and their layers in section 2.2. Effectiveness of data collection and their necessity in today's world is discussed in section 2.3. In section 2.4, we present an overview of event detection along with scope from Twitter, Flickr, and news sources: this is followed by a clustering technique to merge events of the same real-world entity in section 2.5. Existing systems which show live social data along with performance and scalability perspectives are discussed in section 2.6. Then, section 2.7 explains routing algorithms, including Single Source Shortest Path (SSSP) and All Pair Shortest Path (APSP) approaches; it also discusses existing work on multimedia and safe routing navigation techniques.

2.2 Existing Map System and Mapping Technologies

Maps today are often crowd-sourced. They make use of ‘Volunteered Geographic Information (VGI)’ [1], where users can seed maps with their own information. Researchers, authorities, and industries generate thousands of map-based analytics every year to meet their social and economic needs [2]. In addition, ‘Live Maps’ now contain data that is updated in real-time. For example, live updates of bus schedules, traffic conditions, restaurant opening hours, and road accidents can be displayed on Google Maps and Waze, among others. With the wide spread of social networks, people have started to post their own social contributions on live maps, such as Foursquare check-ins [3], Flickr images [4], tweets (Taghreed [5], MapD [6]), categorized and analyzed points of interest reviews¹, and news RSS feed [7]. Moreover, NLP techniques have been embedded in online newspapers and tweets to extract spatially-referenced news from them [8]. Many of these maps available online have several types of static and dynamic layers. However, Live Maps still lack intelligence in terms of extracting information about new events occurring at different spatio-temporal resolutions.

- **Google Maps**² have satellite view, map view, bicycling layer as static layers, and a dynamic layer for traffic updates. Google also has street view for a few Western cities: these are collected by a fixed camera on top of vehicles in order to record 360° view.
- **Yahoo**³ and **Here Maps**⁴ have map view and satellite view as static layers and dynamic layers for traffic updates, weather, and Flickr.
- **Bing Maps**⁵ have map view, aerial view, bird’s eye view as static layers, and dynamic layer for traffic updates.
- **Mapquest**⁶ has satellite view, map view as static layers, and traffic updates, point of interest as dynamic layers.
- **Open Street Maps (OSM)**⁷ has cycle, transport, humanitarian, and standard view as static layers. OSM is based on a collaborative mapping service, where users are allowed to update maps and upload their GPS traces. It is free to use under open source license.
- **Wikimapia**⁸ is also based on a collaborative approach. It involves crowd-sourced collection of POIs. As of November 2017, it has 28,000,000 places marked by users. It has rail view, road view, ferry view, and river view as static layers.
- **Yandex Maps**⁹ has POIs list, street panorama view, satellite view, transport view, and hybrid view as static layers, and a dynamic layer for traffic jams. It was developed by Yandex in 2004 and is popular mostly in Russia.

¹ <https://www.yelp.com/wordmap/>

² <https://maps.google.com/>

³ <https://maps.yahoo.com/>

⁴ <https://wego.here.com/>

⁵ <https://www.bing.com/maps>

⁶ <https://www.mapquest.com/>

⁷ <http://openstreetmap.org/>

⁸ <http://wikimapia.org/>

⁹ <https://yandex.com/maps/>

- **ViaMichelin**¹⁰ has list of tourist sites, restaurants, hotel booking, car hiring details as static layers, and weather updates and traffic updates as dynamic layers. It is more popular in Europe.
- **Waze Map**¹¹ is mostly used for navigation purposes. It allows users to share information about traffic updates, over-speeding camera, and police cars.

2.3 Data Collection towards Smart Cities

As per World Health Organization [9], the population of cities will double by the middle of this century. The growth in population will impact transportation, responsibilities on authorities, economic growth, etc. Therefore, it is necessary to convert big cities into smart cities so as to provide better facilities to the people [10]. Big giants—including IBM (Smarter Planet-Smarter Cities) [11] and Microsoft (CityNext) [12]—have launched several projects around the world for enhancing infrastructure and human life style. Smart cities, however, have multiple types of sensors, which lead to the problem of data collection, cleaning, and integration [13].

Publicly available data is increasing rapidly. It will keep growing continuously with the advancement of technologies in sensors, smartphones, and the Internet of Things. Data from multiple sources can improve coverage and provide more relevant knowledge about surrounding events and POIs [14]. The strength of one source of data can compensate the shortcomings of another source by providing supplementary information. In the last few decades, data collection and integration has become integral for decision making in recommendation systems, search engines, and even online shopping. Data collection, integration, and methodology are mandatory to design next-generation smart city applications [13]. Moreover, adding social network data such as Twitter and Instagram can provide fruitful insights and complementary sources of information. Social networking sites provide rich and fast streams of data that can be used in online as well as offline analytics. Such data has been used for various scenarios, such as emergency situations, events in cities, and news. Data collection and integration has been done in the past for many applications, such as: 1) decision support on climate change and health by using open APIs and sources [15]; 2) analyses and extraction of POIs from multiple web sources [16], [17]; and 3) finding traffic flow for intelligent transportation service by using Wi-Fi and BLE [18]. In [19], the authors suggested that data collection, integration, and methodology are mandatory to design next-generation smart city applications. A lot of work has been done for web semantics (W3C) [20] by integrating data systems to provide semantic information that can be a valuable input for the development of smart cities. Data integration for multiple sources requires schema mapping [14], data dependency [21], source authenticity [22], [23], and duplicate detection [13].

From state-of-the-art, we can conclude that multiple sources' data collection and integration can play a vital role towards smart cities' development. In this thesis, we propose next generation systems that can enhance maps and transportation system, emergency evacuation,

¹⁰ <https://www.viamichelin.com/>

¹¹ <https://www.waze.com/>

trip planning, decision making, and many more spatio-temporal queries more intelligent with the help of EOIs extracted from data integration of multiple sources.

2.4 Event Detection along with spatio-temporal scope

Detection of irregular happenings and trends from social data—mainly Twitter data—is already a topic of many scientific articles [24], [25]. These mainly classified events into specified and unspecified methods. Specified events are known and rely on specific features and information about particular classes, including: 1) disaster detection, such as earthquakes along with their epicentres [26] or other natural disasters [27] from Twitter; 2) discovering events based on categories using content segmentation [28] and musical events using factor graph model [29]; and 3) discovering incidences related to traffic conditions from Twitter and user generated data [30], [31].

The author [26] detected disaster events such as earthquakes by treating tweets as sensors. To detect location, kalman filter and particle filter are used in earthquake reporting applications. The application notified all registered users faster than messages broadcast by the Japan Meteorological Agency.

Author [27] developed an algorithm to extract attribute-pair values and a mechanism to fill these values in manually generated schema. This allowed them to identify disaster related events using Twitter. They also enhanced the schema by extracting information from embedded URLs inside tweets.

Event detection is based on categories such as art, entertainment, profession, and parties using content segmentation, event classification, and time extraction approaches [28]. To detect the exact location, authors mapped contents with Foursquare and Yelp and retrieved the exact location of events.

The Massachusetts Institute of Technology discovered musical events from tweets using the factor graph model [29]. The system extracted canonical records using Conditional Random Field model, such as name of the artist and venue, using information from a local guide and Wikipedia.

Twitter is another fast source to know about traffic events such as accidents, blockages, and construction. Authors [30], [31] used twitter and UGC to get dynamic traffic conditions to recommend the optimum path from a given location to destination. They used the grid approach to identify traffic constraints and identify users who can be affected in least response time. Later, the system updates about constraints and recommends another optimum path.

Event recommendation and popularity prediction based on microblogging has been discussed [32], [33]. In [32], the authors first detect events by using term frequency and users' social relations. Later, they identified the popularity of upcoming events using diffusion model of content and users' information. They used two real-world datasets to show the effectiveness of their approach. In [33], authors proposed a demo on event recommender system that scrutinized

tweets related to past events and recommended interval and location for the same type of events using item-based collaborative filtering.

Unspecified events are unknown and can be detected using temporal signal, burst words, or trends. These include:

- Extracting and localizing breaking news from tweets as presented in [7], [34], [35]. NewsStand [7], [34] detects breaking news by suppressing noise using Bayesian classifier, using fast and robust tweet clustering algorithm, and retrieving location from the content of news. They have done spatio-textual accumulation of news and present their display using feature-based and location-based approach. News is displayed at different zoom levels based on its importance.
In [35], authors accumulated news from multiple sources, including online news articles, microblogs, newspapers, and TV news programs. They then pre-processed it by providing unique document IDs and extracting relevant information such as time, body, score, URL, and title. Later, they performed clustering based on extracted bursty feature from each document. Finally, they indexed the event for search and visualization.
- Detection of unspecified hot topics based on text similarity or wavelet spatial analysis [36], [37]. In [36], authors used location, time, and tags attached with Flickr images to detect real-world events in two steps. First, they distributed tags and time from Flickr images into event classes that were predefined. Second, they used scan clustering to identify hotspots. The approach is suitable for finding periodic events at diverse locations but irrelevant in detecting small events where fewer Flickr images are available.
In [37], the authors temporally and spatially exploited tags in Flickr images. Later, they identified the tags related with events and used wavelet spatial analysis to suppress noise. The cons of this noise reduction approach are that sometimes it reduces potential events as well.
- Continuously monitoring news streams to detect new events known as First Story Detection (FSD) [38] and to detect events along with their trends [35] using Location Sensitive Hashing (LSH) technique. FSD [38], detects new events from a stream of posts using LSH, which provides competitive results as compared to traditional methods. FSD method has been tested on 160 million tweets with an order of magnitude speedup in the processing time and minor loss in performance.
In [35], the authors proposed methodology to detect events and their trends using LSH from a cluster of tweets. They tokenize the tweets in English, filter them with stop words including http and URL, use TF-IDF and LSH to find interesting events, and, finally, identify trends based on spatio-temporal information and cluster size.

- Early event detection based on abnormalities in signal oscillation over time and finding the approximate time period of an event based on the number of signals of the same type [39].

Events are also detected using clustering of content-based [34], [40], [41] or feature-based techniques [42]–[44]. Content-based techniques compared actual content using text similarity methods, whereas feature-based techniques clustered on the basis of features such as spatial, temporal, and visual. Furthermore, with advancement in technology, event detection through computer vision over nine years of Flickr images has been done for an entire continent [45].

The work presented in [46], [47], [48] is very close to our work with respect to multi-resolution of events. In [46], the event was only distinguishing between local scale based on location with time and global scale using discrete wavelet transformation method without putting focus on mapping those events to different spatio-temporal resolutions in digital maps. The author also presented statistical modelling and analysis about the spatio-temporal distribution of noisy information in Twitter.

Other work on multi-resolution is related to event forecasting methods [47]. Multi-Resolution Spatial Event Forecasting (MREF) model presented in [47] is based on multi-task problems, where each task is marked as a model with specific location and time. Later, these tasks are learned from each other and the framework detects finest location from coarse locations. The framework was tested with 11 datasets, but none of the results were visualized on different spatial resolution.

META [48] framework intended to extract and summarize events from different views with different non-spatial resolutions. None of the work presented above focuses on actual multi-resolution spatial-temporal resolution, i.e. different zoom level of maps. Therefore, mapping of events to different spatio-temporal resolutions in digital maps is still missing.

2.5 Spatial Clustering techniques

The major challenge in mining spatial data is the large size of spatial data along with its diversity and range. Spatial data mining keeps on increasing as end users, industries, and researchers focus more on heterogeneous, geographically referenced sources of data in the field of health services, marketing, event organization, military, environment, etc. To understand the pattern with respect to location or to reduce the size of spatial data for scalability and complexity issues, we need spatial clustering techniques. Spatial clustering is a mechanism which groups' items together based on location: location can be points, lines, rectangles, regions, and density. It is divided mainly into three categories: partition, hierarchy, and density-based methods [49], [50].

Partition method divides the cluster into k -sub clusters based on similarities, where k is the number of initial clusters. These methods are iterative in nature and converge onto some local targets. k -medoid [51], Partitioning Around Medoid (PAM) [52], Clustering Large Applications (CLARA) [53] along with its variance and k -mean [54] are the popular methods of the partition approach.

k-medoid [51] is a classical partition clustering technique that decomposes ‘n’ objects of data sets into ‘k’ clusters. k-medoid algorithm chooses data point as a center and tries to reduce the distance between points that are labeled to be in a cluster.

PAM [52] is an extension of the k-medoid technique where, after identifying k objects, the k clusters are constructed by assigning each object of the data set to the nearest representative object in an iterative way.

CLARA [53] is based on sampling: it draws multiple samples and handpicks the best clustering as the outcome. It also improves time complexity of PAM for large data sets. Clustering Large Applications based on Randomized Search (CLARANS) [55] and Spatial Dominant CLARANS (SDCLARANS) [53] are different variants of CLARA with multiple sample approaches for spatial and non-spatial components.

k-mean [54] is originally for signal processing and is similar to k-medoid except that it chooses the mean of the points instead of the center. The result of k-mean is in the partition of data space into Voronoi cells. It is one of the popular methods of partition category.

The second type of clustering category is hierarchical methods. Hierarchy methods are sub categorized into agglomerative and divisive approaches. In agglomerative, each point clusters by itself and iteratively adds two nearest clusters into one. It is also called bottom-up hierarchical clustering. Clustering Using Representatives (CURE) [56] and ROCK [57] are some of the examples of bottom-up hierarchical clustering.

CURE [56] is an efficient method for huge databases and to detect clusters having non-spherical shapes and wide variance in sizes. It uses random sampling and partition clustering. Unlike other clustering algorithms, CURE does not follow centroid-based approach or all-points approach: it uses a method to pick a well-formed group of points to detect the distance between clusters.

ROCK [57] studied traditional algorithms based on Boolean attributes and categorical attributes and proved that there are many shortcomings in them, such as in calculating the distance between points for clustering. To overcome it, ROCK introduces concepts of links—and not distance—to measure similarity/proximity between a pair of data points. The experimental study with real-life data sets proves better clusters in categorical attributes and is easily scalable for large data sets.

Divisive is the opposite of agglomerative, where big clusters are further broken down into small clusters. It is also called top-down hierarchical clustering. Statistical Information Grid (STING) [58] and Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [59] are some examples of top-down hierarchical clustering.

In STING [58], the spatial area is divided into a hierarchical grid where higher level cells decompose into several numbers of cells at the lower level. The clustering starts from root to

leaf cells of tree in breadth-first search to form clusters. Lower computational cost and the tendency to run in parallel give an advantage to the STING method.

BIRCH [59] employs the concept of Clustering-Feature (CF) balanced tree that consumes less memory and requires single scan of database. This is prepared by joining closed clusters together and rebuilding a smaller CF tree. It is linear with respect to space and I/O time.

Density-based methods constitute the third type of clustering categories. Density is defined as the number of points within certain distance of each other. If the number of points are more than it is denser as compared to less points. Density based spatial clustering of applications with noise (DBSCAN) [60], generalized DBSCAN (GDBSCAN) [61], and ordering points to identify the clustering structure (OPTICS) [62] are some examples of this third category of clustering.

DBSCAN [60] generates clusters with a given minimum size and number of points within a certain distance of each other, known as density. It is able to detect noise points, core points, and points on the border of clusters. It can detect any number of clusters with different sizes and shapes. The lesser the number of points as input, greater the number of clusters with noise. Selecting best minimum number of points to form a cluster is a challenging task and requires several runs to see the results.

GDBSCAN [61] is a generalizes DBSCAN in two important ways. First, it uses the notion of a neighborhood of an object and, secondly, you can use your similarity function based on spatial or non-spatial attributes to calculate the distance between the points.

OPTICS [62] is based on the basic idea of DBSCAN, but it also detects meaningful clusters in data of varying density. The algorithm employs an ordering of the points and equivalent reachability-values to assign the order in which the objects are treated.

There are manifold advantage of density-based clustering, especially DBSCAN, that we used in our implementation: 1) unlike other categories, it does not need prior information about the number of clusters. This is one of the important factors, for while dealing with social media we do not usually know the number of clusters. 2) The average run time complexity of DBSCAN is $O(n \log n)$, where 'n' is the number of vertices in the graph. 3) In contrast with other approaches, we do not need balanced clustering. This is important to detect all types of events, including small, medium, and big events.

2.6 Existing System-Performance and Scalability Perspectives

A considerable body of work has presented systems that visualize geo-tagged social streams on maps, such as Flickr images [4], tweets [5], MapD [63], Yelp reviews, and spatially-referenced news [8], [34]. The Taghreed system [5] provides a mechanism to query tweets and visualize them on maps by using the spatio-temporal indexing technique to run in real-time; this is also used for historic data, where MapD uses GPU machine to handle streaming of tweets.

NewsStand [7] is a scalable system that extracts news from RSS feeds and visualizes them on a worldwide map. Furthermore, the system can apply spatio-temporal and keyword-based filtering of news. However, this system displays news at different spatial scales by only ranking them based on the number of views, without detecting and clustering events of interest along with their spatial scope.

TwitterStand [8] is a system that identifies tweets related to latest breaking news and visualizes them on maps. The author used a naive Bayes classifier [64] to remove noise, i.e. tweets that are not related to news, and leader-follower clustering algorithm [65] to cluster tweets belonging to the same news. Finally, the location associated with the news was determined.

The author in [30] developed a system that used tweets and user-generated content to identify traffic constraints and recommend an optimum path in minimal response time.

The author in [42] developed a system to detect new crime and disaster events (CDE) with their spatial and temporal patterns and usefulness based on the number of tweets talking about particular CDEs.

Other work which has been done on the above as analytics includes: detecting community interest [66], vacation finder [67], event popularity [32] [68], detecting disaster events [26], and forecasting upcoming events [32] [47].

With a large volume of incoming streams, data indexing and the distributed processing of data represents an essential part of any system that implements ‘event-enriched maps’. An important challenge here consists in managing input data streams, cleaning and as well as extracting events for efficient processing and retrieval. Browsing Multi-Resolution Event-Enriched Maps requires smooth and fast panning and zooming capabilities. Events of different levels of abstraction are shown on the fly depending on the user’s navigation behaviour. Both real-time and historical data (up to a certain threshold) need to be indexed and processed for extracting different categories of events. Consequently, such a system should provide support for both main-memory and disk-resident indexes.

2.7 Shortest Path and routing

The shortest path is amongst the most popularly used techniques in Graph theory. It is employed to find the minimum distance between two given nodes, source ‘S’ and destination ‘D’. The shortest path algorithm finds use in routing queries to determine the best path in terms of time and distance. It has numerous approaches, such as static, dynamic, time-dependent, stochastic, replacement path, alternative path, weighted region, and parametric [69]. Here, we discuss Single Source Shortest Path (SSSP) and All Pair Shortest Path (APSP) along with existing, state-of-the-art multimedia and safe routing protocols.

2.7.1 Single Source Shortest Path

The simplest instance of SSSP is when the graph is unweighted. As Cormen et al. [70] recommend, the breadth first search can be employed by commencing a scan from a root vertex and then inspecting all neighbouring vertices. For each neighbouring vertex, SSSP probes non-visited vertices till it identifies the path with least edges from the source to the destination vertex.

Dijkstra's algorithm [71] solves the SSSP problem from a given vertex to all other vertices in a graph. This algorithm is usually employed in directed graphs with non-negative weights. It identifies two types of vertices: (1) solved and (2) unsolved vertices. To begin with, it sets the source vertex as a solved and then checks all other edges—through unsolved vertices—connected to the source vertex for the shortest path to the destination. The corresponding vertex is added to the list of solved vertices upon identification of the shortest edge by the algorithm. This process is repeated until all vertices are solved. In this way, the algorithm achieves a time complexity of $O(n^2)$. One of its advantages is that it need not investigate all edges: this is particularly useful when the weights on some of the edges are expensive. However, major disadvantages are that the algorithm deals only with non-negative weighted edges and that it applies only to static graphs.

Furthermore, Dijkstra's algorithm is considered a greedy algorithm because it performs a brute-force search in order to find the optimum shortest path. It follows a successive approximation procedure based on the optimality principle advocated by Bellman Ford [72]. In other words, it can employ the reaching method to solve the dynamic programming equation [73]–[75]. Dynamic programming tackles sub-problems and, so, circumvents the brute-force search process. Algorithms premised on dynamic programming are able to probe an exponentially large set of solutions even as they simultaneously avoid explicitly examining all possible solutions. The two versions of Dijkstra's algorithm—greedy and dynamic programming—are the same when it comes to finding optimal solutions, though both may present different paths to reach them.

Fredman and Tarjan [76] made an improvement over Dijkstra's algorithm with the help of a Fibonacci heap (F-heap). Employing it results in $O(n \log n + m)$ running time because the total incurred time for the heap operations is $O(n \log n + m)$ and the other operations cost $O(n + m)$. Fredman and Willard [77], [78] introduced an extension which includes an $O(m + n \log n / \log \log n)$ variant of Dijkstra's algorithm through a structure referred to as the AF-Heap. This structure provides constant amortized costs for most heap operations and $O(\log n / \log \log n)$ the amortized cost for deletion.

Improved priority queue implementations present another line of optimization. Boas' [79] implementation is based on a stratified binary tree, a method which enables online manipulation of a priority queue. The resultant algorithm has a processing time complexity of $O(\log \log n)$ and storage complexity of $O(n \log \log n)$. The presence of an analogy between sorting and the SSSP problem was indicated by a study conducted by Thorup [80], in which SSSP is not harder

than sorting edge weights. This study [80] also describes a priority queue resulting in a complexity of $O(\log \log n)$ per operation and $O(m \log \log n)$ for the SSSP problem.

SSSP algorithms report distances from a given source vertex. Dynamic algorithms compute update and query operations online. While the update operation inserts, deletes, or modifies the edge's weight, the query operation probes for distance from the source vertex to a given target vertex.

Fakcharoenphol and Rao [79] proposed an algorithm for planar graphs with real-valued edge weights. It achieves a time complexity of $O(n \log^3 n)$ and performs update and query operations in $O(n \log^{4/5} \log^{13/5} n)$ amortized time.

Bernstein and Roditty [81] proposed a dynamic shortest-paths algorithm capable of achieving an update time superior to $O(n)$ without loss of query time. Significantly, they obtained $O(n^{2+o(1)})$ as total update time and constant query time. This algorithm can achieve its results premised on moderately sparse graphs. Consequently, Bernstein and Roditty proposed two randomized decremental algorithms operating over unweighted, undirected graphs for two approximate shortest-path problems.

2.7.2 Multimedia Routing

The term “geo-tagged multimedia” is not innovative in the state-of-art. It has been widely used in various scenarios. However, using such data to add semantics to routing services is a novel approach. It is tailor-made to users' smartphone bandwidth and resolution requirements. For example in [31], [30], geo-tagged multimedia tweets and multimedia user generated content are used to identify traffic constraints—such as accidents and road closures—along with multimedia information to inform users. The framework informs users about traffic constraints and suggests new optimized paths in minimal response time using the spatial grid approach.

The authors in [82] were the first to propose work on venue semantics using user generated content. Unlike existing state-of-the-art, system provides semantic information about POIs at given locations. It uses a semantic algorithm to recommend interested POIs based on data collected from Foursquare and Instagram.

The authors in [83] presented a landmark-based navigation mechanism to collect POIs based on attractiveness in order to identify routes. They developed a method to identify landmarks automatically from given datasets by visual, semantic, and structural attraction.

The author [84] detected landmarks by computing visit popularity, indirect visibility, and direct visibility using Foursquare check-ins, geo-tagged tweets, and digital maps' data. The author visualized routes on web based applications with detected landmarks and recommended that landmark based routes are easier to remember as compared to Google based-navigation.

Our system stands out from [31], [30], [82] in that it: 1) collects the source and destination of the route from geo-tagged multimedia information submitted in real-time for route discovery; 2) shows publicly available POIs with multimedia associated with them within a certain radius of the calculated route to semantically help users; and 3) indexes different types of multimedia data separately for efficient real-time retrieval. Moreover, we have enhanced the indexing method [83], [84] to support different types of real-time multimedia data with high arrival rates.

2.7.3 Safe Routing

The term ‘hotspots’ and ‘hot times’ in a criminal activity and accident-prone roads are very well known in state-of-the-arts: considerable work has been done to detect the same. The rationale of safe routing is to present the safe fastest path from source to destination, including crime free and accident free areas. Safety awareness is a major aspect that determines social comfort [85]. Safety awareness and sentiments about places can be used to measure actual number of crime cases [86]. Thanks to recent changes in policies that allowed governments of several countries to publish transparent and open government statistical data including US [87], Australia [88], Canada [89] about accident and crime.

The author [90], [91] detected hotspots using individual diary data along the route for better understanding of the path. They used spatial clustering (DBSCAN) to identify the clusters and also compared the results with non-parametric method KDE.

SocRoutes [92] demo used real-time geo-tagged tweets’ sentiments to find safe and pleasant paths for walking, bicycling, and driving with little increase in the distance as compared to normal paths. The demo proves that there exists a relation between negative sentiments and crime areas: it bypasses such areas while suggesting routes.

Crowdsafe [93] demo provides a mechanism to update interested users about crime information in real-time along with search and report crime features for a given location. The demo leverages crime information to schedule safe and convenient paths. The visual analytics of the system can be beneficial for authorities and governments to handle crime.

The patent [94] used multiple types of data—such as airbag deployment data, accident data, vehicular crime rate data, general crime rate data, and general characteristic data—to recommend safe and optimized route plans.

Other work on safe paths includes: 1) finding safe path for a robot that avoids collision with stationary and non-stationary obstacles [95]–[97]; 2) safe path for military groups to go from one place to another place by avoiding enemy areas [98]; 3) safe path for cruise missiles that uses grid based approach and mandatorily passes from certain grids [99]; and 4) unmanned aerial vehicles to reach their destination safely [100], [101].

We have used knowledge from crowd-sourced data and open government data to identify accident prone areas as spatial points or lines and crime prone areas as spatial polygon. Later,

we used spatial data while recommending safe optimized paths based on accident free and/or crime free areas.

2.8 Conclusion

In this chapter, existing map systems and mapping technologies are discussed including details about different layer. We also presented work done about role of multiple sources data collection and integration towards smart city development. Later, we discussed different techniques of event detection from social media data. We also reviewed work done to detect specified events and unspecified events. After detection of events, we need a method to cluster event that belongs to same real-world entity into one cluster based on TF-IDF and spatial criteria. Therefore, we presented many work that are done to achieve spatial clustering. Later, we showed different existing systems that visualizes social media or news on maps. Finally, we presented shortest path techniques for single source shortest path and all pair shortest path along with multimedia and safe routing.

Chapter 3

Constraint-Aware Optimized Path in Lack of Physical Infrastructure Environment

3.1 Introduction

Traffic congestion is one of the most challenging problems in urban areas around the world. In the US, *The Atlantic* estimated that “congestion causes urban Americans to travel 5.5 billion hours more and to purchase an extra 2.9 billion gallons of fuel”¹². Both blockages and emergencies can be avoided and managed by timely detection of dynamic road conditions. Map-based services need to be efficiently adapted to navigate to points of interest (POI): this will require computing an optimal route between users’ locations and their selected destinations. Presently, applications offer users the choice to select the shortest route along with other options as per existing road conditions, such as road accidents, traffic jams etc. Road accidents and temporary blockages become big challenges in lack of physical infrastructure environment i.e. if there are no real time updates of road maps. Users too use different applications, which may well produce different results depending on their location, data, dynamic road conditions, and design. This problem becomes even more complex in large, culturally diverse crowds. In these contexts, real time crowd-sourcing can provide relevant input data to current applications for computing optimized routes to existing and forthcoming users. For example, we studied the crowd during Hajj, when more than three million pilgrims from over 140 countries congregate every year to for a week. Users need intelligent routing mechanisms to ensure hassle-free mobility between different spatio-temporal zones during their stay in Makkah [102].

¹² <http://www.theatlantic.com/business/archive/2013/02/the-american-commuter-spends-38-hours-a-year-stuck-in-traffic/272905/>

Table 3-1, indicates some statistics for the last three consecutive Hajj crowds (1433, 1434, and 1435 Hijri¹³) in terms of registered pilgrims, number of vehicles used, and number of pilgrims per vehicle, assuming that all vehicles carried an equal number of pilgrims. However, it should be kept in mind that a considerable portion of these pilgrims joined the crowd locally and were not registered.

Table 3-1: Number of Pilgrims, Vehicles used and pilgrims per vehicle for the last three years of Hajj¹⁴

<i>Year</i>	1435 Hijri	1434 Hijri	1433 Hijri
<i>Pilgrims</i>	2085238	1980249	3161573
<i>Vehicles</i>	46108	31567	99444
<i>Pilgrims Per Vehicle</i>	45	63	32

The current advanced state of mobile technologies enables users to share geo-tagged content with timelines through social networks. Geo-tagged data offers information about the location and time from where users share content. Investigating this data can lend semantics to the prediction of dynamic road conditions. These conditions include traffic flow, temporary road blockage due to flooding or construction, outdoor advertising, etc. Recommending optimum paths as per these conditions can fruitfully influence users' navigation choices. Such an optimal route can serve as a key service for large crowds (Table 3-1). Globally, the availability of smartphones makes it possible to provide raw sensory data which can analyze users' locations so as to provide location-based services. For the purposes of this study, we assume that a large number of pilgrims during Hajj carry smartphones with high speed internet. We then assume that they used location based services through our deployed application during Hajj 2014 [102] [103].

Figure 3-1 shows the percentage of different services used by pilgrims based on users' location during Hajj 2014. Among this basket of services, complaints, traffic updates, nearest mosque, find favorite POI, currency exchange, find POI, and find friend services add up to 49.1%. These users need an intelligent routing service that can resolve their problems by helping them reach their PoIs.

¹³ Islamic Year

¹⁴ Central Department of Statistics and Information, KSA

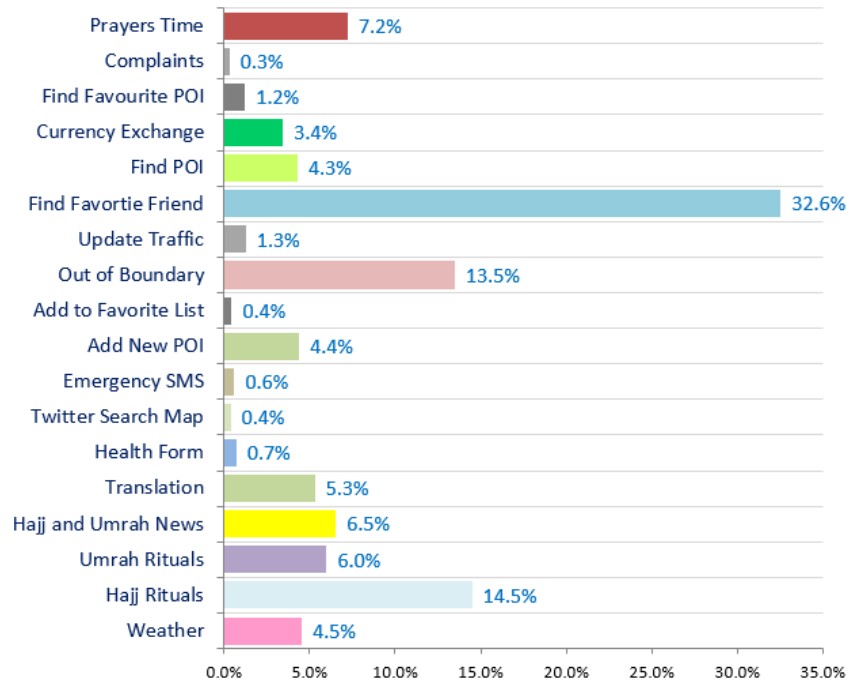


Figure 3-1: Percentage of different services used by pilgrims in Hajj 2014 (cf. ‘Perform Hajj and Umrah Application’¹⁵).

Figure 3-2 reveals a traffic constraint collection framework. In this framework, constraints are collected from social networks and mobile applications by offering location-based services. Proficiently extracting social networks and other sensor-based data and making them available for real time computation is a major challenge. In this study, we present a framework to recognize evolving situations from massive data streams taking crowd-sourced data from users of our deployed iOS/Android¹⁵ based application during Hajj 2014 [102]. This chapter uses a framework which captures users’ data as it gets generated when they interact with location-based services and social networks. The system also store geo-tagged data collected from the Hajj application we deployed: this occurred even as users continuously interacted to find the trajectory flows of the roads. This helped in identifying the live condition of traffic, the time it took to pass by an intersection, etc. The system can, thus, be used to analyze relevant information about road dynamics. Whenever a user searches for an “optimal” path from a given source to a destination, our proposed system processes this query by: 1) searching collected geo-tagged social networks and mobile application crowd-sourced data; 2) extracting related information which might have a bearing on the query; and 3) finding the optimal path based on the real time, dynamic road conditions [31].

¹⁵ <https://play.google.com/store/apps/details?id=com.hajjandumrah&hl=en> ,
<https://itunes.apple.com/us/app/perform-hajj-umrah/id917265874?ls=1&mt=8>

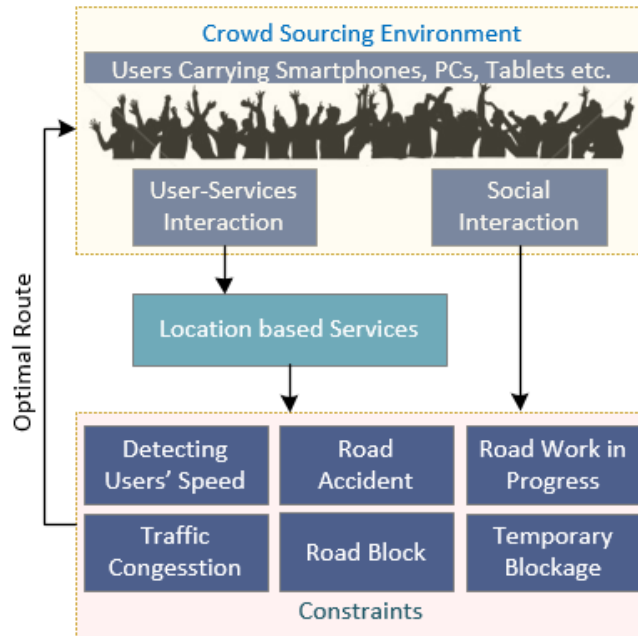


Figure 3-2: Constraint collection framework.

The remainder of this chapter is as follows: Section 3.2 describes the services and routing challenges. Section 3.3 presents our constraint-aware optimized path algorithm. Section 3.4 presents an overview of the system architecture. Section 3.5 highlights different implementations and test evaluations. Finally, section 3.6 draws conclusions and future challenges.

3.2 Services and Routing Challenges

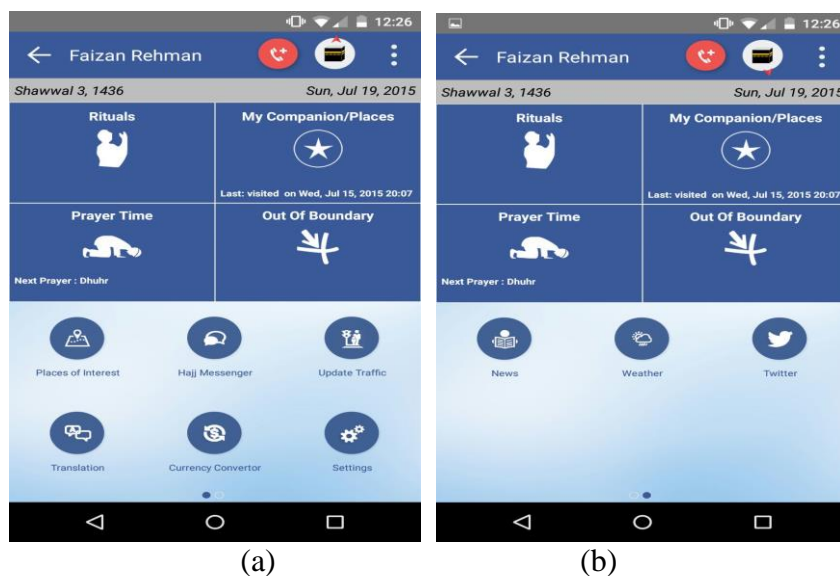


Figure 3-3: Main landing page of the new Hajj and Umrah Application (2015) with the list of Services

In Hajj 2014, we have launched a suite of applications for pilgrims and family members. Figure 3-3 shows the list of services in the new application. Based on the previous application survey published in [102], we found rituals, my companions/places, prayer times and out of boundary services were used the most. This year, we are enhancing the application based on users' experience. Not only we have resolved the usability issues, we have also enhanced the functionalities especially with respect to routing. Routing is one of the heavily used services that pilgrims used while finding friends and point of interest. We have categorized the services of the launched application in three categories namely location-based services, routing-based services, and data collection services. These services are briefly described below

3.2.1 Location-based Services

Following are the list of services where we are fetching user's location in order to provide location-based services.

My Companion/Places shows the list of added friends and favorite places by their locations. **Prayer Times** shows the timing of the prayers and nearest mosque. **Out Of Boundary** works as a geo-fence and continuously fetches and updates user's location after every 20 seconds. It will inform the user whether he is inside of Haram, Mina, Muzdalifah or Arafat boundary. **Places of Interest** shows nearby point of interest. **Hajj Messenger** works as a messenger to share the text, audio and images. All the data that are exchanged are geotagged. **Currency Converter** shows rate of the currency conversion and the nearest currency exchange shops with respect to user's current location. **Weather** shows the weather details with respect to user's current location. **Emergency SMS** allows users to share the location through SMS. The application provides 3 free SMSes when user submits any traffic updates.

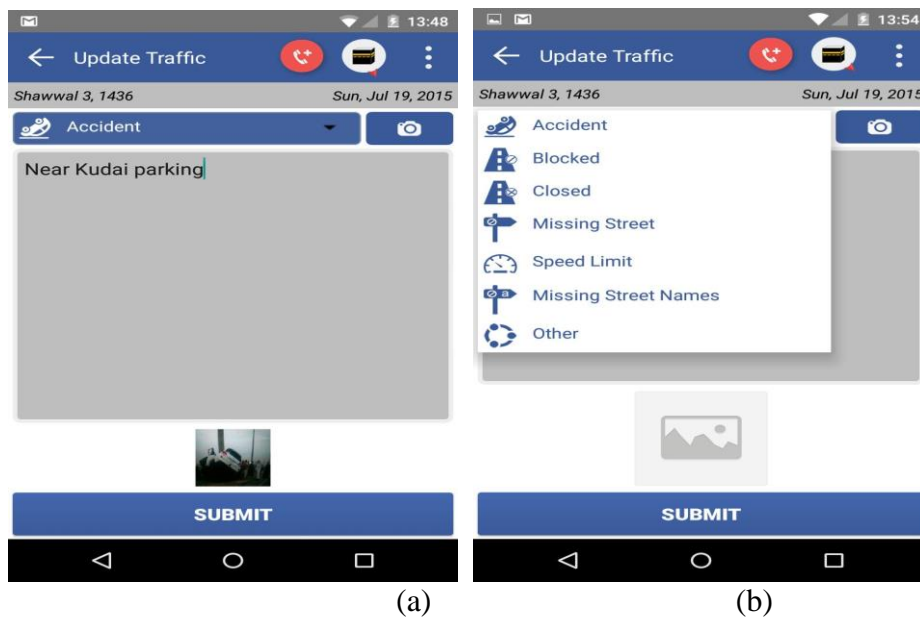


Figure 3-4: Incentive based traffic update module (a) Accident with image and comments (b) List of other options in the drop down

3.2.2 Routing based services

Following are the list of services that use routing algorithm for navigation. The system stores the user's updated location after every 20 seconds.

My Companion/Places to find the routes towards the friends, family members and favorite places. **Prayer Times**: to find the routes of the nearest mosque. **Places of Interest**: to find the routes towards the selected point of interest. **Currency Converter**: to find the routes of the nearest currency exchange.

3.2.3 Traffic Updates Data Collection Service and Geotagged Crawler

To collect traffic updates, we use geotagged social network data and incentive module based mobile application. Crowdsourcing data helps other users to inform about dynamic road conditions.

- I. **Mobile Application**: Figure 3-4 shows the traffic update incentive module, where pilgrims can share the traffic updates to our system. System administrator will verify the traffic update using admin login web based portal on our product website¹⁶.
- II. **Geotagged Social Network**: Apart from traffic update through our developed mobile application, we are also collecting geotagged tweets from twitter. There are many hash tags in twitter that are related to traffic for each and every city such as @NEWS1130Traffic¹⁷, @jed_traffic¹⁸. Figure 5 shows the visualization of the tweets using taqhreed [5] that our system stores and related to the traffic constraints such as accident, road closed, road blocked and congestion.



Figure 3-5: Visual Interface of the tweets related to traffic constraints

¹⁶ <http://www.smarthajj.com>

¹⁷ <https://twitter.com/NEWS1130Traffic>

¹⁸ https://twitter.com/jed_traffic

3.2.4 Routing Challenges in large crowd

We are enhancing the application and resolving following issues that pilgrims faced in hajj 2014:

I. Dynamic Road Updates

In large crowd gathering, managing traffic is a big challenge. Traffic ministry closes and open many roads at real-time, which makes the routing application ineffective. In Hajj 2015, we are planning to collect dynamic road updates using incentive module based mobile application (as shown in Figure 3-4) and geotagged social network.

II. Incomplete Road Network

Figure 3-6 shows the output of the navigation in our old application and google maps. Google map shows more accurate and shortest way as compared to the OSM. On analyzing the issue, we found that road network of Makkah is not complete in OSM. For next Hajj, to complete the road network, we will install the GPS device ASTRA AT200¹⁹ in cars to collect the GPS traces after every 5 seconds and store it in openGTS. Later, we convert the openGTS data into GPX format and upload in Open Street Map. We have tested the framework for small part near to Arafat to complete the road network.

III. Online Maps

Routing is one of the heavily used services when people are at new place either to find point of interest and friends or family members. Many pilgrims find it difficult to use due to the requirement of Internet and online maps. To further enhance the service in Hajj 2015, we have developed Offline maps using Mapbox²⁰ library and working on offline routing service.

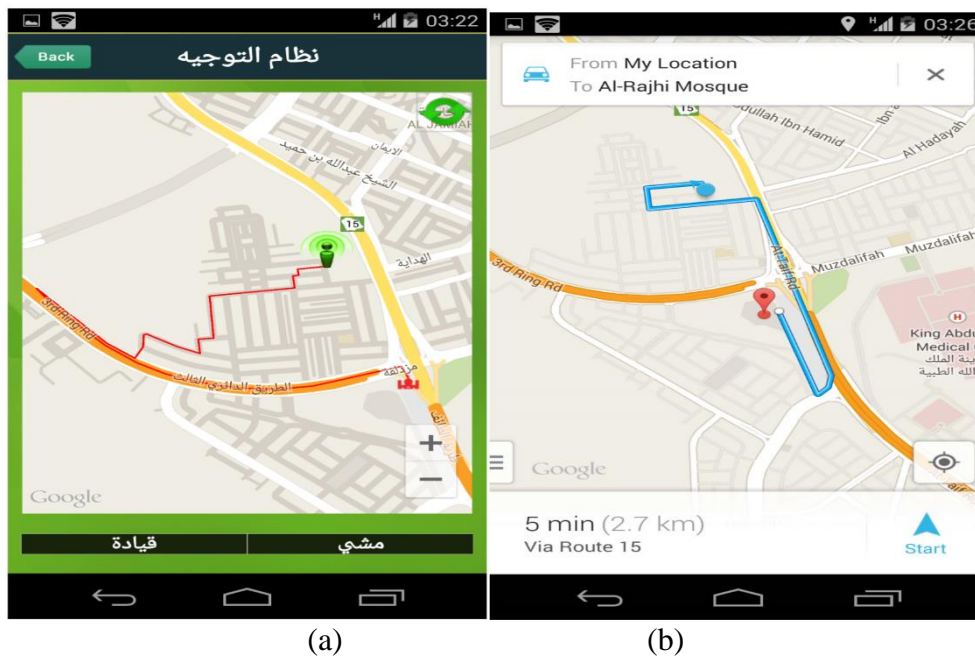


Figure 3-6 Comparison of Navigation issue based on A) Open Street Map Data in our Hajj and Umrah Application (2014) and B) Google Maps

¹⁹ <http://www.gps-telematics.co.uk/vehicle-tracking-products-at200.htm>

²⁰ <https://www.mapbox.com/>

IV. Optimized Path Algorithm

Based on the data collected framework for dynamic road updates, our proposed algorithm in section-iv will help the pilgrims in next hajj to recommend optimized path. In case of any constraint, the system will find the affected pilgrims, notify them and recommend new optimize path.

3.3 Constraint-Aware Optimized Path Algorithm

3.3.1 Modelling

In this work, we implemented an algorithm that detects nearby users coming towards the constraints such as accident or roadblocks, in a minimum response time. The users are provided an alternative optimized path and are informed about the dynamic road conditions. A grid-based modeling approach is presented in this section which is used to update paths for the queries based on road conditions. This modeling approach helps finding the affected users for a given query in a minimum response time by dividing the world into regular square cells. A particular cell identification number ($cell_i$) is assigned to each user, constraint and trajectory.

$$Cell_i = \left\langle \begin{array}{l} Users_{1,2,3...u}, Constraint_{1,2,3...c} \\ Trajectories_{1,2,3...t} \end{array} \right\rangle \quad L \quad (1)$$

If a user ' u_0 ' met a road accident and finds the road is blocked/closed due to it; other users u_1, \dots, u_n have to be averted that are moving across the area because of this accidental point. This traffic congestion can be prevented if the information provided by the user u_0 is communicated to the rest of the users. The suggested alternative paths helps to avoid traffic congestions. Each constraint message occur at location "X" has the following attributes:

$$X = \langle lat, long, tm, txt, img, src \rangle \quad L \quad (2)$$

Where lat , $long$ are the latitude and longitude, tm is the time, txt is the textual information about the accident, img is the image shared by the user, and src is the source of the message (social network, Hajj and Umrah mobile application, etc.).

Our aim is to disseminate this information to other users approaching this point "X". The earliest this information is sent, the lesser would be the congestion on the roads.

To reduce the response time, the grid approach is very helpful, we just need to search for the affected user in current cell (where constraint occurred) and neighbor cells.

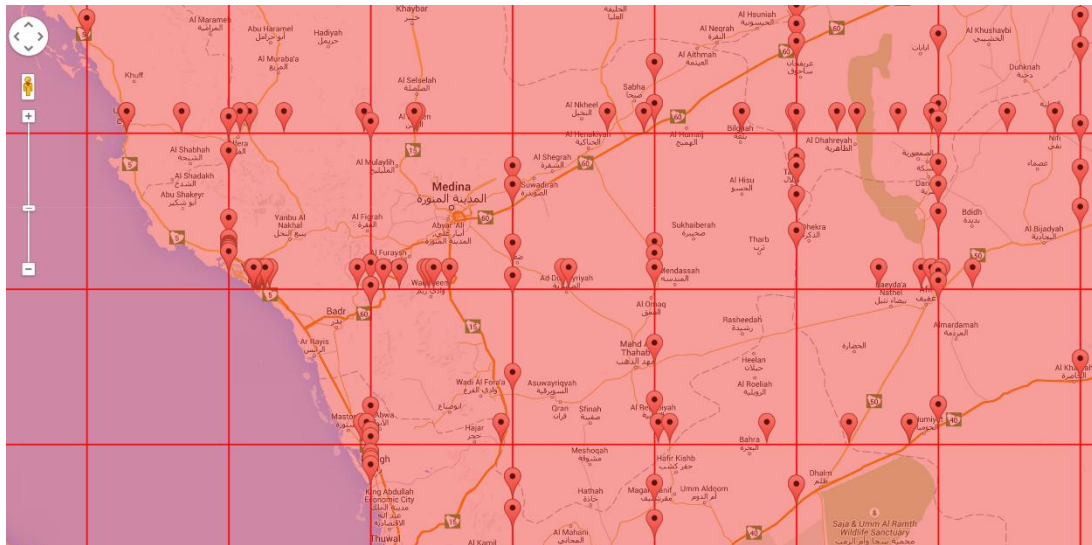


Figure 3-7: Show cells and boundary points after the initial step of the algorithm

3.3.2 Algorithm

The algorithm takes the minimum and maximum latitude, longitude, and size of cells to create the spatial grid. Then it marks all the intersections of the roads that are on the boundary (b) of cells. Then it deletes the edge from source(s) to destination (t) and creates two separate edges i.e. (s, b) and (b, t) as shown Figure 3-7. We have consider the size of cell is fixed based on the spatial resolution, which is application-dependent. Each cell has the size of (one degree of *latitude* * one degree of *longitude*). We can run the algorithm in the following two ways:

- Without Preprocessing

We use the modern hardware and multithreading to use multi-directional Dijkstra algorithm for all the cells that are in between source to destination as we have marked intermediate points on the boundary, which helps to run the algorithm parallel in intermediate cells.

- With Preprocessing

We calculate the shortest path for all the boundary points with in the cell and stores the preprocessed data in spatial database. In case of any query from source to destination, we calculate shortest path only from source to boundary points and target to boundary points for the current cell and used preprocessed data of in-between cells.

Algorithm 3-1: Pseudo code to identify the affected user, notify them about constraints and recommend optimized path

Data: *constraint;*

constraintType<*constLat*, *ConstLng*>

Output: Recommend Optimize path to the affected user:

- 1 | *Constraint Aware Query Processor receives constraint from Relational Database*
 - 2 | *Search for the cellid based on constLat and constLng*
 - 3 | *Search all the eight neighbor's cells of the current cell*
 - 4 | *Fetch the list of active users of all the nine cells (current affected cell Id and all its eight neighboring cells)*
 - 5 | *Update current position and remaining path to destination of all the active users*
 - 6 | *Update route and query table for all the users of the affected cells*
 - 7 | *Trace the remaining path of all the users and find out the affected users by checking their remaining path*
 - 8 | *Send Notification to affected users about the changes in the road network*
 - 9 | *Recommend new optimized path to affected users from their current location to the destination*
 - 10 | *Update the entry in relational database*
-

This algorithm, based on dynamic road conditions, recommends the optimized path. It tracks all the active path queries, starting and destination points for a given user, the current user location, and the remaining path for a given query. This component detects the constraint type and location after receiving any constraint from the repository. The particular edge and its corresponding value of speed and time is then updated. The system notifies all the users affected by that constraint through a push notification. Based on the current location of the corresponding users, constraints-aware optimized paths are then computed. The current situation of the dynamic road network is taken into consideration by the system for any new request issued by a different user. In order to recommend an optimized path, the algorithm's knowledge of road conditions received from the repository such as the current flow for traffic, and the time for crossing intersection. Algorithm 3-1 shows the pseudo code of optimized path recommendation.

3.4 High-Level Architecture

Figure 3-8 shows the high-level architecture, which is designed and implemented using the **Model View Controller (MVC)** approach and distributed among presentation, business and data tiers. The **presentation tier** is responsible to control the user requests by sending commands to the model to update its state. The user also sends commands to its associate view to change the views presentation of the model. On the **business tier**, the model is responsible to communicate with the **data tier** in order to store and retrieve data as per the controller's request and displays on the view. The view on the other hand requests information from the model. The model uses the request information to generate an output representation to the user. The **business tier** is flexible enough to communicate with the third party system using the REST FULL API's. The **data tier** is further divided into two different types of RDMBS to satisfy high scalability, high performance, distributed, and robust environment.

Third Parties API's: There are some concerns in using the third party API's. For example, free or not free is not the only measure for company to run the service to support the API. Other important concern is the transparency in the latency and uptime, as API calls are happening in real time and easy availability of status check is very important. **Twitter**²¹, **Yahoo weather**²², **XE currency exchange**²³ provide a status page for current and historical performance and availability.

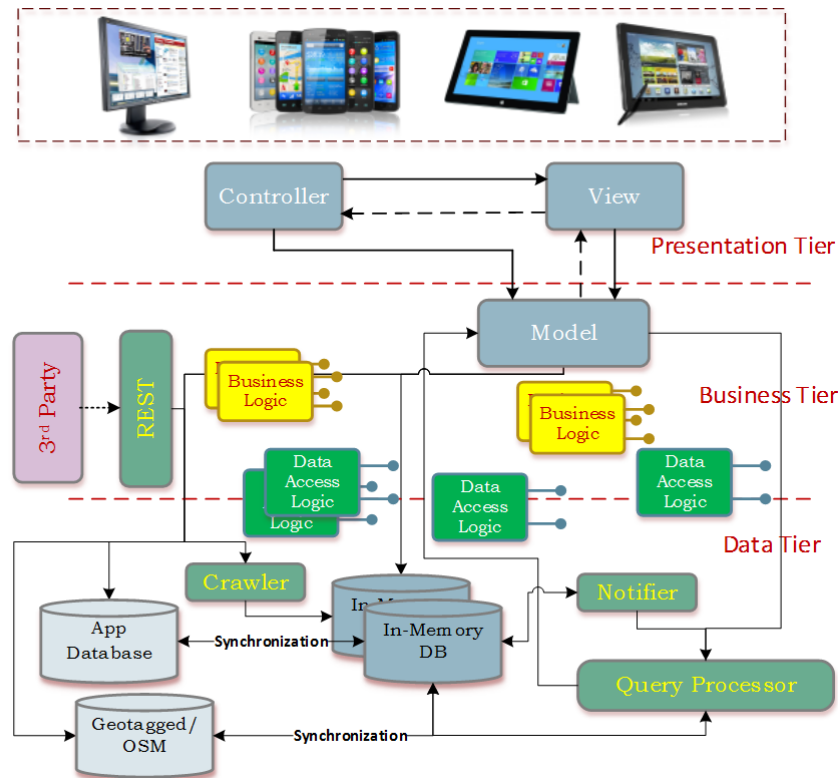


Figure 3-8: High level system architecture

²¹ <https://dev.twitter.com/rest/public>

²² <https://weather.yahoo.com/>

²³ <http://www.xe.com/>

Providing real time services requires efficient and reliable data storage as well as a retrieval mechanism. We maintain several **database** according to the relevant information such as data receive from our application is stored in **App Database**, twitter data in **Geo-Tagged Database** and **Open Street Map Data** (OSM) Database which is one time storage. The **In Memory Database** is used to store the live sessions and significant information such as profiles, friends, family, current and trail of history location, and most frequently access services.

Crawler is a java-based routine that runs continuously and collect social geotagged data for the Saudi Arabia from the twitter server. It contains a rich source of information including the name, user profile, time, location, language, followers, city, and country for a given tweet. The crawler is also responsible for data archiving at the end of every week.

The information collected by crawler is searched by the **Notifier** on predefined keywords like accident, roadblock, congestion etc. on per minute basis. As soon as the keyword hits, the system starts searching the In Memory Database for the users present in the same cell or neighbor cells and sends notification to each user in case their path is affected by constraints.

The **query processor** executes spatio-temporal queries through efficient and flexible retrieval techniques to provide low query response, high throughput and low latency which is of the order of milliseconds and feeds the query answer to the Model to be viewed by the end users through the View on the presentation layer.

3.5 Implementation and Test Results

3.5.1 Implementation

Table 3-2 illustrates the technologies used to develop the complete system. We have used Java technology to create a highly scalable, secured, multi-threaded, high performance, distributed, and robust server application. We have preferred Java 8 over 6/7 because multiple security issues are resolved, and it is highly configurable and efficient in a multi-threaded environment. Spring 4.1.x works more efficiently in Java 8 environment. We have also used Servlet version 3.1 in the project to build the project accordance with servlet architecture used in Tomcat 7.x +. Servlet 3.0 allow asynchronous request processing but only traditional I/O was permitted, which can restrict scalability of applications. Non-blocking I/O feature of Servlet 3.1 allows to build scalable applications.

Presentation Tier Android and iPhone Mobile applications send asynchronous web service requests to business tier, in order to process the request asynchronously. We have used asynchronous response handling feature of servlet 3.1.

Business Tier each request is handled by a thread, and if all the threads are busy, response is put on hold until any thread is free while response has not timed out i.e., 50 second for most of the API. All the critical APIs e.g., create user, login, save log-file data, etc. are handled in queue, asynchronously. Log file of each user is processed by thread, asynchronously, in queue. We have tested the logging API and it takes 30ms to 5s, in general case. Suppose all the application threads are busy meanwhile some of the threads are busy logging file to database and someone tries to create a user, in that case it may affect the user. As we are planning to use auto-scaling

group and load balancing, such situation should be rare. Business Tier supports modular architecture using Spring 4 for security, and dependency injection.

Technology	Java 8, Servlet 3.1.0.
Business Layer	Spring MVC (4.1.6.RELEASE), Spring Security (4.0.1.RELEASE), Spring Security OAuth2 (2.0.7.RELEASE), Spring based REST web-services, Java Bean.
Database Layer	Hibernate (4.3.10.Final), Hibernate-spatial (4.3.1-SNAPSHOT), C3P0 connection pooling, EhCache, Envers auditing, Postgis-JDBC, JDBC.
Database	PostgreSQL 9.4, Postgis 2.1, Apache Geode.
Web Server	Apache Tomcat 8.x.
Platforms	Linux, Windows.
IDE	Eclipse Luna 4.4.2.
Others	Maven, Log4J, Jackson-databind 2.5.2, HTTP-Client 4.4, GCM server, APNS.

Table 3-2: Server Implementation Detail

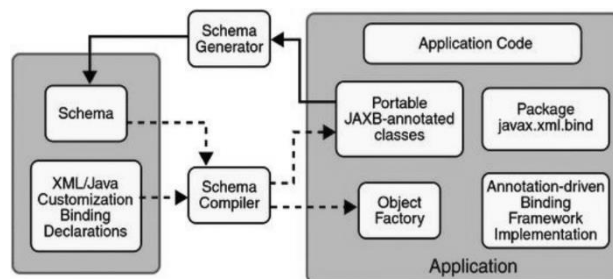


Figure 3-9: JSON/XML based serialization /de-serialization

In Figure 3-9, the Jackson data bind technology is illustrated, which is very effective and efficient. It converts JSON/XML at runtime from objects on the basis of content-type, and parses the request / response body in JSON/XML format by serializing or de-serializing the Java bean, at runtime. It means if client is requesting XML then server provides XML response and if client requests JSON then server provides JSON response with same code based on “Accept” header value. And also, if client sends XML then server processes XML request and

if client sends JSON request, server processes JSON request with the same code based on “Content-Type” header.

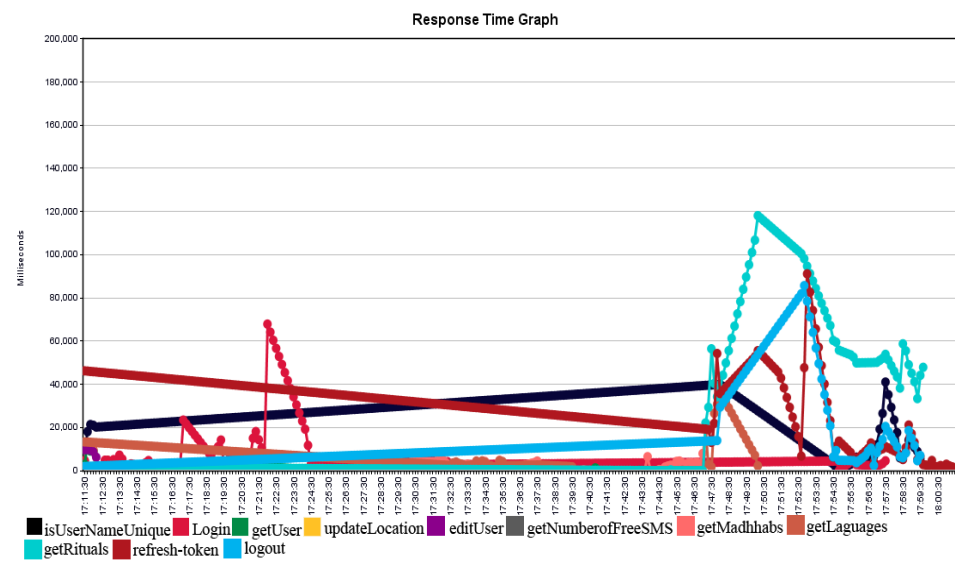


Figure 3-10: Response time graph for load testing.

Database Tier we used Hibernate ORM framework (4.3.10.Final), which is a robust, scalable and database independent framework and enables us to wrap tables and views in JAVA classes, so that instead of writing SQL statements to interact with database, methods and properties of objects can be used. We have used Hibernate-spatial framework to handle geographical data and functions of OSM database. Currently, Hibernate-spatial framework provides more support to geometrical data than geographical data. In Hibernate 5.x spatial is a part of ORM and all the geographical data objects and functions are also provided. We have used four databases. **Application database** for user-details, routing queries, trajectories, routes and configuration tables in PostgreSQL. **Geotagged Database** to store and process the tweets in PostgreSQL [31][5]. **OSM Database** used for queries to fetch navigational map between source and destination, in PostgreSQL database. OSM database with separate Hibernate C3P0 connection pooling for each database. In Memory, **Apache Geode database** to store and retrieve live sessions and frequently used information.

We store the movement of each user interacting with social network or using location-based services continuously in trajectory table $\langle t, e, u, time \rangle$ where t, e, u is the identifier of trajectory, edge and user respectively [31][104]. We store routing queries $\langle q, u, r, snode, cnode, dnode, curcellid \rangle$ in the query table, where q, u, r are identifiers of query, user and route respectively. $snode, cnode, dnode$ are the source node Id, current node Id and a destination node Id. $curCellId$ is a current cell Id. Route Id stores the complete list of edges that are in the path in a separate route table.

3.5.2 Evaluations and Routing Algorithm Output

We have performed load testing the APIs of Smart Hajj application on Amazon's EC2 c4.xlarge instance of AWS. We have tuned the test performance of test-case and server-application. The configuration used in load test-cases is 4000 users and each user requesting each case 10 times. Response time graph for load testing is shown in Figure 3-10 and Table 3-3.

Figure 3-11 shows the output of the routing algorithm. Figure 3-11 (A) shows the places of interest where route is towards the hotel and Figure 3-11 (B) shows the navigation to nearest mosque. We have other services where we use routing such as track friends, nearest money exchange, favorite places.

Summary Report												
Name: Summary Report												
Comments:												
Write results to file / Read from file												
Filename: F:\testSummaryReport.csv												
Browse... Log/Display Only: <input type="checkbox"/> Errors <input type="checkbox"/> Successes <input type="checkbox"/> Configure												
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes			
isUserNameUnique	4146	17509	0	48810	8030.76	63.34%	1.4/sec	2.13	1517.8			
Login	3872	7473	0	145916	11132.15	6.56%	1.4/sec	0.88	653.0			
getUser	3638	2089	0	22834	4339.58	81.34%	1.8/sec	3.08	1712.5			
updateLocation	582	4427	0	20623	5453.55	55.91%	7.5/sec	9.83	1339.1			
editUser	194	1485	0	14007	3489.66	84.02%	3.1/sec	5.65	1866.4			
getNumberOfFreeSMS	18	0	0	1	0.23	100.00%	1.8/sec	3.73	2089.0			
getMchAdHabs	3588	7013	0	60499	7588.37	29.56%	1.7/sec	2.51	1527.6			
getLanguages	3545	7357	0	86188	13885.43	90.01%	3.4/sec	6.51	1988.1			
getMchAIs	3329	39695	0	171423	27595.32	99.85%	3.2/sec	15.20	4890.9			
refreshToken	2852	29121	0	139106	27265.85	100.00%	3.3/sec	3.67	1147.9			
logout	1174	10342	0	100151	15348.91	23.08%	1.4/sec	1.15	847.3			
TOTAL	28946	14495	0	171423	19877.54	63.27%	8.9/sec	15.94	1830.3			

Table 3-3 Load testing summary report

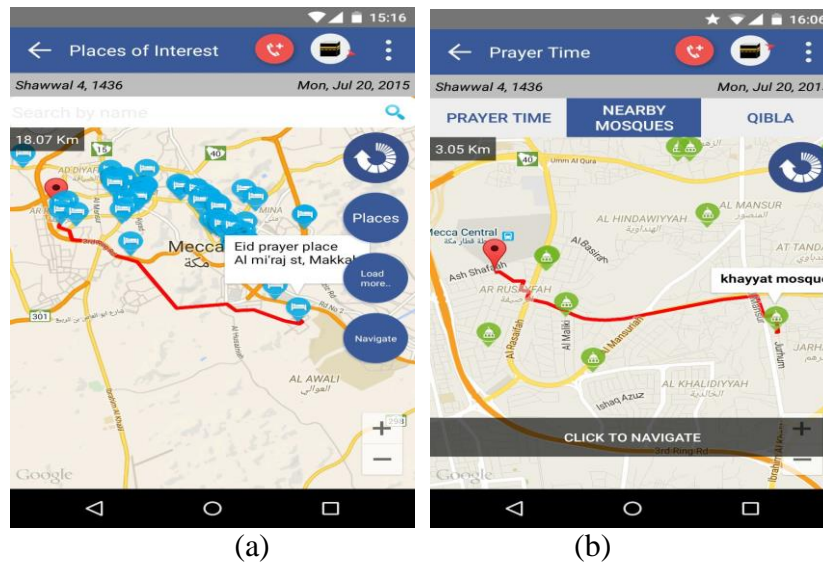


Figure 3-11: Algorithm output for a) finding points of interest and b) nearby mosques

3.6 Conclusion and Future Work

This chapter presents a framework which collects data from different sources in lack of physical infrastructure environment. These sources include location-based services, an incentive-based traffic update module, and geo-tagged social networks. The system extracts information from social networks and sensor-based data. It also extracts different constraints related to traffic flow and average crossing time for a given intersection. It then recommends the optimized path over dynamic road conditions. This chapter also explains the grid-based algorithm on the basis of which optimum paths taking the least possible time are recommended. Presently, the algorithm has been implemented using a fixed grid approach.

Moreover, with the advancement in Internet and mobile technologies, social networking platforms such as Facebook and Twitter have become popular modes of communication. They allow users to share a spectrum of information, including spatio-temporal data, both publicly and within their community of interest in real-time. Scrutinizing knowledge from different types of available, rich, geo-tagged, and crowd-sourced data and incorporating it on a map has become more feasible. This presents a real opportunity to enrich traditional maps and enhance conventional spatio-temporal queries with the help of different types of data extracted from a variety of available data sources.

Chapter 4

Building Smart Maps from Heterogeneous Data Sources

4.1 Introduction

With the expansion of technology for sensors, smartphones and the Internet of Things publicly available data is increasing with a rate of 30% annually and will continue to grow with further advancements. Data from multiple sources can improve the coverage for providing relevant knowledge about surrounding events and points of Interest (POIs) [14]. Social network data can provide fruitful insights and complementary sources of information.

These data can be used in various scenarios, such as emergency situation, inferring events in cities, and showing breaking news. Digital maps nowadays are enormously use with the aim of sharing preeminent information about current locations and spatial characteristics of surroundings. In spite of generating dynamic layers about traffic updates, such as traffic jams, accidents, and congestions, big giants like *Google*, *Yahoo*, *MapQuest* and *Bing* still lacks in providing knowledge about statistical trends, ongoing events, and POI semantics and ranking from crowdsourced data. Such knowledge can be extracted and enriched from heterogeneous available data sources. The existing traditional maps can be augmented with different knowledge-based layers to know more about surroundings and to enhance existing spatio-temporal queries (see Figure 4-1)

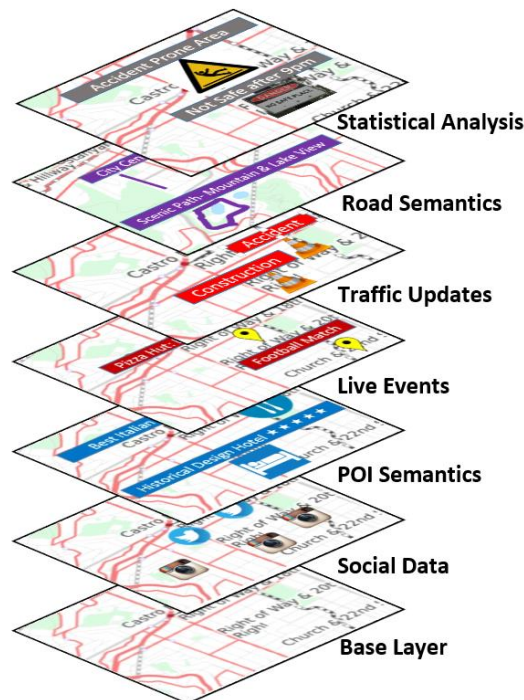


Figure 4-1: Dynamic layers of smart maps

In this chapter, we present a framework that collects and integrates data from different sources including crowdsourced data (social data including Twitter and Yelp), Open Street Map (OSM) data, and other online data (Google traffic, Open government). The proposed system applies different mining and clustering techniques on the available data. Smart maps are self-updating intelligent maps. The updates are based on information extracted dynamically from heterogeneous data sources including social media streams, crowd-sourced data, sensors and online news sources. Smart maps can identify new points of interests, events, or findings and discover new content automatically that were not precisely entered to the map. We believe that smart maps are the next generation of digital maps by providing awareness about surroundings, such as events, traffic updates, road semantics (e.g., scenic or safe path), POI semantics (e.g., fast food restaurant), online offers, and statistical analysis (e.g., dangerous areas) as illustrated in Figure 4-1.

The features of Smart maps on top of existing maps are as follows: 1) *Events of Interest (EoI)*: live events in cities, such as concerts, football matches, jobs hiring and other relevant information (e.g., pizza hut discount, sales in CityMax) are displayed at different levels of abstraction; 2) *Statistical Analysis*: illustrates analysis of (un)safe areas by extracting knowledge on accident prone areas, safe or polluted areas; 3) *Traffic updates from social data*: shows information about traffic constraints, such as accidents or road blocks collected from social data; and 4) *POI Semantics*: describes semantics and ratings of geographical places from crowdsourced data (e.g., using Yelp data to judge quality of POIs, such as best Italian pizza, historical design hotel, etc.).

This chapter serves to: a) collect, store and clean structured, semi structured or unstructured crowdsourced data. This includes digesting microblog social data (Twitter, Yelp), OSM, online data in real-time (Google Traffic APIs) and other historical open government data (Crime and accident data); b) design a common schema to resolve data conflicts and integration issues of social data, and to increase the conciseness and correctness of data; c) extract relevant knowledge by applying state-of-the-art text mining techniques and correlation to find Events of Interest (EoI); d) visualize smart map layers in an interactive way, and f) taking leverage of smart maps to enhance spatio-temporal queries. The following three applications used the concept of the smart maps framework:

- City Explorer: provides up-to-date information about historical places, touristic places, dining, ongoing and upcoming events, shops, news, live social feeds, weather updates, traffic updates and semantics of points of interest collected from numerous sources.
- Routing Service: It takes into consideration not only the traditional spatial and temporal data analysis, but also the safe areas to avoid accident and crime prone places and recommends an optimized trips and paths to the users
- Multimedia Routing Mobile Application: MM routing is an enhanced routing framework that leverages geotagged multimedia data such as images, audio, video, and text, in order to add semantics to the conventional spatial queries. MM routing framework collects, stores, and spatially tags multimedia data shared by users through social networks or through our developed mobile application. The system then uses such data in order to enhance the conventional routing services by resolving existing usability issues and by providing semantics to the routes in terms of enriched points of interest while taking dynamic road conditions into account.

The remainder of this chapter is as follows. Section 3.2 discusses the related work from different perspectives. Section 4.2 introduces an overview of our system architecture. Section 4.3 highlights the implementation and map visualizations. Section 4.4 discussed multimedia routing mobile application; and Section 4.5 draws conclusions and future challenges.

4.2 High-level architecture

We present a smart map building framework that retrieves data from multiple sources, processes that data in order to find knowledge-based layers and visualizes those layers on maps. This framework collects microblog social data streams, open government data, and online data under one platform in order to provide relevant knowledge to users. This helps users in understanding their surroundings, such as live or upcoming events within cities, traffic accidents, road constructions, temporarily closed paths, POI semantics, as well as offers and discounts. Figure 4-2 shows an overview of our proposed smart maps architecture with the salient components.

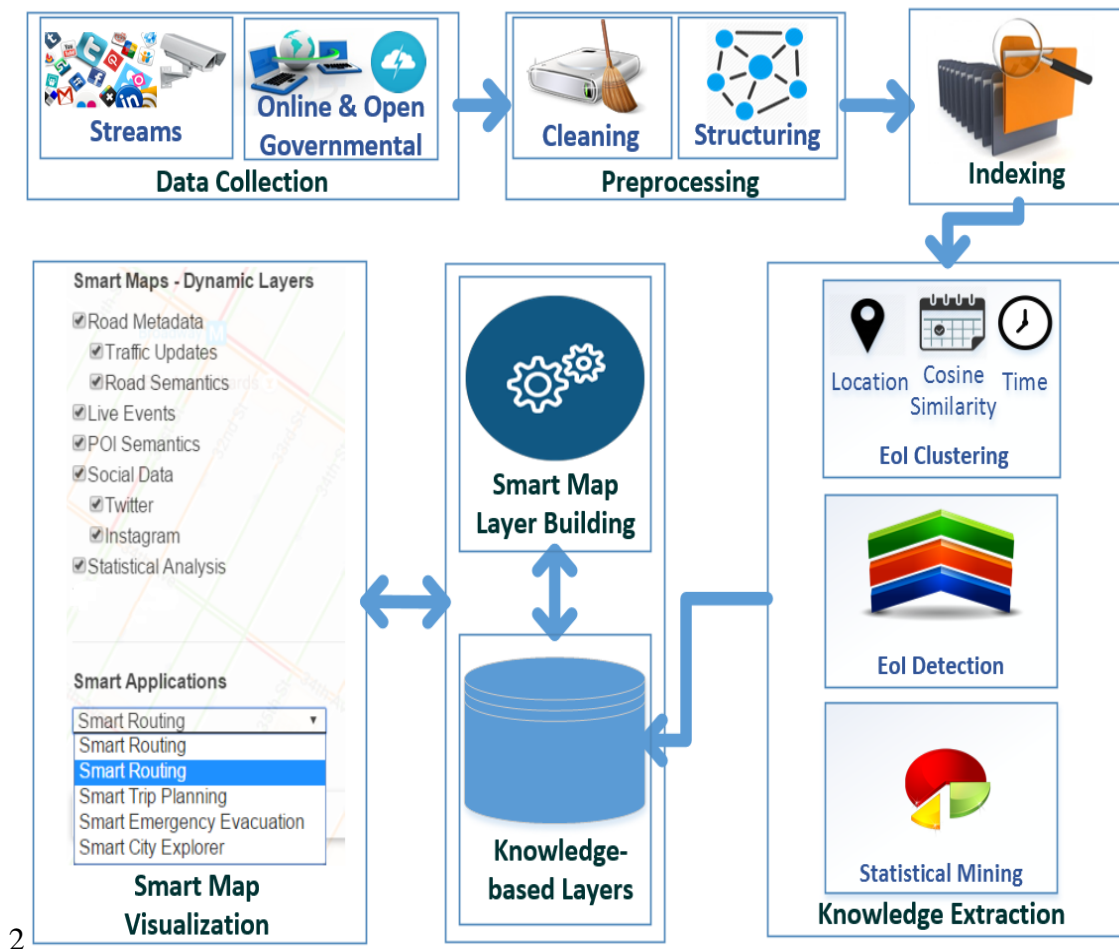


Figure 4-2: Architecture of the smart maps framework

4.2.1 Data Collection and Management

Data is retrieved from different data streams, available APIs and online web services. Multiple techniques and policies (frequency, format, structured or unstructured) should be applied on each source of data to be collected. Following are the three ways that we have used to collect disparate data from multiple sources:

- **Data chunks:** In data chunks mode, we download files from different source links that contain partial or full datasets. This data can then be imported or converted into another format for further processing (e.g., crime and accident statistical data). For our use case, we used datasets provided by authorities or open data for different cities, such as New York, USA (including area boundaries, transportation networks, geological data, resources etc.), and other crowdsourced data from Open Street Map and Yelp.

- **Single Query:** In a single query mode, we have used an interface to fetch data by using a single query. This can be achieved by using Restful API to retrieve data in XML and JSON (e.g., Google Traffic and Weather API).
- **Continuous Query:** In a continuous query, we run Taghreed[5] crawler that collect streams of data for each data source (i.e., Twitter and Flickr). Data in this mode is retrieved through Restful APIs of social networks. This mode requires specific handlers for each source of data.

```
{"header":{"time":"Fri Jan 01 00:39:11 +0000
2016","geo":{"type":"point","coordinates":[34.84863834,-95.54509333]},"source":
"twitter","eventType":["Party"],"type":"text","potentialEvent":"true",
"eventName":"Social events/party","country":"United States","city":"Oklahoma"},
"payload":{"tags[]","id":"2605873770","followers":"581","title":"","text":"Party @
Stacey's ☺","viewers":"0","screenName":"TheLedgenBears","language":"en",
"displayName":"Ledgen","url":null}}
```

Figure 4-3: Sample Data Packet from Twitter Streams

4.2.2 Preprocessing

In the pre-processing step, data cleansing is required by removing noise and irrelevant fields and, in case of social data, converting semi- or unstructured data into a structured format referred to as packets. Each packet has a meta-data header, containing source, location, time, type, potential event and event properties, and a payload, containing the actual content (see Figure 4-3). Based on the list of predefined event corpus database, useless data is discarded and potential packet marked as true. For statistical data, we clean them by removing irrelevant fields. For EoI detection, we used NLP techniques for tokenization and to identify Part-of-Speech by taking into account stop words, out of vocabulary words and other abbreviations, such as *'i know you' (iky)*.

	yelpid character varying(100)	poiiname character varying(100)	twitteraccount character varying(40)
1	bryant-park-new-york-2	Bryant Park	bryantparknyc
2	red-dawn-combat-club-fresh-meadows-4	Red Dawn Combat Club	RedDawnBJJ
3	queens-botanical-garden-flushing	Queens Botanical Garden	queensbotanicl
4	harlem-yoga-studio-new-york	Harlem Yoga Studio	harlemyoga
5	gleasons-gym-brooklyn	Gleason's Gym	Gleasonsboxing
6	stone-street-tavern-new-york	Stone Street Tavern	stonesttavern
7	the-roval-palms-shuffleboard-club-brooklyn	The Royal Palms Shuffleboard Club	RoyalPalmsClub
8	the-fashion-class-manhattan-2	The Fashion Class	TheFashionClass
9	brother-jimmys-bbq-new-york	Brother Jimmy's BBQ	BrotherJimmys
10	the-bahche-brooklyn-4	The Bahche	The Bahche

Figure 4-4: PostGIS screenshot to demonstrate data integration of POIs from Yelp ID and Twitter Screen Name

4.2.3 Indexing

Pre-processed data needs to be stored and indexed for further integration and clustering. Spatio-temporal indexing schemes for efficient retrieval of queries are implemented. As we are dealing with geo-tagged data for the whole world, we propose a hierarchical data structure (similar to a partial quad tree [105]) that helps in an efficient processing and clustering by comparing packets within leaf cells (i.e., nearby geotagged data packets). This approach divides the world into cells at different levels of granularity based on the number of data points. Geo-tagged streams, Yelp, OSM and statistical data pieces are tagged within a particular *cellID*. We design two types of indexers: 1) a spatial geohash indexer for spatial raw data packets and; 2) knowledge-based indexer for extracted events and layers data.

4.2.4 Knowledge-based Layers

Extracted knowledge is then spatio-temporally indexed in a knowledge-based layers database. Each EoI is tagged with the cell Identifier, and pieces of statistical analysis are associated with segments or nodes of the road networks. For temporal aspects and cleaning of expired EoIs, we took two parameters: a) ‘*birth time*’ that indicates the first existence of the event in our system, whenever we calculate the first cluster of packets related to that event; and b) ‘time of occurrence’ that marks the actual time the event occurs (e.g., next Monday). We clean the EoI from our data after the ‘time of occurrence’ has expired using a combination of piggy back and periodic approach. Periodic verification is used to check expired events periodically, whereas piggy back approach is to check for expired events whenever we get any new event within the same cell in the hierarchical tree.

4.2.5 Knowledge Extraction

Text mining and clustering techniques are used to find Events of Interest and to infer statistical analysis, such as accident and crime prone places. We extract time, location and identify text similarity to detect the type of EoI. For statistical analysis, we find nearby roads and points of interest that are unsafe (i.e., accident prone and hot crime zones identified by a certain threshold). We have also divided the knowledge extraction module into the following three levels:

- Direct Detection: At this level, data from trustworthy or authorized sources are considered. We can find authentic social accounts or feeds related to traffic, incidents, etc., in order to classify POIs and to detect social events from trustworthy sources. Most of the announcements related to EoI, such as offers, discounts, upcoming and ongoing events, are published by PoI owners, so it is important to identify authentic sources in social data so that these events can be reported. As illustrated in Figure 4-4 and during the preprocessing phase, the system identifies Twitter screen names and Yelp identifiers of trustworthy sources by using location and string matching techniques between social data and POI data collected from Yelp and OSM.

- Indirect Detection and Mining: The second level is to apply mining techniques on fused data from multiple sources in order to extract EoI semantics from unspecified happenings. This level is more complex as compared to the first level, as here we need to consider a truth probability model. Our system adopts the graph analogy where each potential event stream is considered as a 'node' and the value of 'cosine similarity using term frequency - inverse document frequency (TF-IDF)' between streams as a weight of the bidirectional 'edge'. Data packets with a high text similarity value are clustered using the DBSCAN clustering algorithm. The DBSCAN is suitable in our approach as, unlike most of the other clustering methods, it does not require a prior knowledge of the minimum number of clusters. It can help to detect unspecified events and the hot topic detection as well.

- Statistical Detection and Mining: The third level is performed on historical open governmental data. We apply mining and detection techniques in order to extract statistical analytics, mainly, accident prone areas and crime hotspots. Clustering of crime and accident data is performed within each cell of the hierarchical index tree, and based on a predefined threshold value, edges and POIs inside that area are marked as a crime/accident prone area. We use this approach to find safest path (see Section 4.4).

4.2.6 Smart Map Layer Building

This component is used to fetch index events from main memory and/or disk. It has two main components; a) query optimizer and b) query engine. The main task of the query optimizer is to find the best query plan. It handles predefined queries related to existing layers displayed on map. The query engine retrieves the plan from query optimizer, so that sub-queries to fetch data from the main memory or disk can be performed. Finally, the smart map layer builder accumulates different results and sends it to the visualizer.

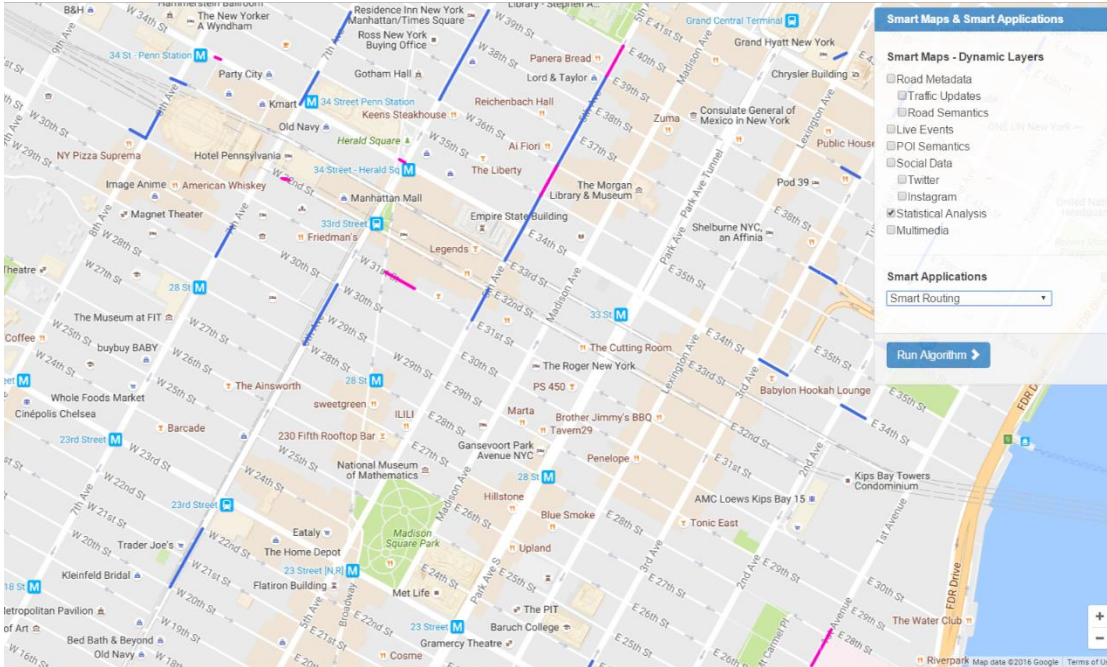


Figure 4-5: Overview of Implemented Smart Maps with Crime and Accident Roads

4.2.7 Generating and Visualizing Smart Map Layers

The visual interface provides a rich set of spatio-temporal dynamic layers on top of existing maps. The visual renderer interacts with the map building engine to run queries including range, k nearest neighbour K_{NN} , aggregated and routing queries. Figure 4-5 and Figure 4-6 illustrate the visualization of implemented smart maps with POI semantics (displayed as marker), traffic updates and statistical information (blue and pink thick colour lines demonstrating accident prone and crime prone roads)

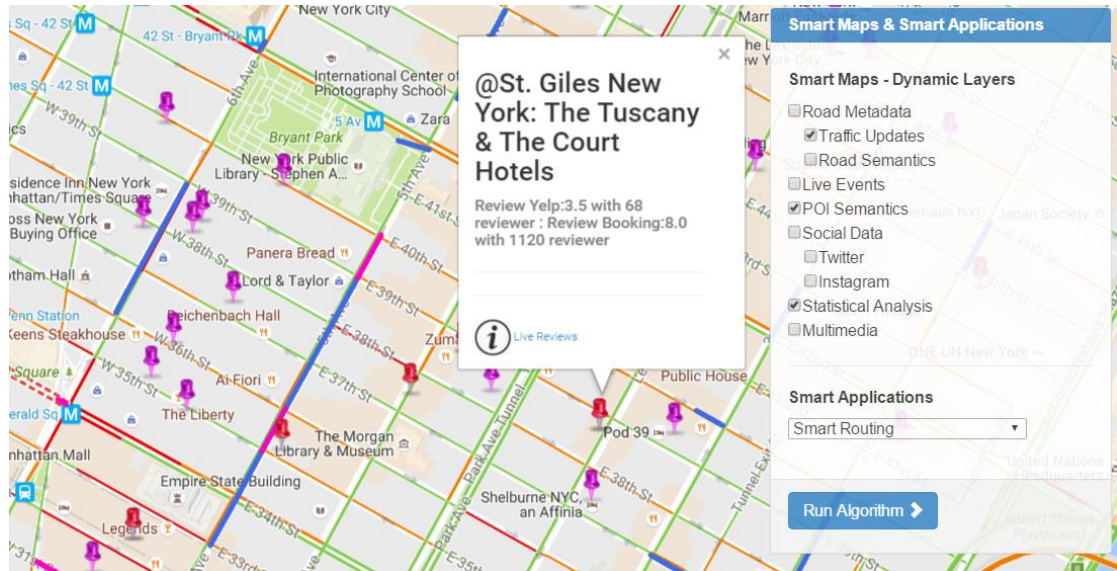


Figure 4-6: Overview of Implemented Smart Maps with POI Semantics

4.3 Implementation

To validate our approach, we have developed a prototype based on 30Million+ geotagged tweets of world and Yelp, Booking.com, Open Statistical Data from USA governmental website, OSM road network and OSM POI list of New York, USA. The front-end is a web-based application that is used to visualize *smart maps* with dynamic layers and to provide an interface to perform the queries related to enhanced routing and city explorer. We are using i7-4712 HQ-CPU @ 2.30GHz with 16GB DDR-2 RAM for in the back-end for processing using the following libraries and software:

- **osm2pgsql:** The Open Street Map component contains planet dump data that is converted into PostGIS datasets by using the osm2pgsql [106] tool.
- **PostGIS:** We installed PostGIS for spatial query over PostgreSQL to store road network data, POIs extracted from Yelp, Booking.com and OSM with their semantics.
- **NLP:** We used Ark-tweet-NLP [107] library for part-of-speech and annotation. This library is trained for Twitter and produce better results than Stanford NLP. It takes care of the out-of-vocabulary words, and stop words used in Twitter.
- **Clustering Algorithm:** Data packets with a high text similarity value are clustered using DBSCAN[60] clustering algorithm. The DBSCAN is suitable in our approach as, unlike most of the other clustering methods, it does not require a prior knowledge of the minimum number of clusters.
- **Taghreed Crawler [5]:** The same crawler was used to collect geotagged tweets.

- OsmPoisPbf: We used OsmPoisPbf [108] tool to extract POIs from OSM in PBF file format.
- Other libraries: Apart from above, we used Jackson JSON and Yelp, and the Brezometer weather REST APIs in our system. OSMPgRouting[109] A* algorithm used to calculate the path and we updated edge values using google maps API in case of traffic and increase the weight of edges based on crime and accident frequencies. Back-end implementation is done in Java 1.7.

4.4 Application on smart maps

Current routing queries are limited to show shortest path or fastest path such as on Google, Yahoo maps. This section describes importance of smart maps framework to enhance existing spatial queries. This approach lays the ground for delivering different intelligent services and applications, such as: 1) city explorer that provides latest information collected from multiple sources about places and events; and 2) route and trip planning that leverage smart map framework to recommend safe routes. Following are the “City Explorer” and “Routing” applications as demonstrated on top of the smart maps framework.

4.4.1 City explorer

The city explorer query allows users’ to explore city on or without maps. Lots of mobile and desktop application are available that provides details about city explorer. We enhance maps to demonstrate dynamic layers on the existing statistical maps. These dynamic information about events, POI offers and details are interested to end users’.

Figure 4-7 illustrates an example output by showing different type of EoIs which includes a) ‘Sports League’; b) ‘Graduation Party’; c) ‘Traffic Incident’; and d) ‘Jobs Opening’. This output allows us to show live events that provide the latest information collected from multiple sources about historical places, touristic places, dining, events, shops, news, live tweets, weather updates, traffic updates, semantics of points of interest, and visualize multimedia information that enhance the system usability. Our smart maps framework can help the end-user by providing different knowledge about city in decision making by using interactive visualization that graphically presents EoIs on maps.

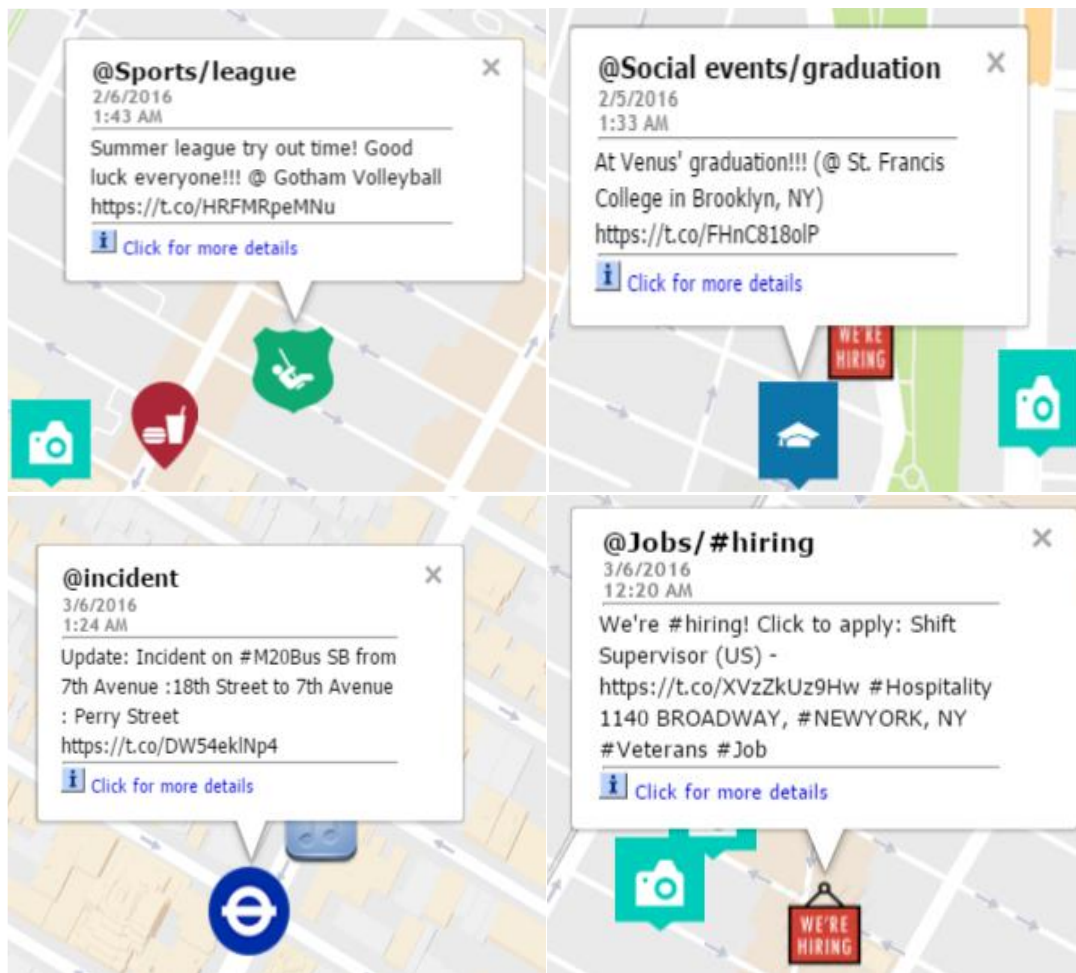


Figure 4-7: Enhanced city explorer on smart maps: Different type of EoIs on 3rd June: sports, graduation party, road incident, jobs hiring (from top left, top right, bottom left and bottom right)

4.4.2 Safe Routing

For a given spatio-temporal area, we use the *smart maps* system to enhance traditional queries. We grouped extracted EOIs including social gatherings, 'hotspots' and 'hot times' in a criminal activity, POI semantics, accident-prone roads, in order to provide enriched response of users' routing queries. For example, consider a city dweller who is interested in current ongoing activities in her neighborhood district and she posts the following query, "Show safe and fastest path to find the nearest music concert with some ticket discounts". Currently, existing maps are unable to provide answers to this type of queries.

Keywords	Layers	Sources
Safe	Accident (Prone Edges)	Open Government Data
Safe	Crime Hotspots	Open Government Data
Music Concert	Social Events	Social Data and News
Discounts	POI Announcements	Social Data

Table 4-1: Relation between Keyword of User's Query, EoIs and Sources of Data Collection

Smart maps framework can support such queries, by determining the relationship between the list of keywords in that query, and the corresponding layers and sources of data (cf., Table 4-1). Figures Figure 4-10 shows the results of the query: a) Figure 4-8 shows a map with crime (pink color) and accident prone (blue color) edges; b) Figure 4-9 shows an optimized path with respect to time between two markers (showing with red color), c) Figure 4-10 shows a crime free path by avoiding crime prone edges (light green color), d) Figure 4-11 shows the safest path that can be calculated by avoiding both accident prone and crime prone edges (showing with blue color).

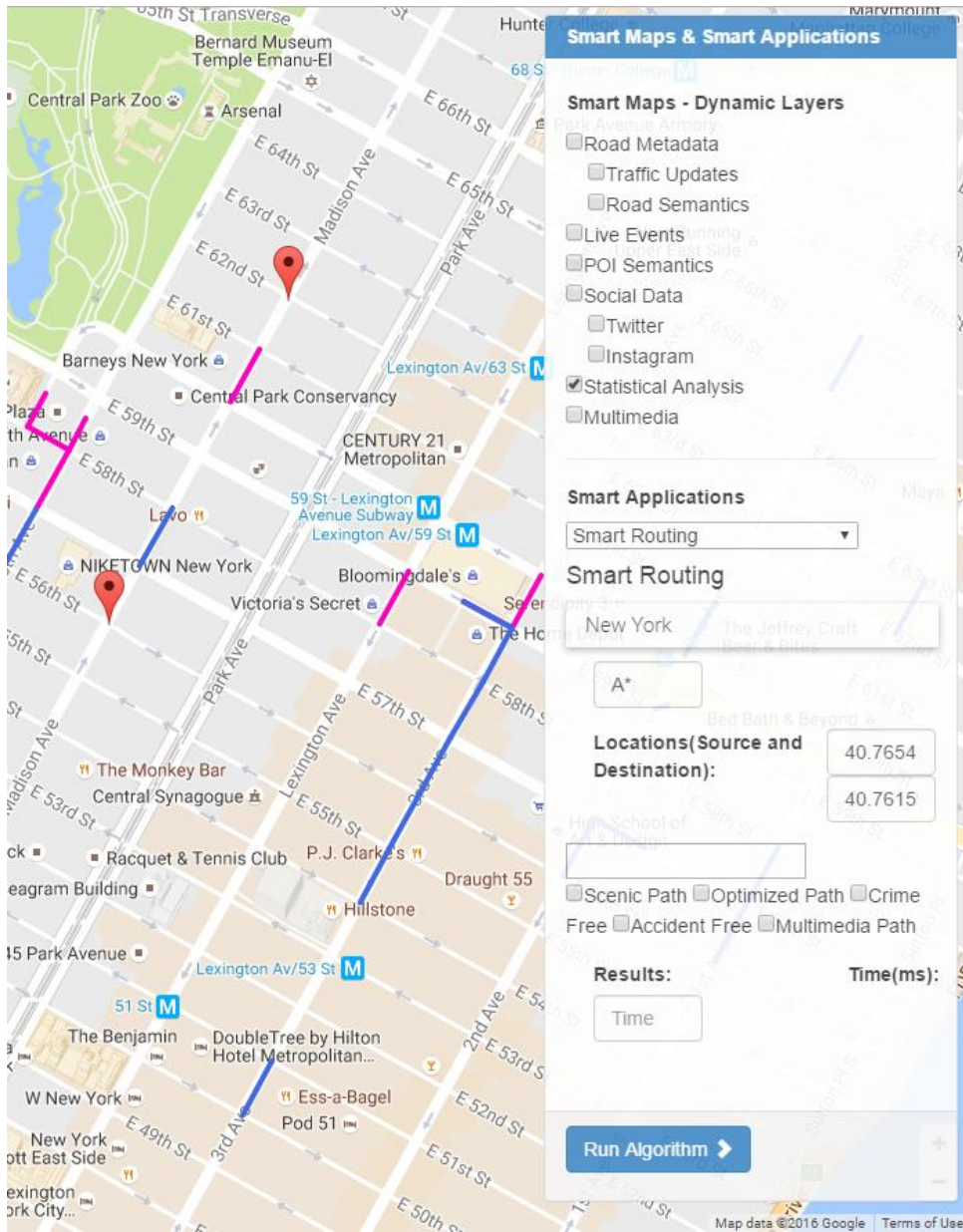


Figure 4-8: A Map with Crime (pink color) and Accident Prone (blue color) Edges along with two Red Markers to Denote Starting and End Point

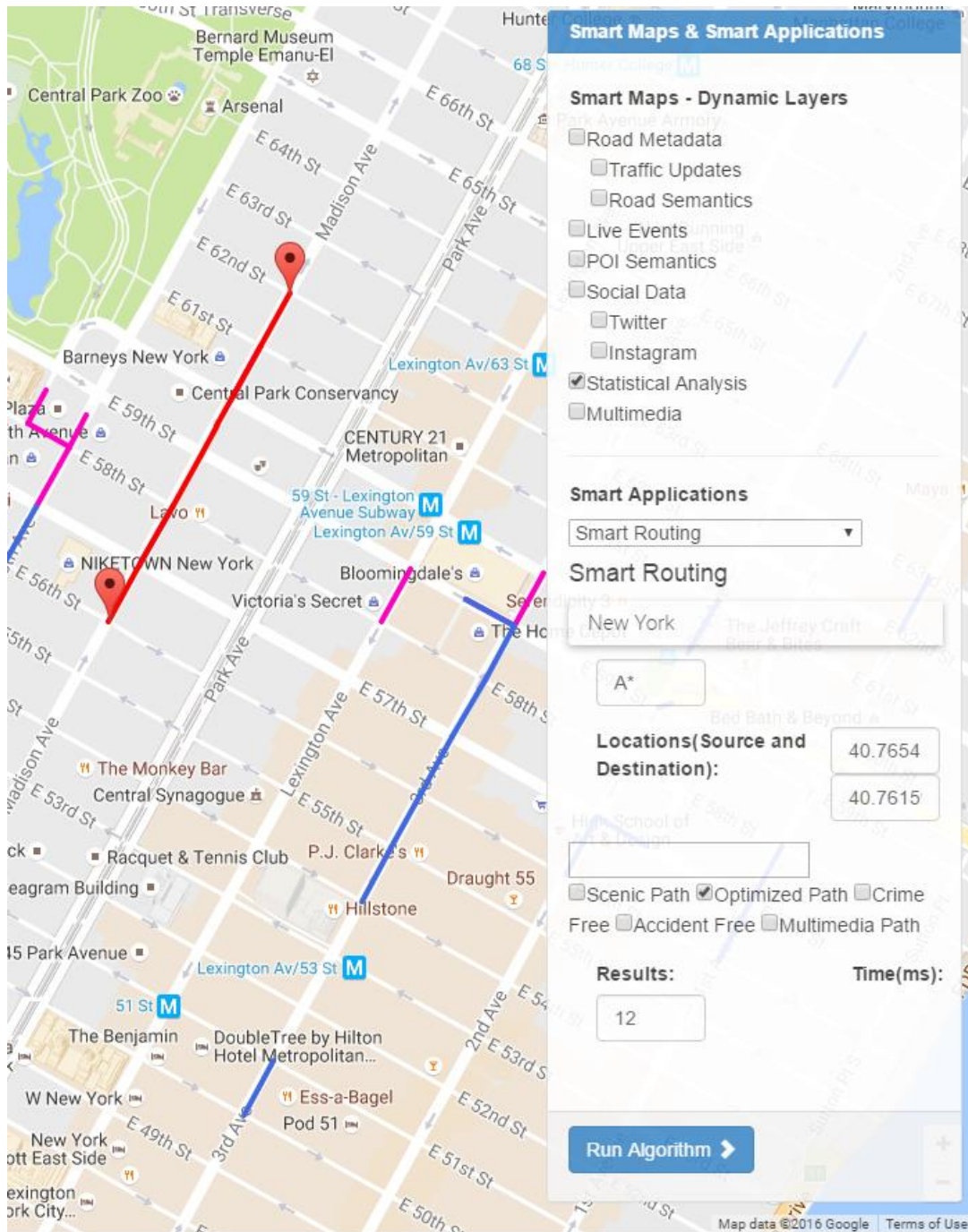


Figure 4-9: An optimized path with respect to time between two markers (showing with red color) without considering crime and accident prone edges

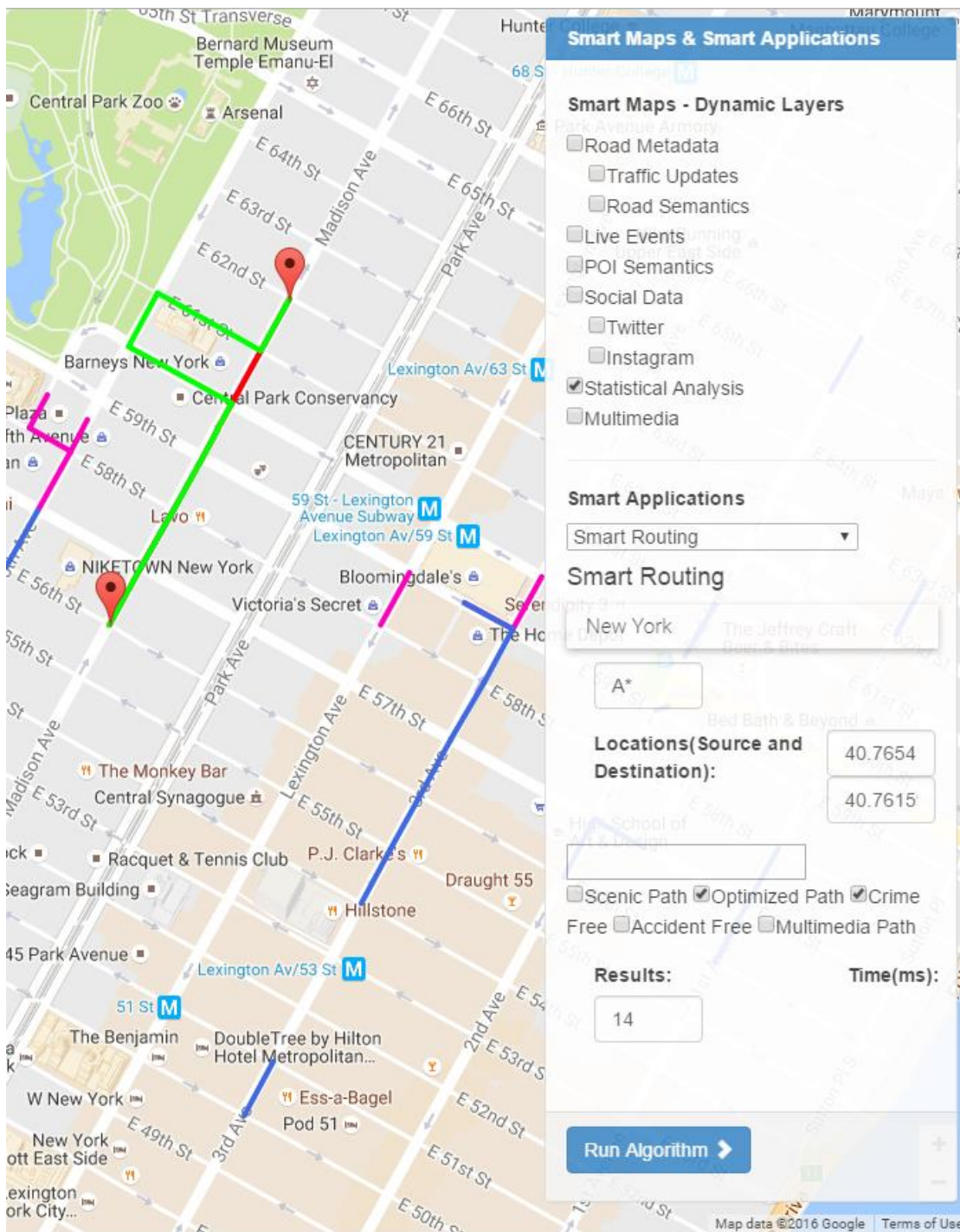


Figure 4-10: A crime free path by avoiding crime prone edges (light green color) without considering accident prone edge.

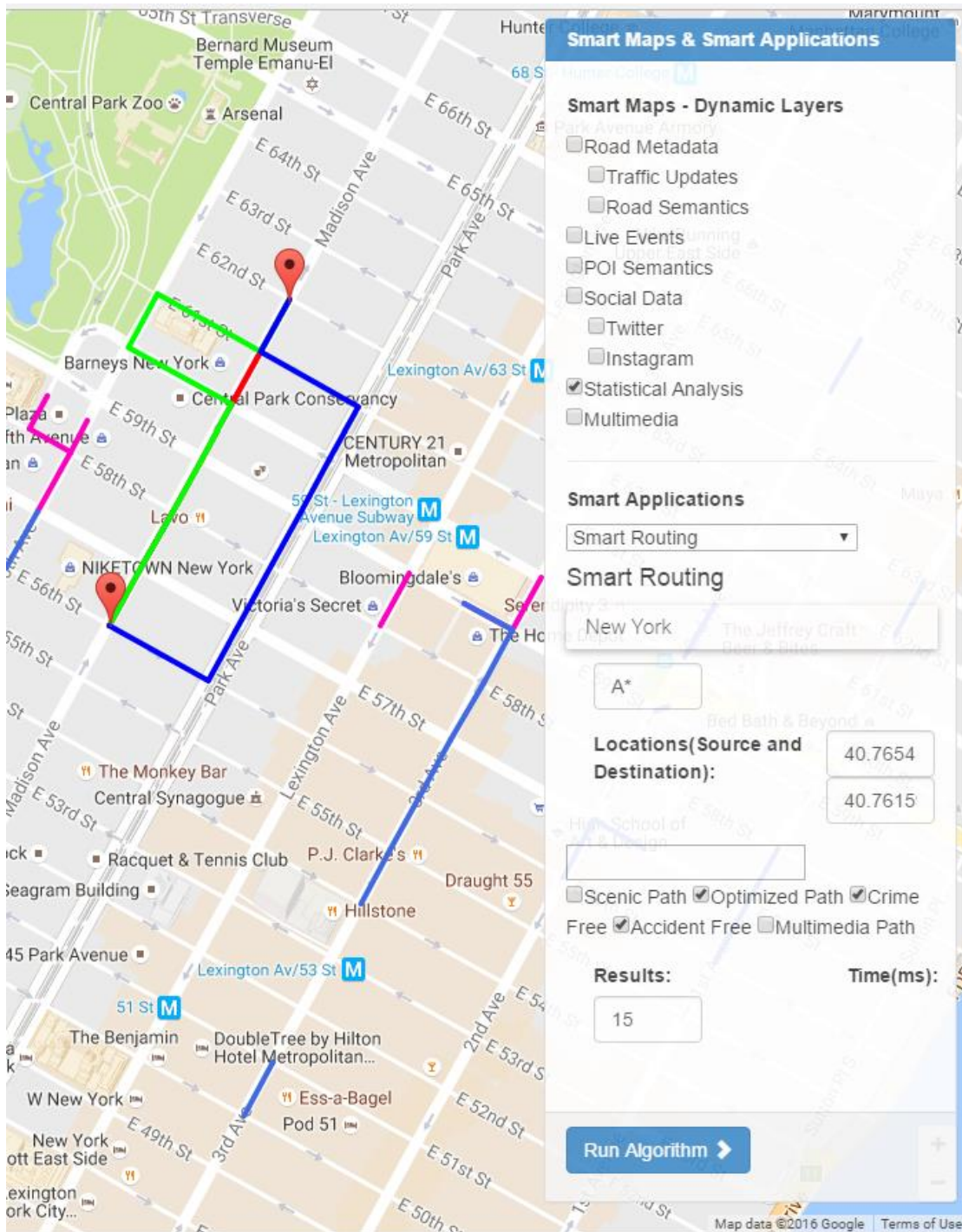


Figure 4-11: Safest path calculated by avoiding both accident prone and crime prone edges (showing with blue color).

4.4.3 Multimedia Routing - Mobile App

Social networks such as Facebook, Twitter, and Instagram are among the world's top visited mobile applications. Thanks to the mobile technology advancement, intuitive ways are available to share a variety of information including geotagged multimedia data within users' communities of interest (COI) or publicly in real-time. For instance, out of millions of tweets that are posted per second, a large number of tweets is geotagged. Analyzing geotagged data can provide additional semantics to spatial queries such as shortest path queries, range queries, and k-nearest neighbor queries.

In unplanned and densely populated urban areas where multiple road endings concur at a single point or at a small road segment, conventional routing techniques that propose a static path towards a destination become less efficient and difficult to be followed by inexperienced users. Many users, especially elderly users that are navigating in heavily crowded environments find it difficult to use conventional routing services to locate points of interest (POI) or individuals. More than 3 million pilgrims perform Hajj¹ every year with multi-disciplinary culture and language, and majority of them don't have any prior experience of the Makkah, Saudi Arabia [102]. It leads to a unique challenge to conventional routing services through smartphones using maps. Hence, finding points of interest or lost individuals in a large crowd poses a great difficulty for event organizers, pilgrims, governments, ministries, health industries, emergency departments, and family members. Moreover, the number of pilgrims sharing geotagged making crowdsourcing a reality. We assume that adding semantics in routing with the help of multimedia can make it easier multimedia data, which carries semantics, have increased manifold, for mobile users with limited or no prior experience of map-based routing techniques. In this section, we present a novel multimedia-enhanced spatial querying framework that leverages geotagged multimedia data such as images, audio, video, and text, in order to add semantics to the conventional spatial queries. For instance, in case a user submits a query to find a hotel, the system will show live results from all nearby hotels with dynamic information such as availability, charges, public and private parking, customer reviews, and traffic constraints. Multimedia data is collected from social networks and through our developed mobile application as shown in Figure 4-12. Our system stores data in different resolutions and spatially tags data using a spatial grid index as explained in section 4.4.3.2. In this context, we present a novel way to find individuals using geotagged multimedia data. Users can share geotagged data so that the system can recommend paths by showing multimedia-enriched POIs as well as traffic updates.

The term "Geotagged multimedia" is not innovative in the state-of-art. It has been widely used in various scenarios, however, using such data to add semantics to routing services is a novel approach. For example in [31], geotagged tweets are used to identify traffic constraints such as accidents and road closed; whereas in [82], a semantic algorithm is used to recommend interested POIs based on data collected from Foursquare and Instagram. The authors in [4] present a landmark-based navigation mechanism to collect the POIs based on attractiveness in order to identify the routes. In contrast to the works presented in [2, 3, 4], our system differs by: 1) collecting the source and destination of the route from the geotagged multimedia information submitted in real-time for route discovery; 2) showing the publicly available POIs with multimedia data associated with it within a certain radius of the calculated route, in order

to semantically help the users in discovering their environment; and 3) returning the result based on user smartphone's bandwidth and resolution.

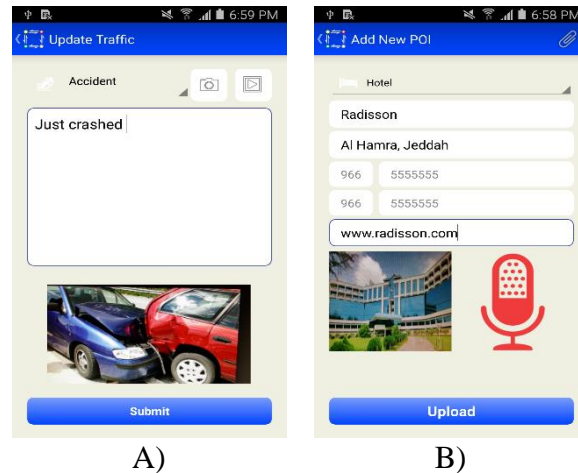


Figure 4-12: Multimedia geotagged data collection through our developed mobile application. A) Multimedia-enhanced traffic update; and B) Adding multimedia-enhanced points of interest

The remainder of this MM routing mobile, we presents an overview of the system architecture and discusses the implementation issues. Later, we highlights different demonstration scenarios, and finally draws some conclusions and future challenges.

4.4.3.1 Architecture

Figure 4-13 shows the high-level architecture of the system. The system augments conventional routing with multimedia information integrated within the spatio-temporal query engine. The system uses road networks and aggregated geotagged multimedia data for calculating the optimal routes. An overview of the different components is described as follows.

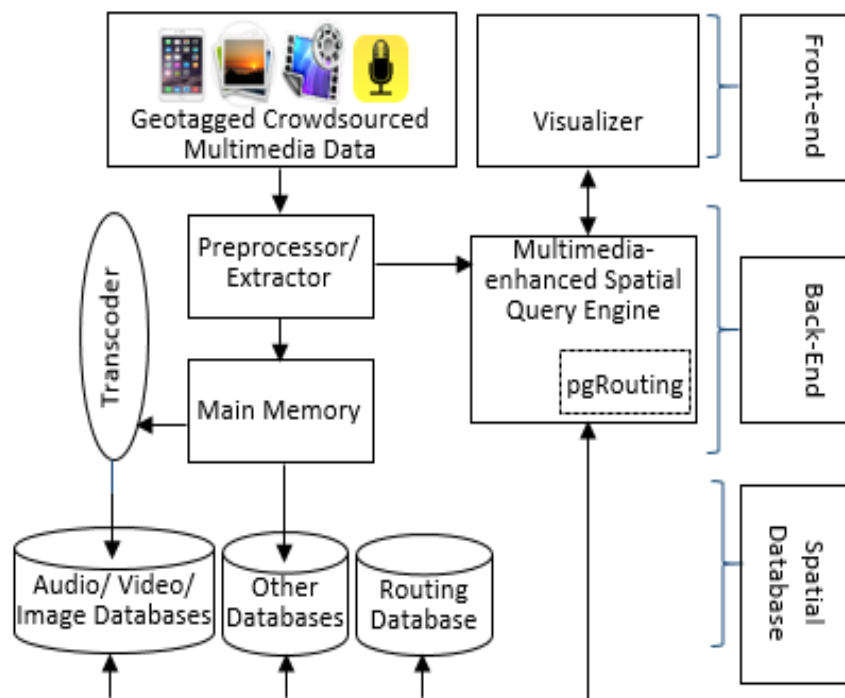


Figure 4-13: High-level architecture

User choice, accidents and roadblocks are some of the constraints taken into account. The system accumulates both intra- and inter-user shared geotagged media, for example, the destination, metadata of the multimedia, date and time and multimedia payload (audio, video, image, text, location, time and user profile), as well as from crowdsourced or publically available geotagged data from social media. Data is first received by the pre-processor, which extracts user's location and the associated multimedia parameters. The query engine and the main memory respectively receive the input data for spatio-temporal queries (see Figure 4, 5 and 6). Pre-processed data is then temporarily stored in the main memory. The flushing process then transfers the data, on the basis of its threshold value, to the spatial database from the main memory. The multimedia parameters are transformed into various resolutions by the transcoder to support different user bandwidth and resolution. The original and transcoded data is finally stored in the spatial database. This spatial database stores routing data used by pgRouting for spatial queries, the multimedia data including the geotagged audio, video and images in different resolutions. In other databases, we have stored the resolution type, data type, data source, and the extracted multimedia data. Each record in the multimedia data table is labeled with nearby edge ID and cell ID. The multimedia-enhanced spatial query engine processes the spatio-temporal queries by fetching the routing information, the user constraints, such as bandwidth and resolution, the media preference and the data type from the pre-processor. It then generates the dynamic route, augments geotagged multimedia parameters with the appropriate resolution and shares the resultant route to the visualization interface (see Figure 3). Online and offline results of multimedia enhanced spatio-temporal queries are displayed on maps by the visualizers.

4.4.3.2 Implementation

We have developed front-end applications for smartphones running on Android 4.4.x (KitKat) and 5.x (Lollipop) –with the capability of sharing and receiving multimedia geotagged data. The front-end of the system communicates with the back-end by sending the parameters in JSON format using HTTP REST API. For the back-end, we have used Amazon Web Services (AWS) framework with EC2 c3.4xlarge machines for processing data from different sources such as publicly available geotagged social media data and user generated geotagged multimedia data through our developed application. The advantage of using AWS is to deal with a large crowd by scaling horizontally and vertically for sharing the load on multiple instances based on predefined latency of the threshold. The transcoder has been implemented using open source FFMPEG library. We extract the road network from open street map [5] for the part of Saudi Arabia, where the number of edges and nodes are 669436 and 597808 respectively. We use osm2pgsql [6] library to convert the open street map data into postGIS-enabled PostgreSQL databases. We compute the conventional spatial queries such as shortest path and k-nearest neighbor using pgrouting [7] in-built Algorithms (see Figure 4-14A, Figure 4-15A and Figure 4-16A) and range query using postGIS spatial queries operators (see **Figure 6A**). The spatial database tables are on a dedicated database server running on PostgreSQL on a c3.2xlarge machine of AWS.

We divide the map spatially using grid approach. Each cell has a fixed size (one latitude degree * one longitude degree) and it stores users id, constraints id, trajectories of the users and extracted geotagged multimedia data such as Audio, Video, Image and text as shown in **equation 1**.

$$\forall Cell_i = \left\langle \begin{array}{l} Users_{1,2,3...U}, Constraints_{1,2,3...C}, \\ Trajectories_{1,2,3...T}, Image_{1,2,3...I}, Audio_{1,2,3...A}, \\ Video_{1,2,3...V}, Text_{1,2,3...T} \end{array} \right\rangle \dots (1)$$

Each cell has a unique id. Each cell has eight neighboring cells, and each cell also store the id of the neighbor cells as well in the database. Grid approach minimizes the response time to fetch multimedia data for different types of spatial queries.

4.4.3.3 Application Functionality

We developed a prototype that receives user-generated data through our smartphone application and twitter data. Our geotagged social network data crawler collects data from Twitter for Saudi Arabia. Saudi Arabia ranks first in Twitter with an average of 150 million tweets/month. Our system currently collects only geotagged tweets, extracts multimedia data and stores it in the database. The following four scenarios were selected to show the system in providing multimedia augmented spatial queries.

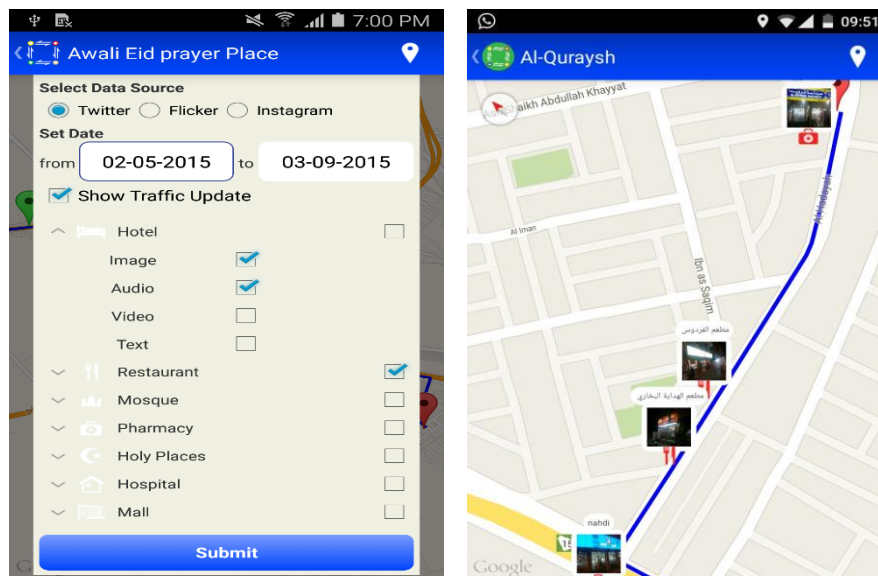


Figure 4-14: A) Select multimedia sources, POIs and date, B) Multimedia-enhanced routing

Scenario 1-Multimedia-Enhanced Routing: Multimedia enhanced routing gives a new dimension to conventional heavenly used routing algorithm, which helps the user in more accurate and rich manner; for example, take the right turn after the Radisson Hotel (image of the hotel will also be displayed). **Figure 3A** shows the user-defined preferences for multimedia enhanced routing. In this scenario, destination is known and system displays the geotagged multimedia along with the conventional routing as shown in **figure 3B**. **Figure 4** shows the multimedia enhanced routing query pseudo code for the same.

```

SELECT * FROM mmData JOIN
SELECT seq, id1 AS node, id2 AS edge,
cost FROM pgr_dijkstra(
SELECT gid AS id,
source::integer,
target::integer,
length::double precision AS cost
FROM ways',
30, 60, false, false)
AS route ON mmData.gid = route.gid
Where (mmData.msgType = 'Image' or mmData.msgType =
'Text') and (mmData.source = 'Twitter' or mmData.source =
'OwnApp') and (mmData.poiType = 'SuperMarket' or
mmData.poiType = 'Restaurant') and mmData.date >=
'2015/05/25';

```

Figure 4-15: A) Conventional shortest path query and B) Multimedia-enhanced routing query with images and text related to supermarket and restaurant fetched from twitter and our system from 25-May-2015 onwards.

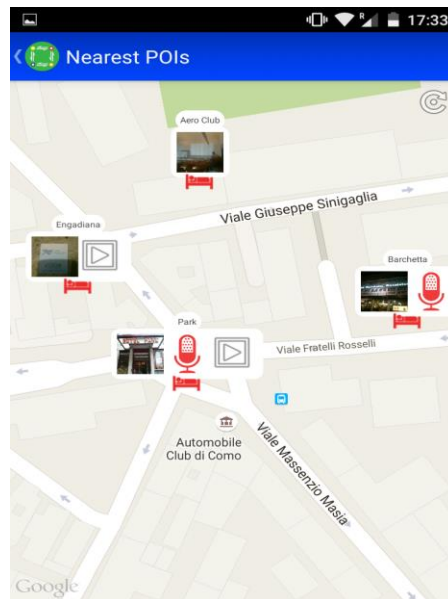


Figure 4-16: K-nearest multimedia-enhanced neighbor query

Scenario-2: K-Nearest Multimedia-Enhanced Neighbor Query: Conventional k-nearest neighbor display nearby points of interest but they do not provide related dynamic information. Such dynamic information is retrieved from the multimedia data that are collected through our application and geotagged tweets from twitter. Figure 4-16 shows the query to fetch k-nearest multimedia enhanced neighbor query, where it fetches the multimedia information regarding the POIs and the traffic constraints. The output generated by the system is shown in Figure 4-16. .

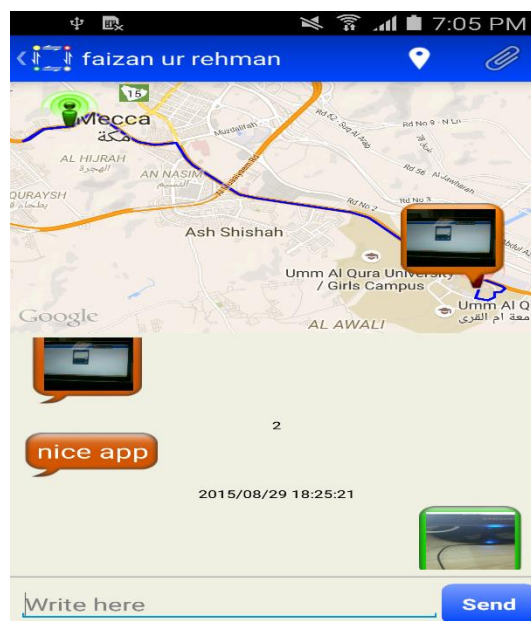


Figure 4-17: Multimedia finding lost individuals

Scenario-3: Multimedia-Enhanced Range Query: Conventional range query solutions display points of interest that are in the given range but they do not provide related dynamic information.

```

SELECT * FROM mmData JOIN
  SELECT seq, id1 AS path, id2 AS edge,
  cost FROM pgr_kdijkstraPath(
    SELECT gid AS id,
    source::integer,
    target::integer,
    length::double precision AS cost
    FROM ways',
    10, array[60,70,80], false, false)
  ORDER BY cost LIMIT 5
AS knn ON mmData.gid = knn.gid
Where (mmData.msgType = 'Image' or
mmData.msgType = 'Text') and (mmData.source =
'Twitter' or mmData.source = 'OwnApp') and
(mmData.poiType = 'SuperMarket' or
mmData.poiType = 'Restaurant');

SELECT * FROM constraints JOIN neighborcells
ON constraints.cellid = neighborcells.cellid
WHERE constraints.type = 'accident' or constraints.type
= 'roadblocks' and constraints.cellid = 23;

```

Figure 4-18: A) Conventional K_{nn} Query, B) Live traffic constraints for nearby area query and C) K-nearest multimedia-enhanced neighbor query.

Such dynamic information is retrieved from the multimedia data that are collected through our application and geotagged tweets from twitter. **Figure-6** shows the query to fetch multimedia-enhanced range query, where it fetches the multimedia information regarding the POIs and the traffic constraints of the nearby area and nearby cells respectively.

```

SELECT * FROM mmData JOIN
  SELECT * FROM poi WHERE
  ST_Dwithin(geog::geography,
  ST_GeogFromText('POINT(long lat)'),
  400) and poi.type = "Restaurant"
AS range ON mmData.gid = range.gid
...
SELECT * FROM constraint
...

```

Figure 4-19: A) Conventional range query, B) Live traffic constraints for nearby area query and C) Multimedia enhanced range query.

Scenario-4: Finding Lost Individuals Using Geotagged Multimedia Data: It is very difficult to track loved ones in a huge crowd such as in Hajj where millions of people (mostly aged) gather in a small place. Elderly people find it difficult to use routing with a conventional map. Multimedia tracking featuring navigation is an alternative solution whereby users can share just geotagged multimedia data with each other. Furthermore, the

system helps the user to display nearby multimedia data and route guidance in an easy and convenient way. **Figure 3D** shows a scenario between patients and doctor where patient is sharing multimedia messages while they are moving towards each other. For patient, it will guide the routing and for doctor; it will further help to identify the exact location of the patient by analyzing multimedia data and thereby saves time to reach to the patient. In **Figure 7**, pseudocode extracts the location from the geotagged multimedia messages and generates the multimedia path. In Line 5 and 6, we are extracting edge Id and cell Id from the path and adding the layer of multimedia data, traffic constraints over the path.

```
Algorithm: Finding_Individuals_Geotagged_Messenger  
Input: Geotagged_Multimedia_Message (message1, message2)  
Output: Multimedia-enhanced Navigation  
1. while(message1, message2)  
2.   loc1 := extract_Location(message1)  
3.   loc2 := extract_Location(message2)  
4.   path[] := route(loc1, loc2)  
5.   add_Multimedia_Layer(path)  
6.   add_Traffic_Constraint(path)  
7. end while
```

Figure 4-20: Pseudocode of the algorithm for extracting location from geo-tagged multimedia messages and generating multimedia-enhanced route.

4.5 Conclusion

This chapter presents a *smart maps* framework that adds dynamic layers to traditional maps by handling social data streams, and by developing different algorithms for the efficient extraction, clustering, and mapping of live crowd-sourced events. This framework wraps incoming unstructured data streams into data packets, that is, a generic structured format of a potential event. These packets are then processed to extract EoIs based on different dynamic layers. This framework helps in enhancing existing spatial queries including city explorer and routing using extracted knowledge of the dynamic layers. This platform can be easily enriched with new data sources, such as online newspapers. The system can provide valuable knowledge to authorities, governments, market firms, POI owners, event organizers, and end-users in decision making, thus enhancing infrastructure and human life style. Our approach is scalable but not tested for the whole world.

Moreover, developing smart maps requires an efficient and scalable processing and the visualization of knowledge-based layers at multiple map scales, thus allowing a smooth and clutter-free browsing experience.

Chapter 5

Towards a Framework for Scalable Multi-Resolution Event Enriched Maps

5.1 Introduction

The advent of the twenty-first century has seen a tremendous revolution in Map design with the aim of disseminating knowledge layers derived from heterogeneous and diverse data sources. With the wide-spread access to tablets and smartphones, and the ease of availability of positioning technologies like GPS, the frequency of usage of maps, mainly for navigation, is unprecedented. Starting from paper maps thousands of years ago, maps have always been used to aid travelers navigating through their places of interest [110]. As information science emerged and the need for new ways to disseminate information grew, mapping began to include themes. Accordingly, maps went to a new era where people can browse various layers of the map, such as points of interest, roads and terrains. At that time, new layers were mainly seeded by central mapping agencies of the government, and map-making was often considered an activity that has national security implications. However, today's digital maps are often crowd-sourced, allow interactive route planning, and may contain live updates, such as traffic congestion state. Consequently, digital mapping applications are nowadays leveraging live data to improve navigation services, and to detect traffic congestion states.

Our vision for the next generation of maps is based on the observation that a lot of relevant spatio-temporal information is embedded in social media streams, governmental statistics and news. We believe that a major additional functionality that can be integrated into maps will be displaying *live and historical events*, extracted dynamically from user-generated content or crowd-sourced data. If intelligent mapping systems can discover relevant information from these unstructured data sources, the map browsing experience can be enriched significantly. For example, a spike in tweets talking about food at a particular location coupled with new Foursquare check-ins, can indicate the opening of a new restaurant (see Figure 5-1). In contrast

to traditional digital maps, such *smart mapping systems* can discover new content automatically by identifying new points of interests, events, or findings that were not specifically entered to the map.

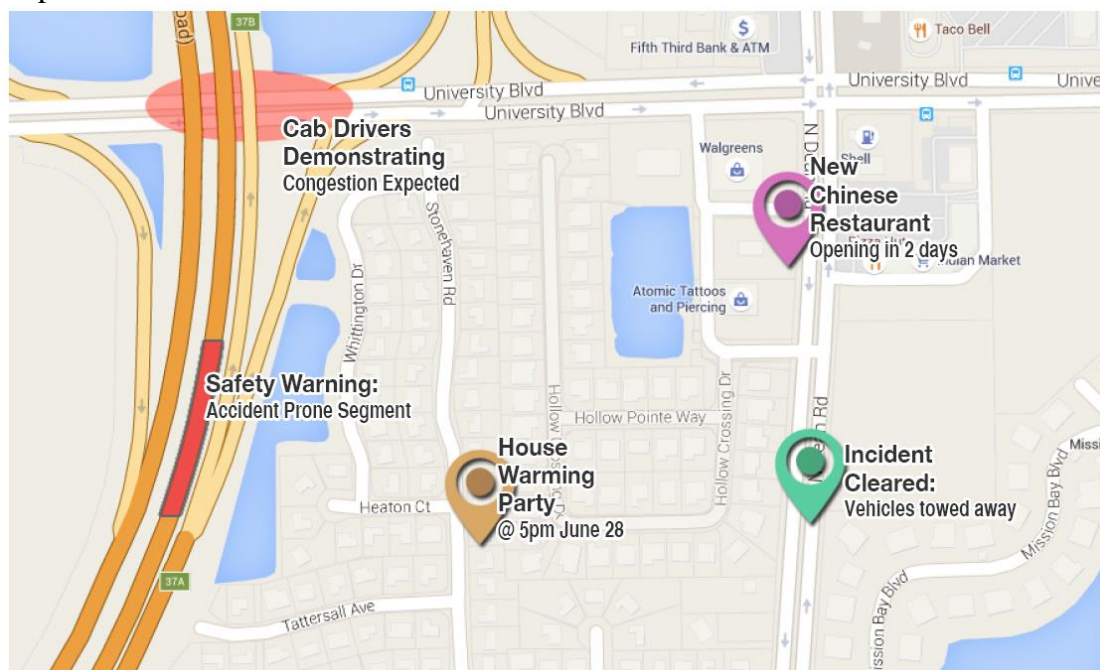


Figure 5-1: Conceptual illustration of Event-Enriched Maps: Findings automatically discovered from live streams, such as a restaurant opening, a neighborhood party, an accident prone road segment, warnings on demonstrations and emergency cases

The challenge in identifying new information to display on the map is multidimensional. First we need to infer map-worthy events and new places of interest from diverse live streams. This is a challenge in natural language understanding and context extraction. Secondly, to display such events on a map, their significance and spatio-temporal extent or scope must be established, so that as a user changes the zoom level, only events of appropriate scope are displayed. Recent state-of-the-art has proposed several methods to extract specific types of events from social media or online news [111], [28], [43], [32]. Yet, none of these are oriented towards discovery of the spatio-temporal scope of detected events on a multi-resolution map. This challenge is of critical importance because events occur naturally at different scales and with varying significance levels. In mapping applications, the moot questions are what to display, when, and on which level(s) of granularity. These are based on the spatio-temporal extents of corresponding items. For instance, when viewing an entire city, events with city-wide interest should be displayed. As the user zooms in, events of progressively narrower scope should be displayed. A soccer match can be of interest at the city scale, whereas a wedding may be of interest only at the neighborhood level. Hence, a framework to articulate the spatial and temporal scopes of events from live data streams is essential to develop a user friendly, multi-resolution map browsing experience.

To address these challenges, we present *Hadath*²⁴, a proof-of-concept implementation to demonstrate the feasibility and adaptability of building *Event-Enriched Maps* by displaying real-time events and unusual happenings by collecting and managing unstructured social streams. Hadath implements different techniques for the extraction, clustering, scope determination, and effective mapping of social data streams. Our approach aims at providing an efficient and scalable framework for the management of a large number of microblogs that are disseminated worldwide, by employing multidimensional in-memory indexing schemes, and a hierarchical clustering of candidate data points. Hadath digests incoming streams into a unique data packet format; and uses a specified string matching technique to detect and match candidate packets with our event classifier corpus in order to identify potential event classes and properties. Moreover, the system implements an unspecified topic detection method that extracts spatio-temporal peaks and unusual happenings based on the occurrence score and diffusion sensitivity.

Subsequently, local events extracted within limited spatial ranges—as calculated by the relevant spatial index—can then be aggregated with similar events in neighboring areas in a hierarchical manner. This will automatically allow the spatial and temporal scope of aggregated events to get updated. Clustering of events is dependent on not just spatial and temporal dimensions but also the cosine similarity between related packets. After determining the spatio-temporal scope for a given clustered event, we define the map level(s) of abstraction for which this event or sub-cluster of events can be displayed: this allows for an effective and smooth EoI visualization. This will also lend a new dimension to existing maps, a dimension which will necessarily be at multiple spatio-temporal resolutions.

The remainder of this chapter is as follows. Section 5.2 motivates the needs for event-enriched maps with several example scenarios and challenges. Section 5.3 introduces our system architecture, and presents the different components in details along with multiple developed algorithms for building multi-resolution event-enriched maps. Section 5.4, 5.5, and 6.6 discuss on data cleaner and wrapper, data manager, and event of interest detector; while Section 6.7 draws conclusions and discusses future work.

5.2 Motivation and Challenges

Traditional mapping systems do not allow real-time, on the move access to spatio-temporal information. A mapping system capable of defining its spatio-temporal scope with an appropriate significance level has the potential to become a city search engine. Such a search engine will allow quick identification of urban events at different scales, such as musical events, concerts, jobs, weather updates, traffic conditions, marches and protests, crimes, robberies, disaster events, or any other happenings.

Consider, for example, a city dweller interested in current ongoing activities in her neighborhood, such as outstanding offers and discounts, opera shows, football matches, or live

²⁴ Hadath is an Arabic word for an event or an unusual happening within a specified time and space.

events in the museum. At the city level, one might need to check hiring status for local recruiters as well as good and safe places to visit with reference to the latest statistics on accidents and incidents. For instance, a query like “*find a safe path to the nearest music concert with some ticket discounts*” can be of particular interest in this context. Current mapping systems are unable to answer this type of query.

Let us consider another scenario: two accidents happened in New York City. While the first was identified by a limited number of tweets from local authorities and other affected users in that particular locality, the second one witnessed tweets from not just authorities and affected users but also a sudden spike in tweets talking about that accident, culminating in the rapid spread of related tweets on such a large spatial extent that it covers all of the US. The viral dissemination of information in the second case may well indicate the involvement of a celebrity or a public figure as a victim of that accident. Hence, these two events should be visualized varyingly on two different map zoom levels given their varying significance.

Thus, the display of live events at differing levels of map abstractions presents a great opportunity as it is a challenge yet to be addressed.

Challenge 1: Dealing with Heterogeneous Data Sources

Information about potential events is scattered among the different data sources, such as Twitter, Instagram or Flickr. One key challenge is how to engineer a system to exploit disparate sources with diverse unstructured content, varying input rates and volumes. Extracting *live events* dynamically from a variety of data sources requires converting unstructured data streams into a common structured format in real-time. Hence, there is a need to have a module that can package unstructured content into a single data composite, which can be mined for inferring relevant event content by higher layers of the system. Adding a new data source to the system requires defining the different package items and its meta-data context, without affecting any other component of the system.

Challenge 2: Discovering Events of Interest

Detecting an event requires building a classification model that indicates whether a given packet represents a potential unusual happening, and if so, labeling that packet with the type or category as provided in the trained data sets. The input raw features that should be provided to the classification model are obtained from the training data on existing built corpora. Feature engineering includes n-gram, TF-IDF, or vector embedding in order to represent the textual data. Moreover, events can be discovered more efficiently on small-scale regions based on nearby locations and text similarity, but it becomes more challenging to efficiently merge similar events as we zoom out of the map. For instance, events of someone's birthday cannot be displayed at a national level, except if this person is a celebrity, and that particular happening had spread throughout the country.

Challenge 3: Understanding Spatio-Temporal Scope

It is necessary to extract not only the event information and location, but also its spatio-temporal extent. In mapping applications, the context of what to display is set by the spatial extents of the visualization. When viewing the entire city, events that have a city-level interest should be displayed. As the user zooms in, events of progressively narrower scope must be displayed. For example, a soccer match can be of interest for the city scale, whereas a wedding may be of interest only at the local scale. However, even in the case of a wedding, that event may need to be displayed on higher levels of abstraction if it involves lots of streaming input from other neighborhoods or cities, as may be the case, for example, of a celebrity wedding. A framework to extract spatial extent from live data streams is essential if a reasonable map browsing experience needs to be generated. Furthermore, the temporal scope needs also to be defined to clean old or unnecessary events after a certain time period.

Challenge 4: Efficiency and Scalability

Browsing *event-enriched maps* requires a smooth and fast panning and zooming capabilities. Events of different levels of abstraction are shown on the fly depending on the user navigation behavior. An important challenge here consists in managing input data streams as well as extracted events for efficient processing and retrieval. With the large volume of incoming streams, data indexing and the distributed processing of data represent an essential part of this system. Both real-time and historical data need to be managed and processed for extracting the different categories of events. Consequently, such a system should provide support for both main-memory and disk-resident indexes.

5.3 Event-Enriched Maps

This section introduces the key concepts and necessary definitions used throughout the chapter 5 and 6. Detecting social events of interest at multiple map scales can lay down the ground for building intelligent event-enriched maps. The different concepts developed in this paper are described as follows.

Definition 1 (Deep Maps): A Map that intelligently self-update themselves based on information extracted dynamically from heterogeneous data sources including social media streams, crowd-sourced data, sensors and online news sources where information can be any knowledge about Events of Interest (EoI) in real-time or based on statistical learning.

Definition 2 (Event of Interest): An Event of Interest (EoI) is an occurrence or happening at a certain place and within a specific time period, that holds several properties and a given level of importance. An EoI is represented as follows.

$$EoI = (DP, S, T, LS)$$

where ‘DP’ is the set of data packets that form the extracted event, ‘S’ is a spatial geometry that depicts the point or region where this event occurred, ‘T’ is the temporal extent for this event, and ‘LS’ is the level(s) of significance where this event should be displayed on map.

An Event of Interest can be any happening occurring within a spatio-temporal peak including concerts, sport events, competitions, accidents, birthdays, meetings, hiring’s, natural disasters, etc. In philosophy, “Jaegwon Kim”²⁵ theorized an event with an Object, property, and time. Taking this concept a shift forward, the spatial extent is associated to an event of interest, so that we can assess and visualize the significance of events on maps. The set of data packets ‘DP’ is represented as follows.

$$P_1, P_2, \dots, P_m$$

Where all $P_{1\dots m} \in P$ are related to a same real-world occurrence with similar properties and within a limited spatio-temporal range. The event spatial geometry S is determined by the centroid of all data packets locations that form the event

Definition 3 (Sliding Window): A sliding window is a window of ‘t’ hours, where ‘t’ refers to a specific period of data collection. For a window ‘t’, system will collect all data in one batch file ‘BF_{ts}’ of ‘t’ hours from a source ‘s’ and later that file is used to generate data packets.

Definition 4 (Data Packets): A data packet depicts a generic structured form of data by cleaning and filtering multiple types of unstructured data sources. Formally, a data packet is defined by DP (Header, Payload)

where a ‘Header’ is (time, geo(type, coordinates), city, country, eventProperties, potentialEvent, eventClass) and

‘Payload’ is (tags, id, followers, title, text, viewers, screenname, displayName, language, url)

Each data packet has a meta-data header that includes the source, location, time, event class, and other properties; and a payload mainly encapsulates the multimedia content, user details, along with the of predefined tags. The value of the ‘potentialEvent’ flag determines whether this event is a specified or unspecified event.

Definition 5 (Specified Events of Interest): A specified event of interest falls into a limited list of expected event categories (e.g., concert, football match, birthday, hiring, and storm); those events are repeatable and can match with existing corpora.

²⁵ Jaegwon Kim (1993) Supervenience and Mind, page 37, Cambridge University Press

Definition 6 (Unspecified Events of Interest): An unspecified event of interest depicts an unknown spatio-temporal peak, where the content does not match with existing corpora (e.g. authorities giving a new name to a hurricane such as Mathew, Agnes, Floyd).

It is necessary to extract not only the specified or unspecified event information, but also its spatio-temporal extent. In mapping applications, the context of what to display is set by the spatial extent of the visualized point or event of interest.

Definition 7 (Spatial Scope): A spatial scope of an event represents the geographical extent where this event has been disseminated. This helps in identifying the importance of an event at multiple map resolutions. As a result, when viewing the entire city, events that have a city-wide interest should be displayed. As the user zooms in, events of progressively narrower scope must be highlighted.

Definition 8 (Temporal Scope): A temporal scope of an event determines the period of time this event remains alive. The temporal scope allows to track the temporal evolution of an event depending on users' interaction, and to clean old or unnecessary events when their time period is drained.

Three main parameters are considered when determining the temporal scope: a) 'birth time' that indicates the existence of a new event in our system whenever we calculate the first cluster of data packets related to that event; b) 'time of occurrence' that marks the actual happening time of the event (e.g., next Monday); and c) 'time to live' (TTL) to depict the survival time of an event based on users' interaction.

Definition 9 (Event Level of Significance): The level of significance LS for an event is a mapping between the spatial scope and the map zoom levels in order to provide a unique and dynamic map browsing experience at different abstraction levels.

Given the above concepts and definitions, we formulate our problem as follows. For a given batch file BF_{ts} with a sliding window of 't' hours from a data source 's', our aim is to generate a list of indexed data packets 'DP' of potential specified and unspecified events. Extracted events at the local scale, that are related to same real-world occurrences should then be clustered together based on spatio-temporal and text similarity parameters, among other factors. Several EoI layers should be computed for event with similar significance levels (e.g., district, county, city or country levels). An $EoI_{Layer_{LS}}$ at a given significance layer 'LS' is represented by $(EoI_1, EoI_2, \dots, EoI_k)$, should be displayed on map with unique panning and zooming capabilities.

5.4 System Design

In Hadath system, the context of what to display is set by the spatial extent of the detected events. When viewing the entire city, events that have a global interest should be displayed. As the user zooms in, events of progressively narrower scope must be displayed. For example, a

soccer match can be of interest for the global scale, whereas a wedding may be of interest only at the district scale. However, even in the case of a wedding, the same event may need to be displayed on higher levels of abstraction if it involves lots of streaming input from other neighborhoods or cities, as may be the case of a celebrity wedding. In addition, unlike existing systems where knowledge is extracted based on users' requests, the key principle behind *Event-Enriched Maps* is to extract knowledge on the fly by digesting social data streams and to infer its spatial scope, whether it covers a neighborhood, town, county, state, national or international level. This section presents *Hadath*: a novel map-based platform that collects social data streams from multiple sources, processes data to find events of interest (EoI) and visualizes detected events in correspondence to the scale of the view. Figure 5-2 illustrates an overview of our *Hadath* architecture with the salient components, which are highlighted as follows.

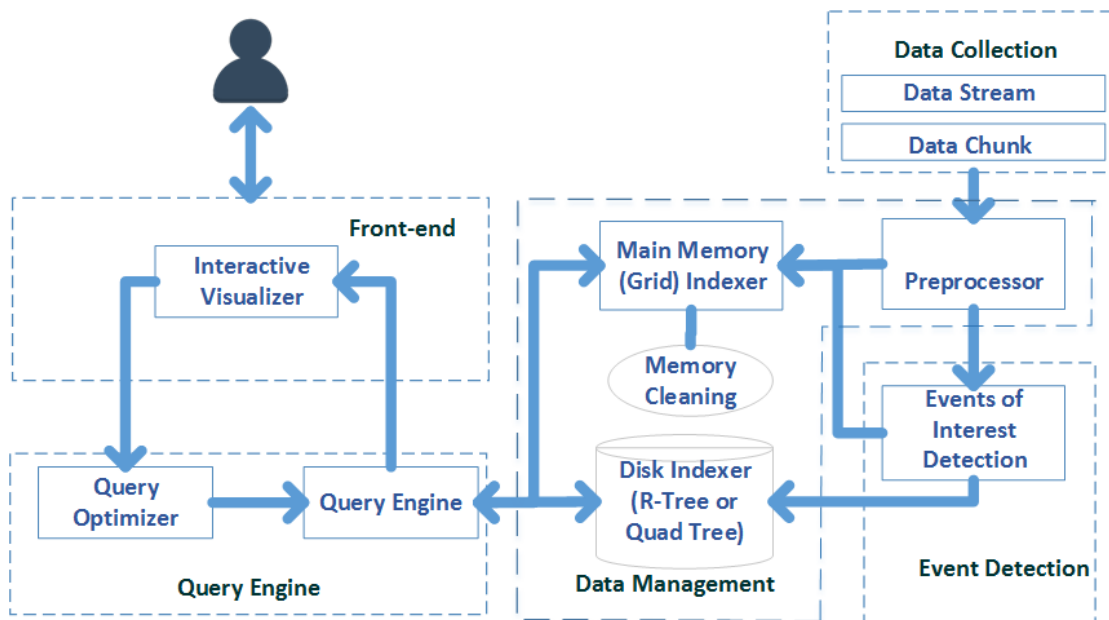


Figure 5-2: Hadath System Architecture

- *Data collection* module involves gathering data from multiple sources with different unstructured forms. This includes digesting data streams (e.g., Twitter, Instagram, Yelp) and data chunks (e.g., open government, news, historical tweets). Digesting data streams is performed by running crawlers that collect bulks of streams based on windows of a specified temporal extent.
- *Data wrapper* manages packaging input data streams into a generic structured form. The major task for the data wrapper is to digest data from multiple sources and bundle it into a unique packet for with a meta-data header and a payload. The data wrapper also assigns a confidence level and classifies data streams based on their potential to reflect some event category.

-
- *Data manager* stores data by implementing a three level temporal and spatial pyramid indexing scheme to allow efficient and scalable access to raw data streams. This also includes a hierarchical clustering and indexing of EOIs in order to detect the spatial scope and the level of detail of a given event.
 - *Events of Interest (EoI)* detection module classifies and extracts events within different categories, such as, social events (e.g., concert, match, graduation party), road accidents, incidents, accident prone areas, non-safe area, and other breaking news. The event detection module is not a part of the query engine, so instead of querying spatio-temporal events with an on-demand basis, *Hadath* works continuously to process data from incoming streams in order to extract and update EOIs.
 - *Query Engine* that has two components; a) query optimizer and b) query processor. The query optimizer creates best query plan based on map zoom level, spatial and temporal characteristics. The query processor executes the query plan in order to retrieve EOIs efficiently. *Hadath's* query engine supports efficient retrieval of raw streams and of pre-processed events based on the main querying attributes that are, the spatial, temporal, and map level of detail.
 - *Visualizer* provides a new dimension to existing maps by illustrating extracted knowledge from live streams in the form of live events with different spatial scopes and at different levels of abstraction. This allows us to show live events in correspondence to the map level of detail (LOD), that is, when viewing at a city scale, events of higher significance are displayed; whereas, when zooming in to a given neighborhood, events of a more local interest are highlighted. The final output creates a unique and dynamic map browsing experience.

5.5 Data Cleaner and Wrapper

The *data cleaner and wrapper* provides an efficient and generic mechanism with the aim of allowing new data sources (e.g., Flickr) to be easily plugged, by supporting new crawlers at the data collection level without affecting the other processing components. Figure 5-3 shows a conceptual example of data packets generated from multitude of data. Major tasks for the data cleaner and wrapper are: 1) to clean irrelevant fields and digest incoming streams into a unique data packet format. Like the analogy of TCP protocol, each data packet has a meta-data header, containing source, location, time, type, and a payload, containing the actual contents including user profile details. This allows our system to digest different types of data input, and to generate structured data from unstructured streams; 2) to use specified string matching technique that detect and match candidate packets with our event classifier corpus in order to identify potential event classes and properties. This approach help us to extract relevant packets related to known event classes including social, disaster, religious, weather, job, traffic, sports, political/government and musical events ; and 3) to apply unspecified topic detection method that extract spatio-temporal peaks and unusual happenings based on the top frequent words. This approach helps us to detect unknown events that are not a part of corpus but their value

are more than threshold at given time. To add new source in our *Hadath*, we just need to add small piece of code without impacting other components of the system. The header and payload are populated in different ways for different data sources. For example, 'event properties' may be populated by the parsing hashtags, noun, verb in case of Twitter, and hashtags in case of a Flickr. Hence, a different data filter is written for each new source which is added to the system. This allows our system to digest different types of data input, and to generate structured data from unstructured streams. These packets are then processed to extract Events of Interest, and can be dealt with in a consistent manner by the higher layers of the system.

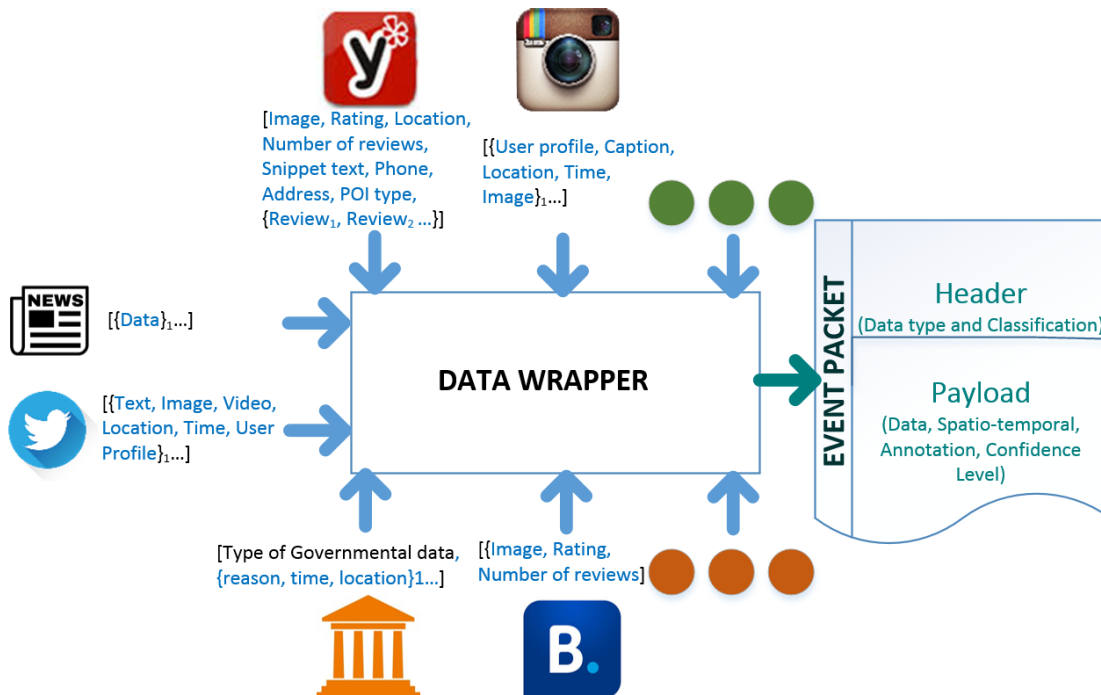


Figure 5-3: Wrapping multiple sources of data into a generic composite event packet

During data integration and preprocessing, the quality of data packets is also checked based on the following parameters:

- *Exactness of data*: data from multiple sources comes with different levels of correctness. Data from authority has a higher priority and trust level as compared to other sources. Incorrect data can be hazardous, so we need to check the exactness of data in order to make sure that no streams are generated by software bots, for instance.

Algorithm 5-1: Data Wrapper and Cleaner

Data T Window-based of geo-tagged tweets

Result DP (Header, Payload): List of data packets

Header: (time, geo(type, coordinates), city, country, eventProperties, potentialEvent, eventClass)

Payload: (tags, id, followers, title, text, viewers, screenname, displayName, language, url)

begin

```
TweetsTokenInfo[(token[ ], type[ ], confidenceScore[ ]) tInfo; //tInfo' is an instance
of class TweetsTokenInfo
```

```
EventField (properties, class, potentialFlag) eFields; //eFields' is an instance of class
EventField
```

```
tInfo[ ] = tokenizationPOS_NLP ( $T$ );
```

```
words[ ] = topFrequentWord (tInfo, frequencyThreshold);
```

foreach tweet $t \in T$ **do**

```
  eFields = matchWithExistingCorpus (tInfo[t]); //check for specified events
```

```
  if eFields.class is null and t.text.contains (words) then //check for unspecified events
```

```
    eFields.class = unspecified;
```

```
    eFields.potentialFlag = true;
```

```
  if eFields.class is null then //if tweet 't' is neither specified nor unspecified
```

```
    discard(t);
```

```
  else // Generating packets for both specified and unspecified events
```

```
    t = preprocessing (t);
```

```
    //preprocessing perform cleaning and structuring of data
```

```
    spatioTempoInfo = extractSpatioTemporalInfo(t);
```

```
    header = createHeader (spatioTempoInfo, eFields);
```

```
    payload = createPayload (t);
```

```
     $DP.add$  (createPacket (header, payload);
```

```
  end
```

```
end
```

```
return  $DP$ ;
```

end

- *Duplication Removal*: Many chunks of data can be retrieved several times, as in the case of a weather streaming source, that continuously publish new streams on the current weather status for a given city. So, we need to remove duplication in case of multiple streams related to the same event and from the same source.
- *Data dependency*: Some information provided by one source may be incomplete and requires more data to extract relevant knowledge [21]. For example: from tweets, we get information about an accident but after analyzing data from physical sensors, we can get details of the severity and impact of that accident.

The algorithm for determining the event class (specified or unspecified events), and event properties, and for converting unstructured streams into a generic composite event packet is illustrated in Algorithm 5-1. Every window-based bulk of geo-tagged tweet T is processed in order to generate unified structured list of packets with metadata (i.e., Header) and (i.e., Payload) DP (Header, Payload). The ‘Header’ contains the time, event class, event properties, potential event flag, city, country, geo-tagged type, and location; whereas the ‘Payload’ contains the list of tags, title, text, number of viewers/followers, screen name of the user, display name of the user, the language and url. Each T has to go through the following steps:

- $tInfo[]$ is an instance of class `TweetstokenInfo` that contains (`token[]`; `type[]`; `confidenceScore[]`).
- $eFields$ is an instance of class `EventField` that contains (`properties`; `class`; `potential flag`).
- `tokenizationPOS_NLP` (T) takes a window-based bulk of geo-tagged tweet T as input, and applies preprocessing techniques including part of speech and tokenization to return a list of $tInfo$ where $tInfo[i]$ contains information about ‘ i^{th} ’ tweet. In tweets, user have limited character so they prefer to use shortform or out-of-vocabulary word. Table 5-1 shows the result for a given tweet “Here with my mate Ben for the Broods concert @2degrees (@ The Opera House in Wellington Central, Wellington) <https://t.co/Sk0P1BthD>”; where the first column denotes the token, the middle column shows the type, ‘R’ for adverb, ‘P’ for pre or post position or subordinating conjunction, ‘D’ for determiner, ‘N’ for Noun, ‘^’ for proper noun, ‘@’ to mention other users, ‘,’ for punctuation respectively and last column shows the confidence score. For example, assume the above i^{th} tweet and $tInfo[i]$ represents the tweet tokenization. The first word in the tweet is ‘Here’ stored in the token $tInfo[i].token[0]$, and ‘R’ is its type $tInfo[i].type[0]$ (e.g., adverb), and ‘0.9737’ is its confidence score $tInfo[i].confidenceScore[0]$.

Token	Type	Confidence Score
Here	R	0.9737
with	P	0.9995
my	D	0.9990
mate	N	0.9982
Ben	^	0.9973
for	P	0.9996
the	D	0.9965
Broods	N	0.4730
concert	N	0.9966
@2degrees	@	0.9962
(@	P	0.8991
The	D	0.9795
Opera	^	0.9267
House	N	0.5923
in	P	0.9983
Wellington	^	0.9977
Central	N	0.5726
,	,	0.9967
Wellington	^	0.9924
)	,	0.9634
https://t.co/Sk0P1BthDU	U	0.9966

Table 5-1: NLP for a given text with type and confidence score

- The `'topFrequentWords'` function takes `tInfo` (i.e., tweet data after tokenization) and `frequencyThreshold` as input to find top the frequent words for a given window-based bulk tweets. We are using processed data `tInfo` in the function instead of bulk of geo-tagged tweets `T`, to avoid irrelevant tokens including punctuation, URL, pre or post positions, and by taking into account the relevant types with high value of confidence score.
- The for loop encapsulates the process of structuring streams into unified packets along with a meta-data content as follows.
- `'matchWithExistingCorpus'` function processes a tweet `'t'` with an existing event corpus. It will match extracted tokens to identify a potential event class, if any. It will return an `EventField` instance that contains an event class, properties and a potential flag (see Figure 5-4).

```

{"header":{"time":"Fri Jan 01 00:39:11 +0000
2016","geo":{"type":"point","coordinates":[34.84863834,-95.54509333]}, "source":
"twitter", "eventType":["Party"], "type":"text","potentialEvent":true",
"eventname":"Social events/party","country":"United States","city":"Oklahoma"},
"payload":{"tags":[],"id":"2605873770","followers":"581","title":"","text":"Party @
Stacey's 😊", "viewers":"0", "screenName":"TheLedgenBeare", "language":"en",
"displayName":"Ledgen", "url":null}}

```

```

{"header":{"time":"Fri Jan 01 00:29:01 +0000 2016","geo":{"type":"point",
"coordinates":[36.644065, -93.213989]}, "source": "flickr", "eventType":["newyears",
"fireworks"], "type":"text", "potentialEvent":true", "eventname":"Social
events/NewYear", "country":"","city":""},
"payload":{"tags":["winter", "red", "holiday", "night", "fireworks", "outdoor", "january",
"clear", "missouri", "newyears", "nightsky", "nightscape", "tradition", "branson",
"firecrackers", "clearsky", "bransonmissouri", "clearnight", "mobilephotography",
"bransonlanding"], "id":"24017620411", "followers":"","title":" New Year's fireworks
", "text":" Branson, Missouri 12:00 am, January 1, 2016.", "viewers":"133",
"screenName":"","language":"","displayName":" yetanotherstephanie ", "url":
"https://farm2.staticflickr.com/1493/24017620411_31fe81c0b6.jpg"}}

```

Figure 5-4: Sample Data Packet from Twitter and Flickr Streams

- If an event class is null (eFields.class) and t.text.contains (words), which means it is not in the corpus of known events, we will check for a potential unusual happening (i.e., unspecified events) by looking into frequent words.
- If an event class is still null then this means it is neither specified nor unspecified event. Packets that show no relevancy with respect to the above steps are discarded at this phase.
- Candidate packets are then processed to extract spatio-temporal information and to create corresponding Header and Payload.
- Finally, the algorithm returns a list of composite candidate packets for a given window-based bulk of geo-tagged tweet T.

5.6 Data Manager

The data manager implements an in-memory spatial indexing scheme to allow an efficient and scalable access to data packets. The spatial index is a multi-resolution data structure (similar to a partial quad tree originally introduced in [105]). Leaves in this data structure correspond to cells that represent the minimum bounding rectangles comprising data packets. Figure 5-5

displays a snapshot of indexed data packets at a fine level of the hierarchical tree, and with a single day specified as a time threshold. Cells are colored lighter to darker based on data packet counts; darker-coloured cells are further expanded at deeper levels in the tree as compared to lighter-coloured cells. *Hadath* employs a big data mechanism that continuously process data packets within the different cells on several execution nodes. The manager also indexes detected EoIs in order to fetch them efficiently based on the map zoom level and scope. Using this multi-resolution indexing scheme, hierarchical clustering of events can be applied for efficient determination of their content and spatial scope. For temporal aspects and cleaning of EoIs, *Hadath* incorporates three parameters: a) `birth time' that indicates the existence of a new event in the system whenever the first cluster of data packets related to that event is computed; b) `time of occurrence' that marks the actual happening time of the event (e.g., next Monday); and c) `time to live' (TTL) is the survival time of an event within the system. Whenever we receive new data packets related to an existing event, we increase its TTL by T number of hours. Moreover, processed data packets are moved to disks based on temporal and memory thresholds. The main task of the disk indexer is to index outdated data packets and events on disk using an R*-tree spatial index to allow efficient retrieval for historical queries.

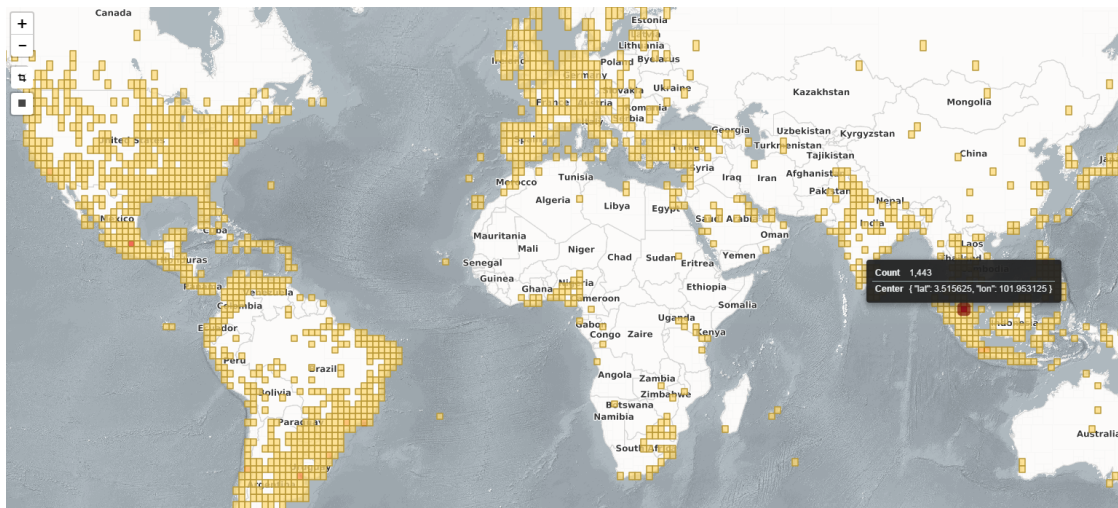


Figure 5-5: A snapshot of indexed data packets at a fine level of the hierarchical tree

Main Memory Indexer

The main memory spatial index is a multi-resolution data structure that stores packets in a hierarchical scheme based on stream density. Other indexes are implemented on the basis of temporal and keywords attributes. For the spatial index, we used a multi-level pyramid structure with a geohashing technique that divides the geographic space into buckets of a grid shape. Geohash converts two-dimensional spatial queries into one-dimensional string matching. With this advantage, Geohash can execute queries faster, with $O(1)$ time complexity. The keyword search is managed by an inverted index that maps a page-centric data structure (page->words) to a keyword-centric data structure (word->pages). Instead of searching text, it searches for the word index first and then find the document (e.g., tweet) related to that word. For temporal indexes, we divided each spatial segment and keyword segment in 'T' hours where 'T' hours can be configurable based on the size of main memory.

Following are the benefits of indexing techniques with respect to our system, 1) It helps to detect local event detection i.e. within specific cell based on different precision; 2) It helps to detect spatial scope efficiently; 3) It index the data in $O(1)$; 4) When memory reaches to its threshold, we can easily flush the previous 'T' hours to disk.

Disk

To get better understanding about the place, users' may also be interested in old EoIs apart from current or upcoming EoIs such as if a user wants to book an hotel in new city then she can browse nearby area and see what happened around in the past. To support EoIs for long periods, our system store data from main-memory to disk based on main memory threshold or based on temporal threshold. However, the disk index is a bit different from the main memory with respect to temporal parameters. Figure 5-6 shows an overview of the hierarchical disk index implementation in Hadath. To access EoIs efficiently, we distribute temporal data packets to monthly and daily bases. The monthly distribution stores month with year whereas daily distribution inside monthly distribution stores actual EoIs of particular day in a same pyramid manner explained in main memory section. For example, if a user requests data of June 2017, then the query processor needs to access 30 indexes inside the June month to fetch EoIs and keyword indexes.

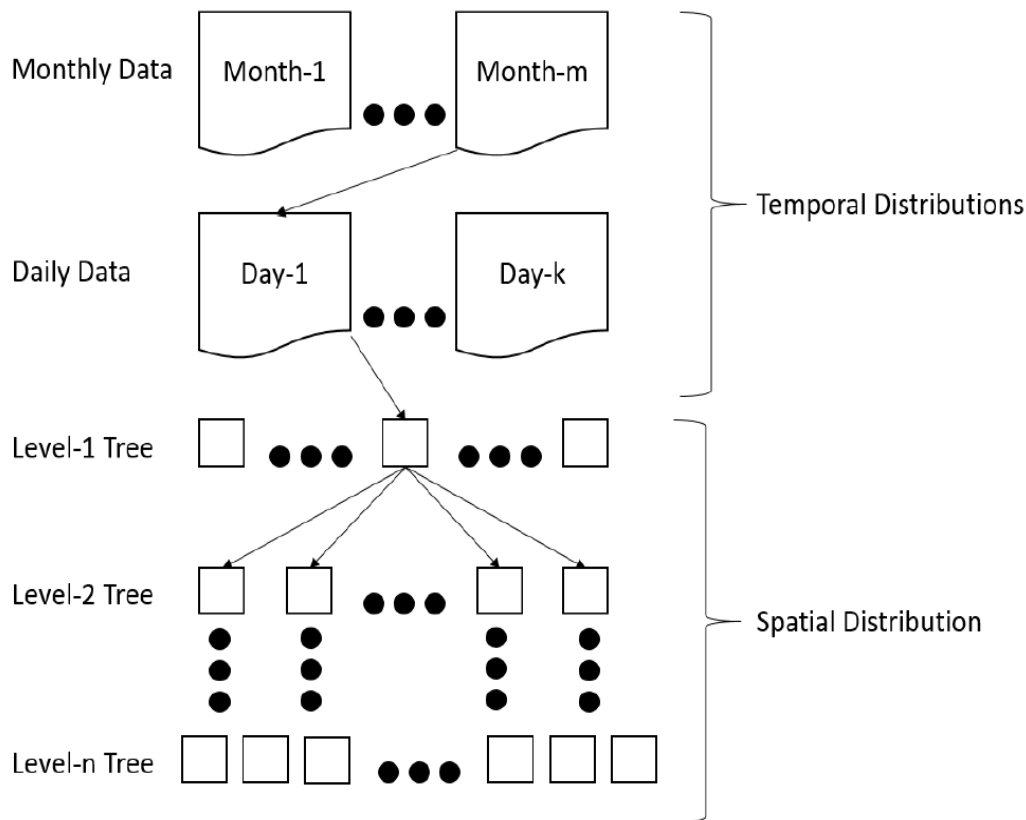


Figure 5-6: Multi-level spatio-temporal index

Memory Cleaning

The main task of memory cleaning manager is to clean expired data packets and EoIs from memory. An expired data packet is determined on the basis of temporal constraints at the TTL of the event. We have used periodic approach to clean EoIs periodically in conjunction with the piggybacking approach over the querying process. Whenever an EoI happens, we can check all EoIs for that particular type and update it.

5.7 Event of Interest Detection

After cleaning, wrapping and indexing of data packets, the event of interest detection module starts from the base level within the multi-resolution data structure to detect events at a local spatio-temporal scope. The base level contains cells of a fine resolution that are considered as leaves within the hierarchical pyramid data structure. Within each leaf cell, Hadath adopts the graph analogy where data packets are considered as nodes and the value of the 'text similarity (TF-IDF)' between data packets is computed as the weight of the bidirectional edges. Data packets with a high text similarity value within each cell are clustered together using the graph-specific Louvain clustering algorithm. The Louvain algorithm [112] is suitable in our approach as, unlike most of the other clustering methods, it does not require a prior knowledge of the minimum number of clusters. For unspecified events that are not matching our training corpus, this module detects frequent tags and keywords, in order to identify spatio-temporal peaks.

As illustrated in Algorithm 5-2, every leaf cell of the hierarchical Spatial Tree \mathcal{ST} has to go through the following actions in order to detect local events. Packets are received as a window-based bulk of data DP , and are indexed depending on the spatio-temporal dimensions. Packets within the leaf cells are then clustered based on their contextual similarities. Details of the algorithm are explained as follows:

- `indexNewPackets` function index new data packets DP in existing indexed hierarchical spatial tree \mathcal{ST} and return list of updated leaf cells in C_{leaf} (Line 1).
- for each updated leaf cell $c_i \in C_{leaf}$ do the following actions (Line 2).
- `retrieveExistingClusters` function to extract the list of existing event clusters in $OC \in c_i$ from \mathcal{CL}_{c_i} . Existing clusters are required to merge new packets with them, and to update their time to live 'TTL' (Line 3).
- `retrieveExistingNonClusteredPackets` function return list of existing data packets in $OP \in c_i$ that are not part of any cluster from spatial tree \mathcal{ST} . Old non-clustered packets are retrieved to check whether they can create new clusters with newly coming packets (Line 4).

Algorithm 5-2: Local EoI detection by clustering of event packets within leaf cells**Data** DP : Window-based bulk of data packets ST : Indexed existing hierarchical spatial tree \mathcal{CL}_{leaf} : Existing clusters at the leaf levels**Result** \mathcal{CL}_{leaf} : Updated clusters at the leaf levels**begin**

```

 $C_{leaf}[] = \text{indexNewPackets}(DP, ST); // \text{list of updated leaf cells after indexing new}$ 
Packets  $DP$ .

```

foreach cell $c_i \in C_{leaf}$ **do**

```

   $OC = \text{retrieveExistingClusters}(\mathcal{CL}_{leaf}, c_i);$ 

```

```

   $OP = \text{retrieveExistingNonClusteredPackets}(ST, c_i);$ 

```

```

   $NP = \text{retrieveNewIndexPackets}(ST, c_i);$ 

```

```

  initialize(D);

```

```

   $D = \text{computeCosineSimilarity}(OC, OP, NP);$ 

```

foreach nonClusterNewPackets $p \in NP$ **do**

```

  if ( $Cl_k = \text{checkSimilarityWithClusters}(OC, p, D)$  is not null) then

```

```

    addPacketsInExistingClusters( $Cl_k, p$ );

```

```

    updateEventTTL( $Cl_k, \text{timeThreshold}$ );

```

```

  else if ( $CP = \text{checkSimilarityWithPackets}(OP, p, D)$  is not null) then

```

```

     $Cl_k = \text{createCluster}(CP);$ 

```

```

     $\mathcal{CL}_{leaf}.add(Cl_k);$ 

```

```

    markPackets( $Cl_k, CP$ );

```

```

  end

```

end

```

  cleanOldClusters( $OP, OC$ ); // cleaning old packets and clusters based

```

```

  // on temporal threshold 'T' for packets 'P' and 'TTL' for clusters'

```

end**end**

-
- `retriveNewIndexPackets` function returns the list of existing data packets $\in NP \in C_i$ that are not part of any cluster from the hierarchical tree \mathcal{ST} . We need new packets $\in NP$ to check whether they are eligible to form new clusters or to be merged with old packets or clusters (Line 5).
 - `computeCosineSimilarity` function calculates cosine similarity between OC, OP, NP and stores the result in D. $D[][]$ is a matrix that shows the cosine similarity D_{ij} where each D_i and D_j can be in NP, OP and OC (new packets, old packets and old clusters) (Lines 6,7).
 - for each new packet $p \in NP$ that is not a part of cluster do the following actions (Line 8):
 - `checkSimilarityWithClusters` function checks clustering eligibility between old clusters OC and the new packet p. If p matches with any existing cluster, the function will return the cluster id Cl_k . If Cl_k is not null (i.e., cluster already exists), then add p to the cluster Cl_k and update TTL for Cl_k by incrementing by 'timeThreshold'
 - If no cluster matches with p, then the `checkSimilarityWithPackets` function check clustering eligibility of p with old packets OP. If p matches the criteria and threshold for clustering with old packets, the function will return the list of packets forming the new cluster in CP. Packets in CP will be marked as part of the cluster Cl_k (Lines 12-15).
 - `cleanOldClusters` function cleans old packet based on temporal threshold 'T' as well as existing clusters if their TTL are elapsed (Line 18). Other than this approach, we check periodically as well for cleaning of old data packets and expired clusters.

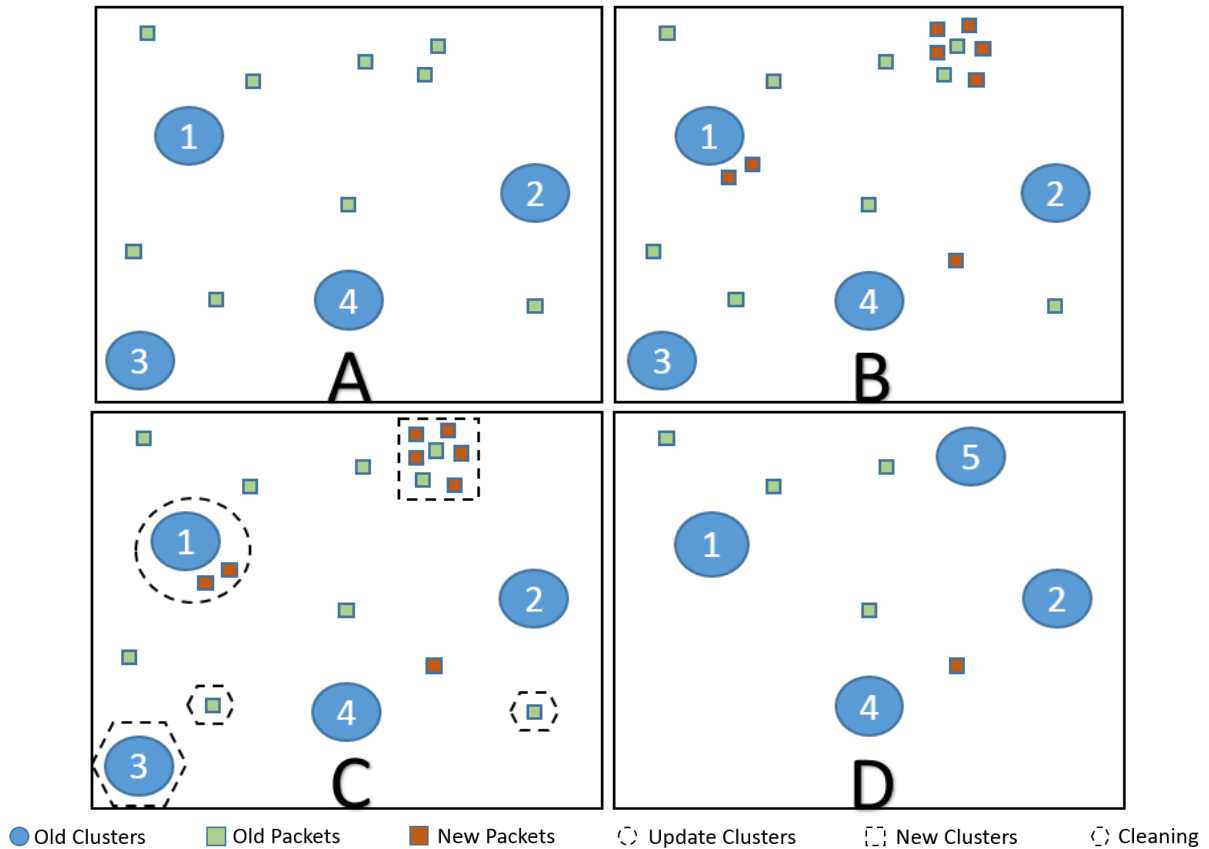


Figure 5-7: Local EOI Detection within the leaf cells A) List of existing clusters and packets, B) shows arrival of new indexed data packets, C) Merging new packets in existing cluster, forming new cluster, and cleaning old packets and clusters based on threshold value, D) resultant leaf cells after all operation mentioned in Algorithm 2

Every window-based bulk of data packets is indexed in the existing hierarchical spatial tree and with references in the temporal index. To create new event clusters within the leaf cells, an algorithm is developed so that similar packets within the local spatio-temporal scale are grouped together, or can be grouped within existing event clusters in the corresponding cell. Figure 5-7 shows stages of an event of interest detector in the following four different stages. A) Shows leaf level cluster with existing clusters (blue circle) and non-clustered data packets (light green square) before the arrival of the newly indexed data packets. B) Shows new indexed data packets (red square) in the same leaf level. Overall, it shows a list of Old clusters, old packets and New Packets; C) New clusters are formed and old ones are updated with new packets. Clusters and packets are considered as nodes in the cell, and distances between them are computed based on the cosine similarity value. As a result, some packets are merged with existing clusters (dotted circle), and the combination of some old and new packets forms a new cluster (dotted square). Also, cleaning of old packets and clusters is performed at this level, based on their temporal threshold and TTL parameter (dotted hexagon). D) shows the final

output of local event detection in leaf cells after creating new clusters, updating old clusters and cleaning of old packets and clusters.

5.8 Spatial Scope Finder

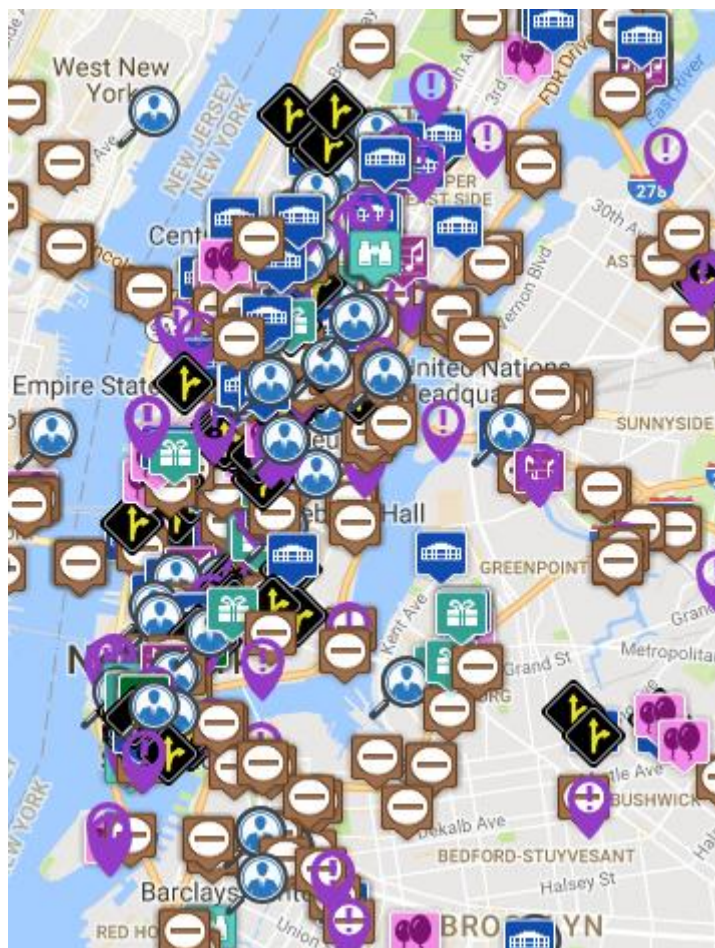


Figure 5-8: Visualization of EoI without scope

As events can be discovered more efficiently on small-scale regions (starting from leaf nodes), a bottom-up approach for clustering close-by and similar events is developed, so that redundant events on different spatial resolutions can be aggregated, and their spatial scope can be upgraded. Visualization of EoIs with the same spatial resolution on maps does not make sense, since these events have different significance from spatio-temporal perspectives and its difficult to browse such maps. Figure 5-8 has varieties of events which can be very useful for city explorer or any decision maker but due to clustered and overlap events its difficult for someone to browse such type of maps without any spatial scope of events. Such maps with lots of potential are still useless for end-users to browse or understand due to its complexity without any spatial scope. For instance, events of someone's birthday cannot be displayed at a national level, except is this person is a celebrity, and that happening had spread throughout the country.

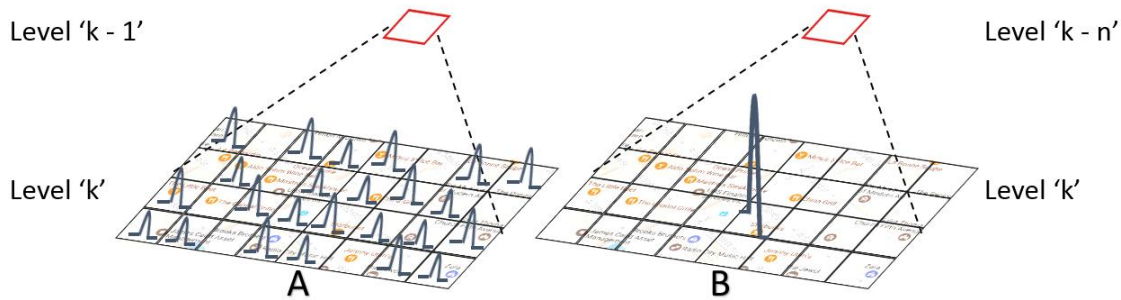


Figure 5-9: Spatial Scope. A) Horizontal and B) Vertical

Our multi-resolution clustering technique for event aggregation works as follows. Starting from events at neighbourhood/district level (i.e., which corresponds with leaf cells in our tree), the system clusters identical events at higher levels of abstraction, and incrementally increases their spatial scope. Local clusters are first compared with their siblings in the hierarchical tree to detect horizontal spatial scope, with the aim of aggregating and updating the scope of similar events such as election at city level where users from all over the city are talking about that event. Merging clusters ($Cl_1; Cl_2$) from different cells ($C_n; C_n$) at a depth level n in the tree, will result in upgrading their horizontal spatial scope from zoom level k (e.g., corresponds to district level on map) to zoom level $k - 1$ (e.g., corresponds to city level) as shown in Figure 5-9 A. Secondly, if the event is not merge with their siblings then we check the importance of event to decide vertical spatial scope such as Opera/concert ongoing live events where attendees of the event, tweets more as compared to near by siblings. In this case, based on importance of event we decide the vertical spatial scope from zoom level k (e.g., corresponds to neighbourhood level on map) to zoom level $k - n$ (e.g., corresponds to city level) where value of n decided based on total number of tweets related to event and number of unique users as shown in Figure 5-9 B. We took number of unique users in calculating 'n' to identify the tweets from bots or same user. For example: user ID 1484740038 (Pen-Y-Renglyn) uploading tweets about weather in every 2 minutes.

An event cluster Cl_i is represented as follows:

```
( id; ptGeom; cellKey; childCellKeys; totalNonEmptyCells;
  eventClass; eventProperties; packetIDs; topContents; imageURLs;
  scopeUpdated; uniqueUserIDs, zoomLevelStart; zoomLevelEnd )
```

where 'id' is cluster identifier, 'pointGeom' is the centroid point location, 'cellKey' is the identifier of the cell, 'childCellKeys' is the list of child cell key of merged clusters, 'totalNonEmptyCells' is the number of non-empty cells that has one or more than one data packets, 'eventClass' and 'eventProperties' depict the event class(es) and a list of top frequent meaningful words within the cluster, 'packetIDs' is the list of data packets identifiers forming that cluster, 'imageURLs' is the list top selected image URLs, 'topContents' is the list top selected tweets text, 'scopeUpdated' is the flag to keep track that cluster spatial scope is processed or not, 'uniqueUserIDs' is the list of unique users and 'zoomLevelStart, zoomLevelEnd' correspond to the multiple resolutions where this event is available to be displayed on map. The details behind using the above cluster properties are highlighted in

Algorithm 5-3. This algorithm runs multiple times for each level X of the hierarchical tree \mathcal{ST} , starting from $X = \text{Leaf level of } \mathcal{ST}$ up until reaching the root.

Algorithm 5-3: Efficient hierarchical clustering to determine spatial scope**Data:** \mathcal{ST} : Hierarchical spatial tree \mathcal{C}_{X-1} : List of cells at level 'X-1' in the tree \mathcal{CL}_X : Generated clusters in the level X cells**Result:** \mathcal{ACL}_{X-1} : Aggregated Clusters with updated spatial scope at level 'X-1'**begin****foreach** cell $c \in \mathcal{C}_{X-1}$ **do****if** $c.hasChild()$ is not null **then** // i.e. subtree is not a leaf nodecluster[] = $getAllChildClusters(t, \mathcal{T})$; // retrieve all clusters from the children of c **foreach** cell $cl_i \in$ cluster[] **do****if** $cluster[i].scopeUpdated()$ is false) **then**aggCl[] = $createAggregatedClusters(cl_i)$ **foreach** cell $cl_j \in$ cluster[] **do****if** $cl_i.cellId \neq cl_j.cellKey$ and $(matchSimilarity(cl_i, cl_j) \geq \mathcal{S})$ **then**aggCl.add(cl_j);**end****if** $aggCl.size()$ is equal to 1 **then** //no matching clusters with cl_i vs = $check_CalculateVerticalSignificance(cl_i)$;//check and if required calculate vertical significance value of cl_i **if** $vs \geq \mathcal{T}$ **then**acl.updateVerticalScope (cl_i); //update start and end zoom levels of cl_i **else if** $aggCl.size() \geq \Gamma$ **then**

acl.setLocation();

acl.updateTopContent();

// update start and end zoom levels for acl

acl.createClusters($aggCl, c$);createClusters($aggCl, c$); $\mathcal{ACL}.add(acl)$;setUpdatedScope ($cl_i, true$);**end****end****end**

At the beginning of this algorithm, we assume that all clusters at the leaf cells have been detected and indexed in corresponding cells. Moreover, this algorithm is repeated from level $X-1$ until reaching the root level, so that each loop will consist in re-assessing the set clusters in a given subtree and if any merging is possible, new clusters at higher levels of the tree are created with an upgraded spatial scope. As illustrated in Algorithm 5-3, every non-leaf cell $c \in \mathcal{C}_{X-1}$ at level $X-1$ of the hierarchical Spatial Tree \mathcal{ST} has to go through the following actions (Line 2):

- If c has child cells, the algorithm looks into all child cells of c and retrieves all child clusters into `clusters[]` using `getAllChildClustersFromChildCells(c)` method (Lines 3-4, see Figure 5-10).
- For each cluster $cl_i \in \text{clusters}[]$, we check whether cl_i scope has been previously updated (Lines 6). If cl_i scope is not set, then we define a new list of aggregated clusters `aggCl[]` that may encapsulate multiple clusters starting from cl_i (Line 7).
- In the second loop, we look into other existing clusters $cl_j \in \text{clusters}[]$, and verify their matching similarity criteria as described in Algorithm 5-2 (Lines 7-10). θ is a specified threshold to determine the matching eligibility between clusters of sibling cells. Matching clusters are added to `aggCl[]` so that they be merged at a higher level of the tree.
- If the `aggCl[]` list size is equal to one, this means that no other cluster from sibling cells can be merged with cl_i . Therefore, we do not need to check for the horizontal spatial scope, as sibling cells have not leveraged that particular event. However in this case, the algorithm checks for a potential vertical significance of cl_i . It calculates the number of packets and the number of unique users that formed the cl_i cluster. If the ratio of the number of users to the number of packets is more than the vertical threshold \mathcal{T} , the algorithm upgrades the vertical spatial scope of cl_i by adjusting the start and end zoom levels based on the current zoom and subtree levels (i.e., $X-1$ in this case) (Lines 12-15). Here, we consider the number of users to identify scams or inputs from virtual bots. This approach helps us in identifying on-going events such as an Opera or Concerts, where users attending the event are more proactive in updating their status from the same location as compared to sibling cells.
- For the horizontal spatial scope, if `aggCl[]` list size is more than a given threshold Γ , then the algorithm creates a new cluster `acl` in c at the level $X-1$ if the tree, by merging all cluster identifiers from the lower level, and by upgrading the spatial scope as well as the zoom start and end to be displayed on map. A new cluster location is then determined based on a weighted centroid of the different sub-clusters

forming acl. The new cluster will also have its top content updated by refining the top frequent words and topics from all related sub-clusters (Lines 16 - 22).

- Finally, the algorithm sets the flag for updated spatial scope as true so that non-visited clusters will be considered in the next rounds for further changes with respect to their vertical or horizontal scope (Line 23).

This algorithm is repeated from level X-1 until reaching the root level, so that each loop will consist in re-assessing the set clusters in a given subtree and if any merging is possible, new clusters at higher levels of the tree are created with an upgraded spatial scope. Consequently, the same detected event might be represented by different clusters that are encapsulated one within the other, and each one is given a relative scope so it appears on a specific range of zoom levels. For instance, on a map with zoom levels ranging from 1 to 20 (such as google maps), clusters that are found at the leaf level of the tree are shown on a zoom level from 18 to 20, and those that are aggregated together at higher levels can be displayed at various zoom level starting from 17 till level 3 or 4 (country and continent levels) depending on their significance.

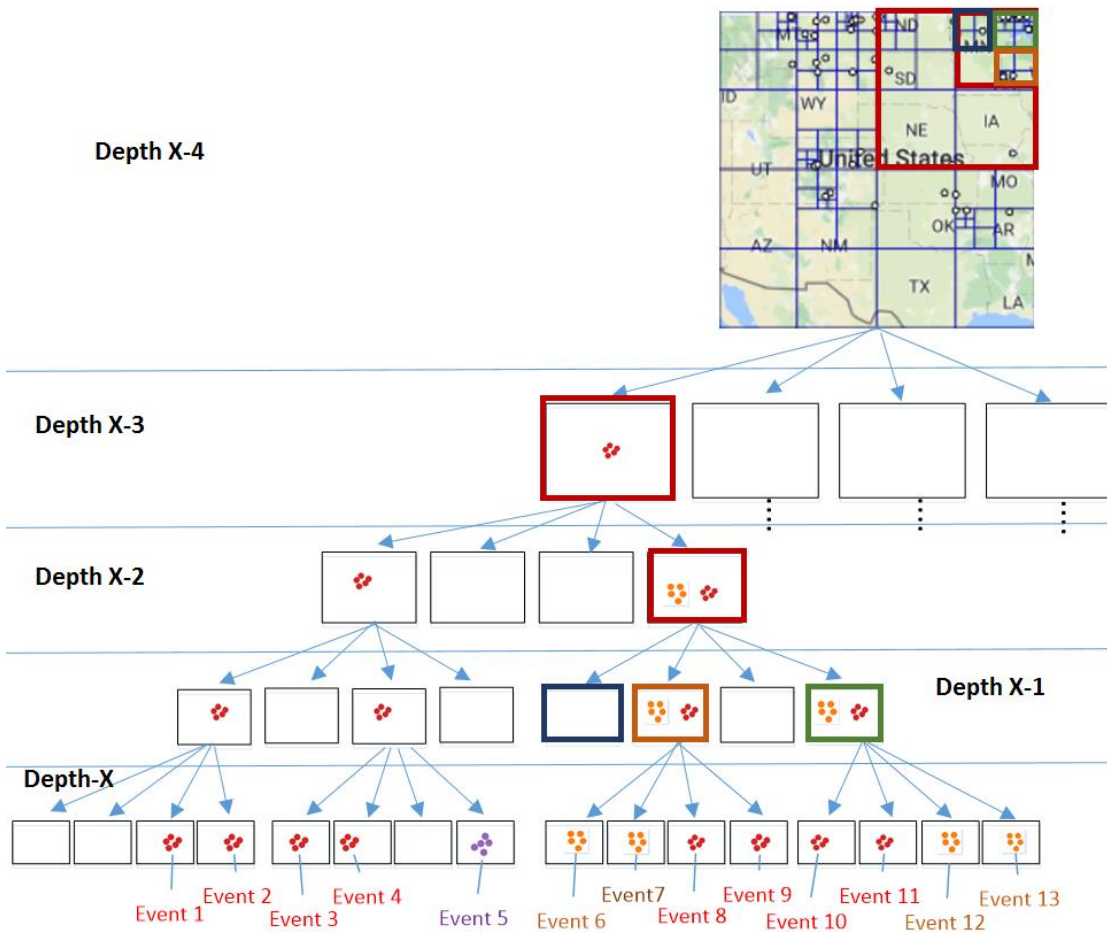


Figure 5-10: Sample horizontal spatial scope of events from Depth 'X' to 'X-4'

Figure 5-10 explains the concept of horizontal scope further by considering Event 1 to Event 13 in different cells (C_{X1} : C_{Xn}) at a depth level 'X' in the hierarchical tree. Event with same color represents same real world entity including red color events (Event 1, Event 2, Event 3, Event 4, Event 8, Event 9, Event 10, and Event 11), purple color event (Event 5), and orange color events (Event 6, Event 7, Event 12 and Event 13). At depth level 'X -1', we analyze child of each cells for merging of events, if two or more events in child cells are merged together then we update their horizontal scope to level 'X -1' from 'X' . At depth level 'X -1', we can see 6 events (4 red and 2 orange) after merging of events in their respective child cells (Event 1_2, Event 3_4, Event 8_9, Event 10_11, Event 6_7, Event 12_13). At depth level 'X -2', we can see 3 events (2 red and 1 orange) after merging of events in their respective child cells (Event 1_2_3_4, Event 8_9_10_11, Event 6_7_12_13). At depth level 'X -3', we can see 1 red color event after merging of events in their child cells (Event 1_2_3_4_8_9_10_11). The scope of red color event is highest (depth level 'X -3') and it is visualized on map from lower zoom

level as compared to orange color event (depth level ‘ \mathcal{X} -2’). Purple color event is not merged with any event so it will be visualized on map at higher zoom level as compared to orange and red color events.

5.9 Graphical Interface for Exploring Maps

The development of multi-resolution event enriched maps has encouraged new research on spatio-temporal queries. Traditional spatio-temporal queries are limited to dynamic and static information. Dynamic information includes traffic updates and social network data. Static information includes list of POIs visible on satellite, transport, and hybrid layers of maps. Both dynamic knowledge and static data are used by traditional routing algorithms for calculating the fastest path or to find nearby POIs. With development of the new application Hadath, it is possible to nuance traditional queries such as multimedia routing, safest routing, live events, and nearby POIs with dynamic information about offers, discounts, reviews etc. by leveraging geo-tagged EoIs into them. In this respect, there is a need to extend existing query language that answers with the assistance of multiple types of dynamic knowledge (EoIs) from multiple sources, including social network, open government data, traffic sensors, and weather updates. For example, consider a city dweller who is interested in current ongoing activities in her neighbourhood. She posts, let us imagine, the following query: “*Show safe and fastest path to find the nearest music concert with ticket discounts*”. This query can be made more interesting by including discount Italian pizza on the way or to find the events in place where the Air Quality Index is less than 75 for polluted big cities.

5.9.1 Overview of the System

Figure 5-11 shows an overview of the query interface that is implemented by leveraging the Hadath Framework. As discussed in the last chapter, Hadath takes social network data as an input and then converts un-structured data into generic data packets by removing noise and irrelevant data. Thereafter, the system processes data and stores it in the main memory temporarily so as to enable fast retrieval of recent data. Based on a predefined threshold value, a flushing process moves data from the main memory to the spatial database. A transcoder transforms multimedia data in different resolutions to support diversified user bandwidth and resolution. Finally, it stores the original and transcoded media in the spatial database. The PoIs so databased comprise of a list of PoIs with information extracted from Yelp, Booking.com, etc. The spatial weather database stores information regarding weather, including climate, air quality index, and recommendations. In the spatial database, we store routing data that is used by pgRouting for spatial queries; multimedia databases, however, store the geotagged audio, video, and image in different resolutions. The statistical database contains knowledge about crime in the region and accident prone roads by extracting open government data. Other databases have resolution type tables, data type tables, data source tables, and extracted multimedia data tables. Each multimedia data in a multimedia data table is labelled with the

nearby edge ID, cell ID, PoIID, EoIID, and data type. The query optimizer processes spatio-temporal queries by identifying keywords in queries with the relevant database: the engine executes the optimized plan that is generated by the optimizer. After this, the visualizer displays results of the spatio-temporal queries on a map, both online and offline. The system augments conventional routing with multimedia information integrated within the spatial query engine. Apart from the aggregated geotagged multimedia data, the system also uses road networks for calculating optimal routes. An overview of these different components is described below: results are tailor-made to users' smartphone bandwidth and resolution requirements.

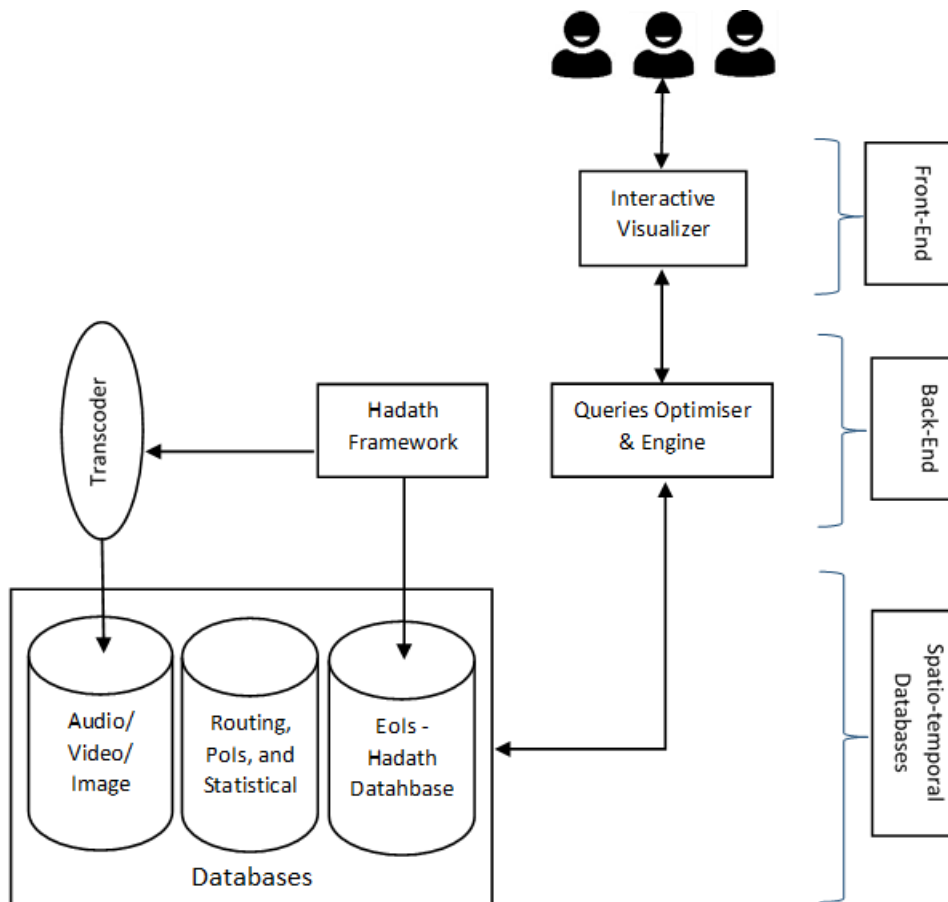


Figure 5-11: Overview of the Query Interface System

5.9.2 Database Schemna

Figure 5-12 shows an overview of tables in the database. These tables allow queries to be answered. Node and Edge tables are used for routing so as to answer optimized, multimedia, and safe routing queries. To make an efficient index, the concept of grid has been introduced in table cells. This will allow EoIs, PoIs, multimedia data, crime, and accident cases to be indexed. It will also decrease the time required to identify upcoming affected users by looking for the trajectory of the users in same cell or, at most, in neighbour cells—i.e., the Cell Neighbour table. This cell approach also reduces the time to calculate optimized path, because it calculates the time taken to cross the cell in advance. Outputs of the pre-processed path are stored in the Path Storage Table. Multimedia tables store multiple copies of similar types of multimedia data based on resolution type, screen size, and bandwidth of the internet speed using the help of transcoders. This is necessary to provide tailor-made results to users based on their smartphone bandwidth, screen size, and resolution. The information of bandwidth type, resolution type, and screen size type are stored in the Bandwidth Type, the Screen Size Type, and the Resolution Type tables respectively. The matching of bandwidth, screen size, and resolution type are stored in the RSB Matching Table. The PoI and EoI tables store information on POIs and EoIs. The PoIType and EoIType tables store icons to visualize different types of POIs and EoIs on map. The Multimedia Type table stores information about the type of multimedia, such as audio, video, and image.

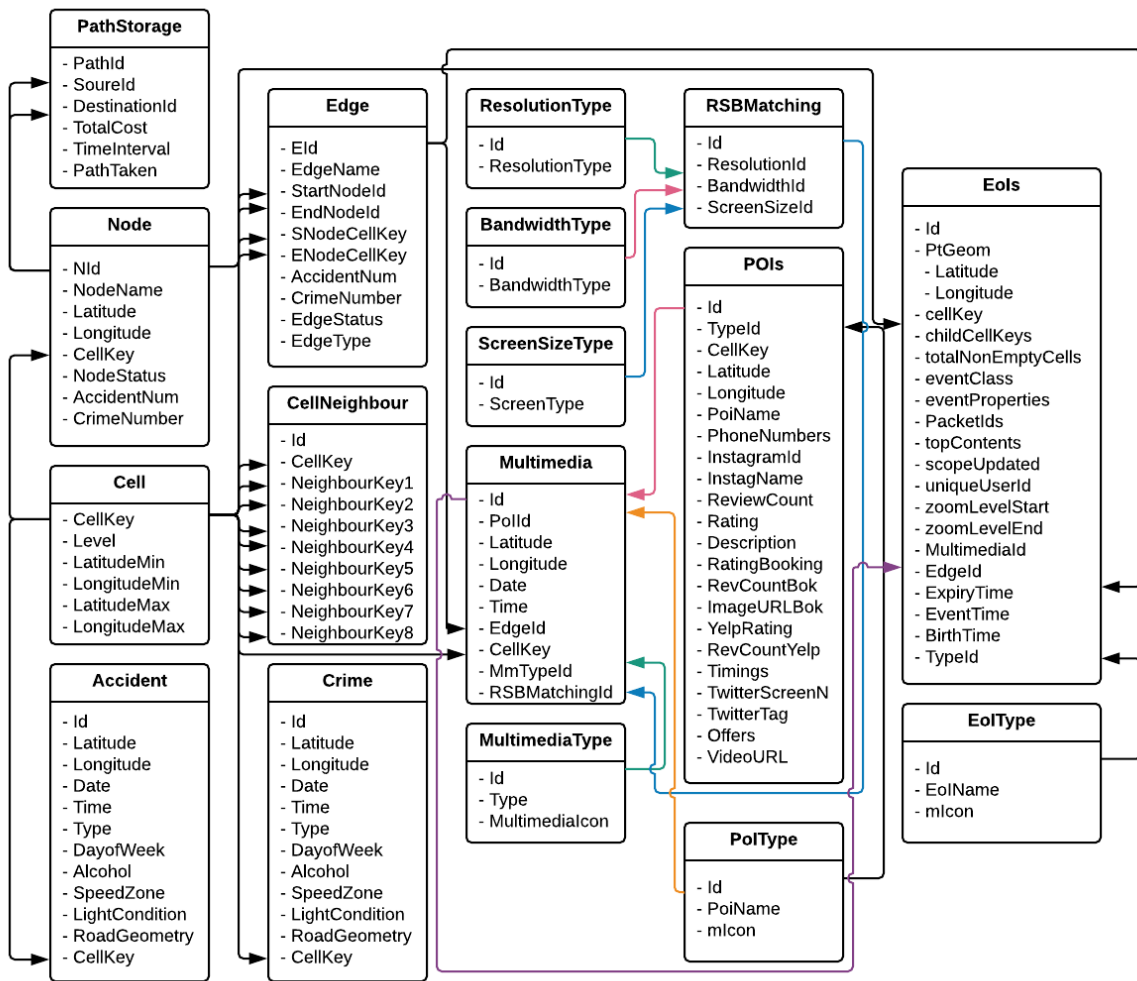


Figure 5-12: Database Schema

Figure 5-12 shows an overview of tables in the database. These tables allow queries to be answered. Node and Edge tables are used for routing so as to answer optimized, multimedia, and safe routing queries. To make an efficient index, the concept of grid has been introduced in table cells. This will allow EoIs, PoIs, multimedia data, crime, and accident cases to be indexed. It will also decrease the time required to identify upcoming affected users by looking for the trajectory of the users in same cell or, at most, in neighbour cells—i.e., the Cell Neighbour table. This cell approach also reduces the time to calculate optimized path, because it calculates the time taken to cross the cell in advance. Outputs of the pre-processed path are stored in the Path Storage Table. Multimedia tables store multiple copies of similar types of multimedia data based on resolution type, screen size, and bandwidth of the internet speed using the help of transcoders. This is necessary to provide tailor-made results to users based on their smartphone bandwidth, screen size, and resolution. The information of bandwidth type, resolution type, and screen size type are stored in the Bandwidth Type, the Screen Size Type, and the Resolution Type tables respectively. The matching of bandwidth, screen size, and resolution type are stored in the RSB Matching Table. The PoI and EoI tables store information on POIs and EoIs. The PoIType and EoIType tables store icons to visualize different types of PoIs and EoIs on map.

The Multimedia Type table stores information about the type of multimedia, such as audio, video, and image.

5.9.3 Use Cases

Query interface on multi-resolution enriched maps gives users options to explore maps efficiently. Figure 5-13 shows an overview of the query interface: this allows users to select an area on the map by drawing a square and selecting events and their scope by specifying time and date on historical or live data. The interface allows users to visualize results on a map or as a word cloud or as a list. There is also an option to visualise the multimedia images or video along with results. Figure 5-14 illustrates a list of specified EoIs in drop-down format: users can then also select ‘Other’ option to mention any unspecified EoI. Figure 5-15 provides an option to handpick the granularity of the event, such as high-level events that are important at country or neighbourhood level. If users are interested in multiple events, they can specify the same by selecting options such as “Less than”, “More than”, or “Equal”. Figure 5-16, Figure 5-17, Figure 5-18, and Figure 5-19 demonstrate an interface for queries on routing, city exploration, evacuation in case of any disaster, and planning for trips. As compared to traditional queries, these queries leverage extracted knowledge of multi-resolution event enriched maps to give new dimension such as show all the events at country level or show all the events that are more than equal to city level. Result of some of the queries are illustrated in Chapter-6.

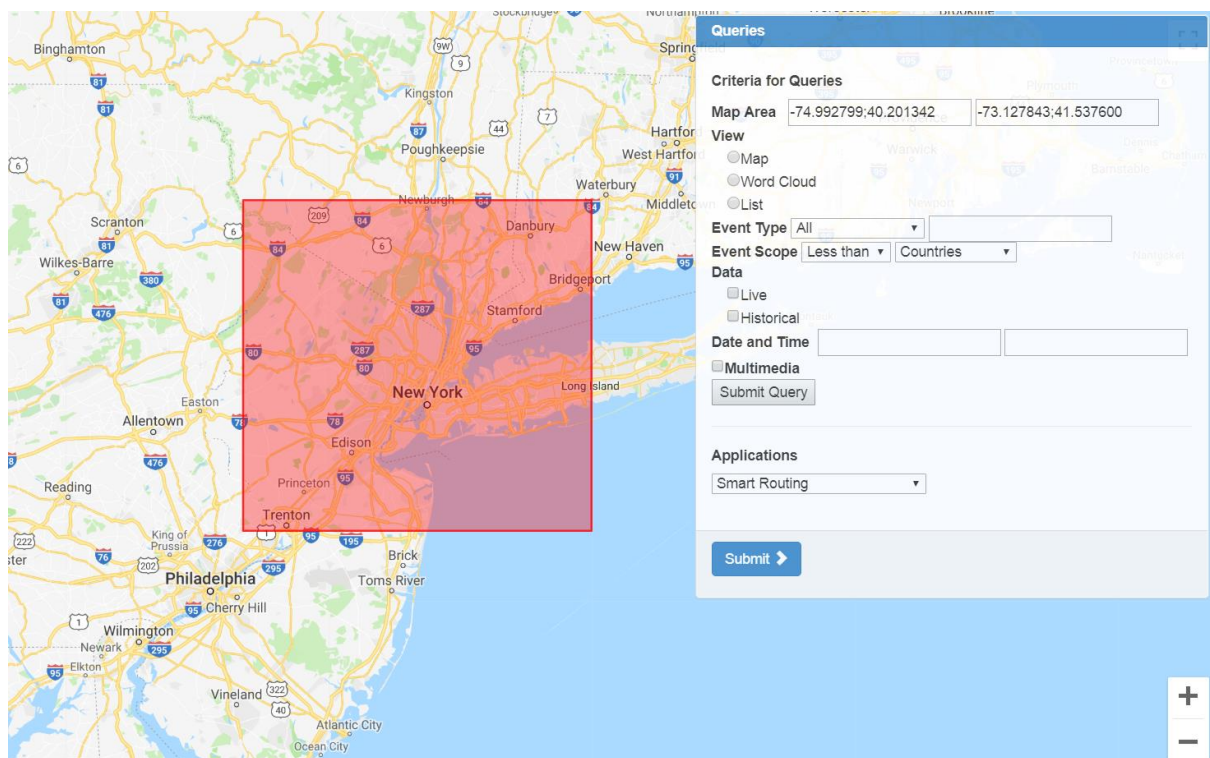


Figure 5-13: Query Interface Overview

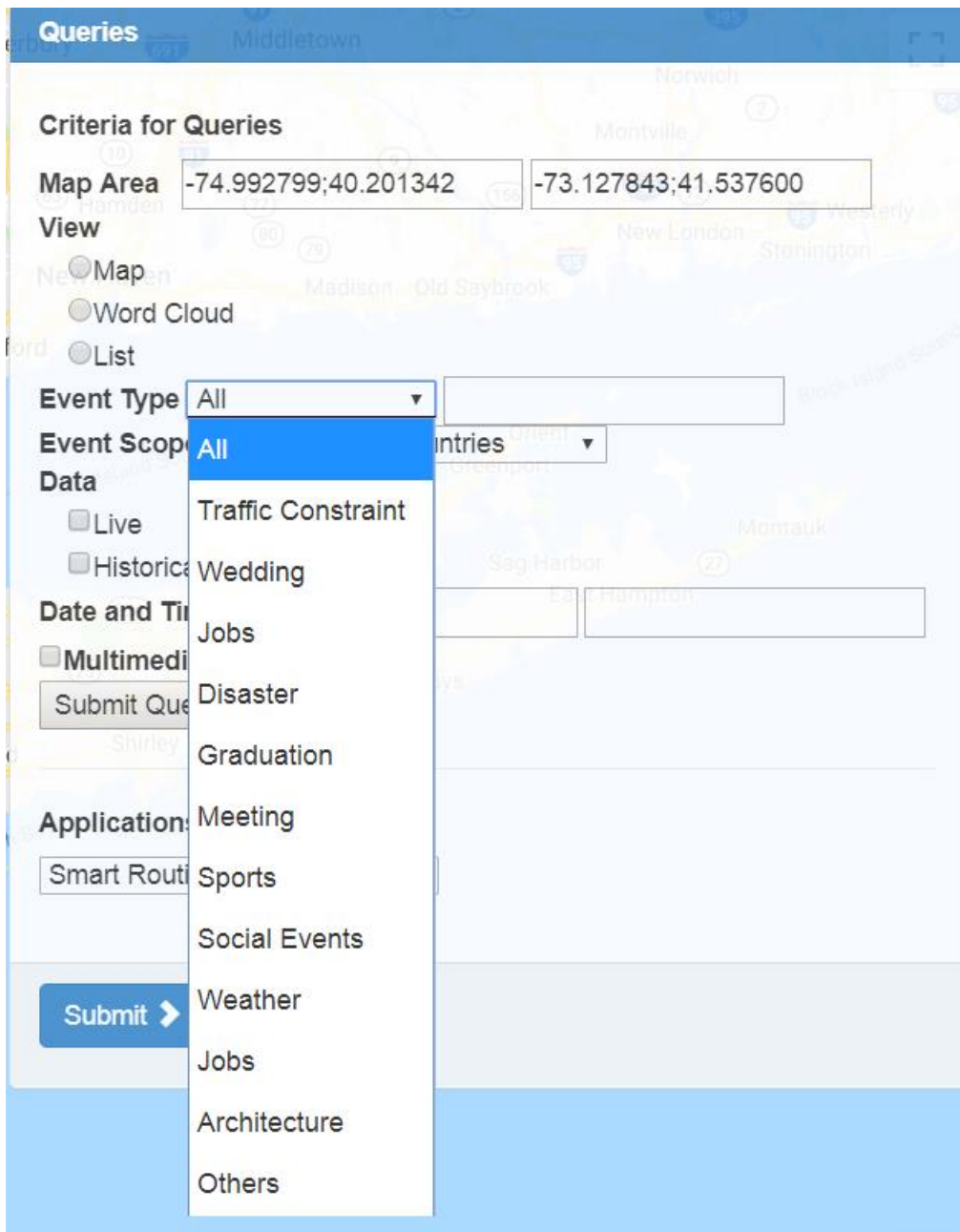


Figure 5-14: List of some EoIs with an option to write in inbox in case of other type of EoIs

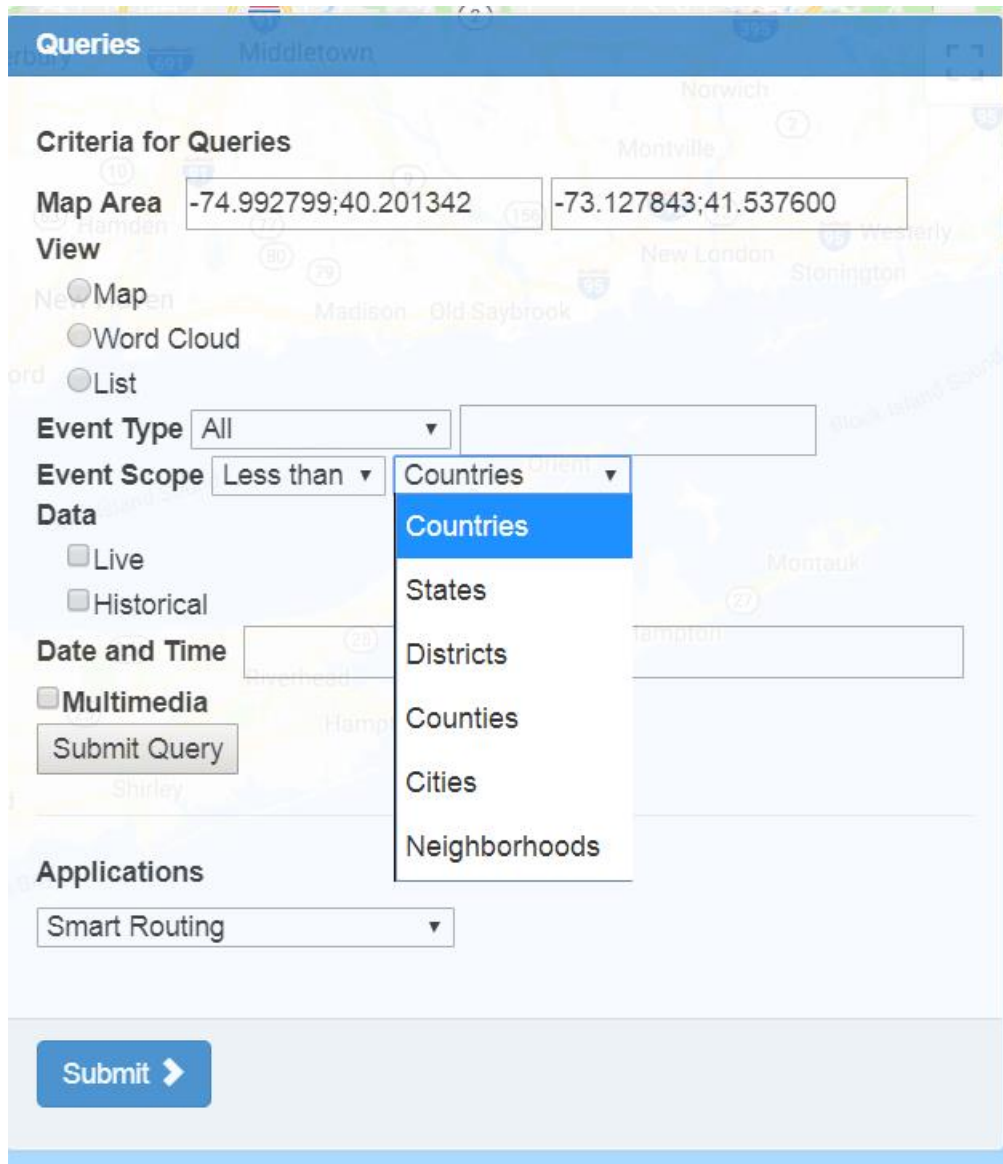
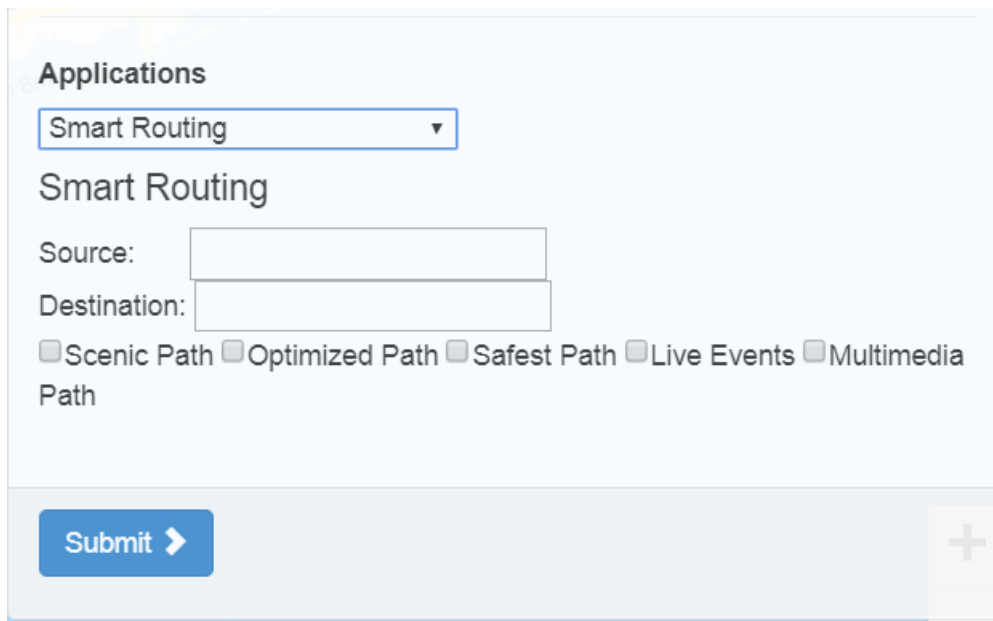
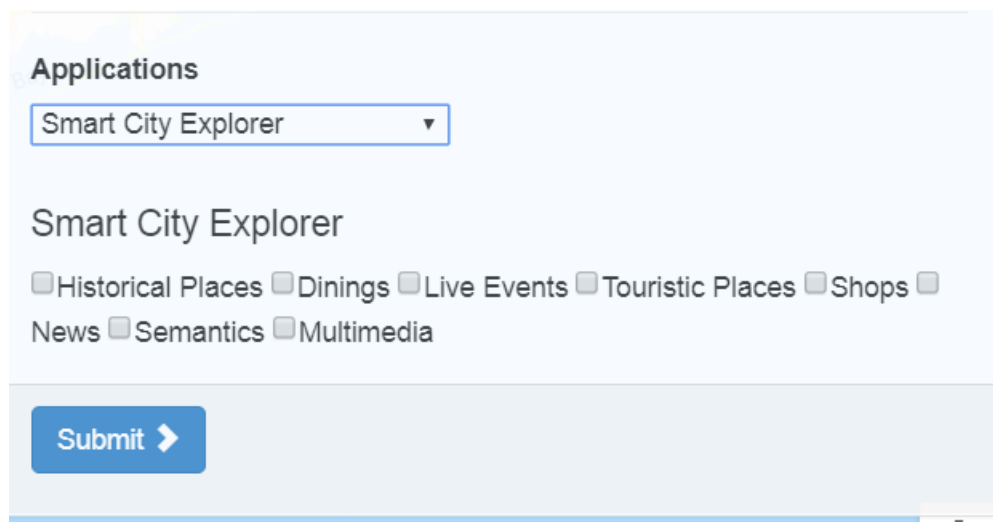


Figure 5-15: Event scope to select importance of different level with an option to select “Less than”, “More than”, and “Equal”



The interface for the Smart Routing application. It features a dropdown menu under the heading "Applications" with "Smart Routing" selected. Below this, the text "Smart Routing" is displayed. There are two input fields: "Source:" and "Destination:". Below the input fields, there are five checkboxes: "Scenic Path", "Optimized Path", "Safest Path", "Live Events", and "Multimedia Path". At the bottom left, there is a blue "Submit" button with a right-pointing arrow. At the bottom right, there is a light gray button with a plus sign.

Figure 5-16: Smart Routing Query Interface



The interface for the Smart City Explorer application. It features a dropdown menu under the heading "Applications" with "Smart City Explorer" selected. Below this, the text "Smart City Explorer" is displayed. There are two rows of checkboxes: the first row contains "Historical Places", "Dinings", "Live Events", "Touristic Places", and "Shops"; the second row contains "News", "Semantics", and "Multimedia". At the bottom left, there is a blue "Submit" button with a right-pointing arrow. At the bottom right, there is a light gray button with a plus sign.

Figure 5-17: Smart City Explorer Query Interface

Riverhead
Hampton Bays

Applications
Smart Emergency Evacuation ▾

Smart Emergency Evacuation

Search City...

Prepared Scenarios
▾

1. Evacuation Sources:

Max Lat;Lng

Min Lat;Lng

2. Shelters:

Lat;Lng +

Capacity

Submit ▶

Map data ©2018 Google | Terms of Use

Figure 5-18: Emergency Evacuation Query Interface

Hampton Bays

Applications
Smart Trip Planning ▾

Smart Trip Planning

Source:

Add Destinations:

Date:

Scenic Path Optimized Path Multimedia Path Notifications

Submit ▶

Figure 5-19: Smart Trip Planner Query Interface

In future, we will setup a new query language based on this interface. The query language will enhance existing spatio-temporal queries and provide results with updated knowledge and EoIs.

5.10 Conclusion

This chapter introduces *Hadath*, a system that builds event-enriched maps by handling social data streams, and by developing different algorithms for the efficient extraction, clustering, and mapping of live events. Hadath wraps incoming unstructured data streams into data packets, that is, a generic structured format of a potential event. These packets are then processed to extract EoIs based on a hierarchical clustering technique, which defines the spatio-temporal scope for each event. The technique defines the spatio-temporal scope of each event by using an in-memory spatial indexing scheme. The spatio-temporal scope of EoIs so created leads to a unique and dynamic map browsing experience: such a mapping system has the potential to become a city search engine, where EoIs can be identified at different scales of relevance for city dwellers as well as visitors/tourists. The system can provide valuable knowledge from crowd-sourced data to authorities, market firms, event organizers, and end-users to help in decision making. Later, we explored knowledge extracted from framework and other sources using graphical interface for exploring maps.

In future, we plan to merge more data sources (e.g., Instagram, online newspapers) to increase correctness and conciseness of detected events. Nonetheless, we believe our system is part of the ongoing change ushering in the next generation of map platform which will intelligently extract EoIs along with their scope.

Chapter 6

HADATH: System Implementation and Results

6.1 Introduction

Hadath is a proof-of-concept implementation to demonstrate the feasibility and adaptability of building *Event-enriched Maps* by displaying real-time events and findings. Figure 5-2 shows an overview of our proposed system architecture with the salient components. The current implementation is not real-time, and uses three months of Twitter data for the whole world.

6.2 Implementation

To validate our approach, Hadath was developed as a proof-of-concept system based on a big data framework towards efficient data management and visualization of events of interest on maps. Our methods were tested with more than 30 million geotagged tweets. The front-end is a map-based application that visualizes spatio-temporal events at different scales. The implementation is done in Java 1.7. We are using i7-4712 HQ-CPU @ 2.30GHz with 16GB DDR-2 RAM at the back-end for processing using the following libraries and software [113].

- *osm2pgsql*: The Open Street Map component contains planet dump data that is converted into PostGIS datasets by using the *osm2pgsql* [106] tool.
- *PostGIS*: We installed PostGIS for spatial query over PostgreSQL to store road network data, POIs extracted from Yelp, Booking.com and OSM with their semantics.
- *NLP*: We used Ark-tweet-NLP [107] library for part-of-speech and annotation. This library is trained for Twitter and produce better results than Stanford NLP. It takes care of the out-of-vocabulary words, and stop words used in Twitter.

- *Clustering Algorithm*: Data packets with a high text similarity value are clustered using louvain clustering algorithm [112]. The Louvain is suitable in our approach as, unlike most of the other clustering methods, it does not require a prior knowledge of the minimum number of clusters.
- Taghreed Crawler [5]: The same crawler was used to collect geotagged tweets.
- Other libraries: Apart from above, we used Jackson JSON and google maps APIs in our system. Back-end implementation is done in Java 1.7.

Number of potential indexed packet at precision 5 level

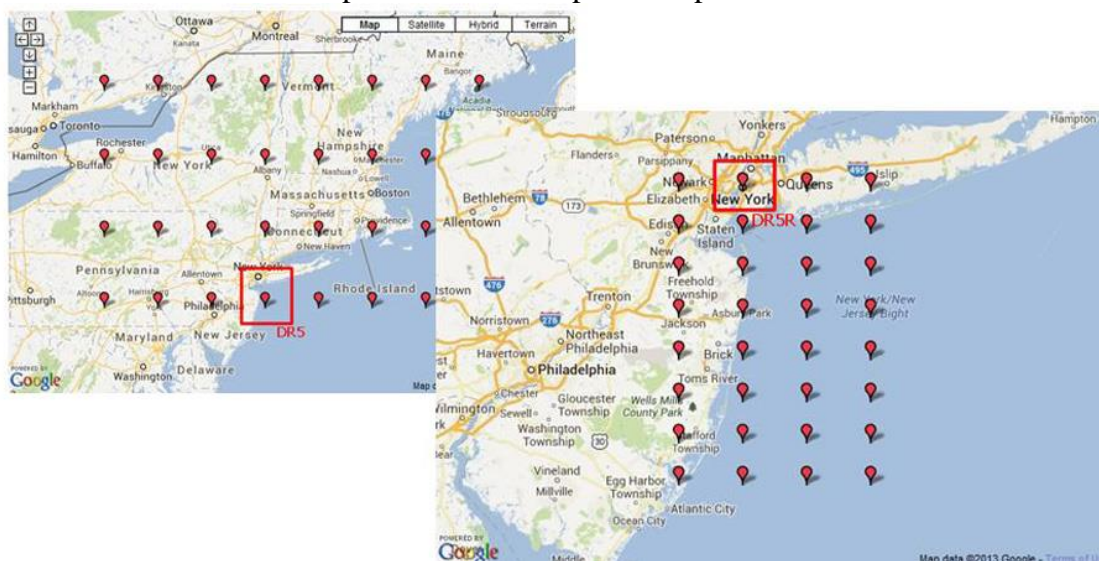


Figure 6-1: Shows view of aggregated geo-clusters with 3 character 'DR5' and 4 character 'DR5R' of indexed tree

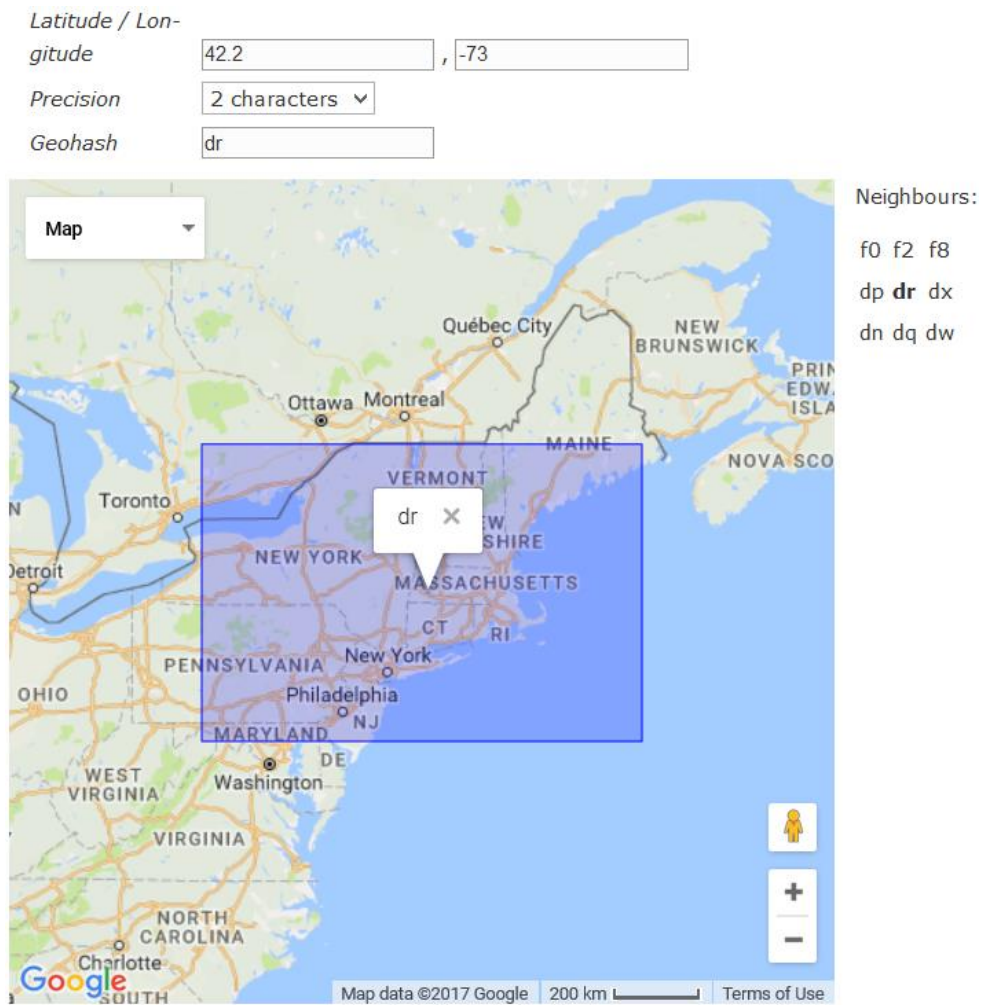


Figure 6-2: Map view of aggregated geo-cluster with location string of 2 character and precise location of 1,250km * 625km.

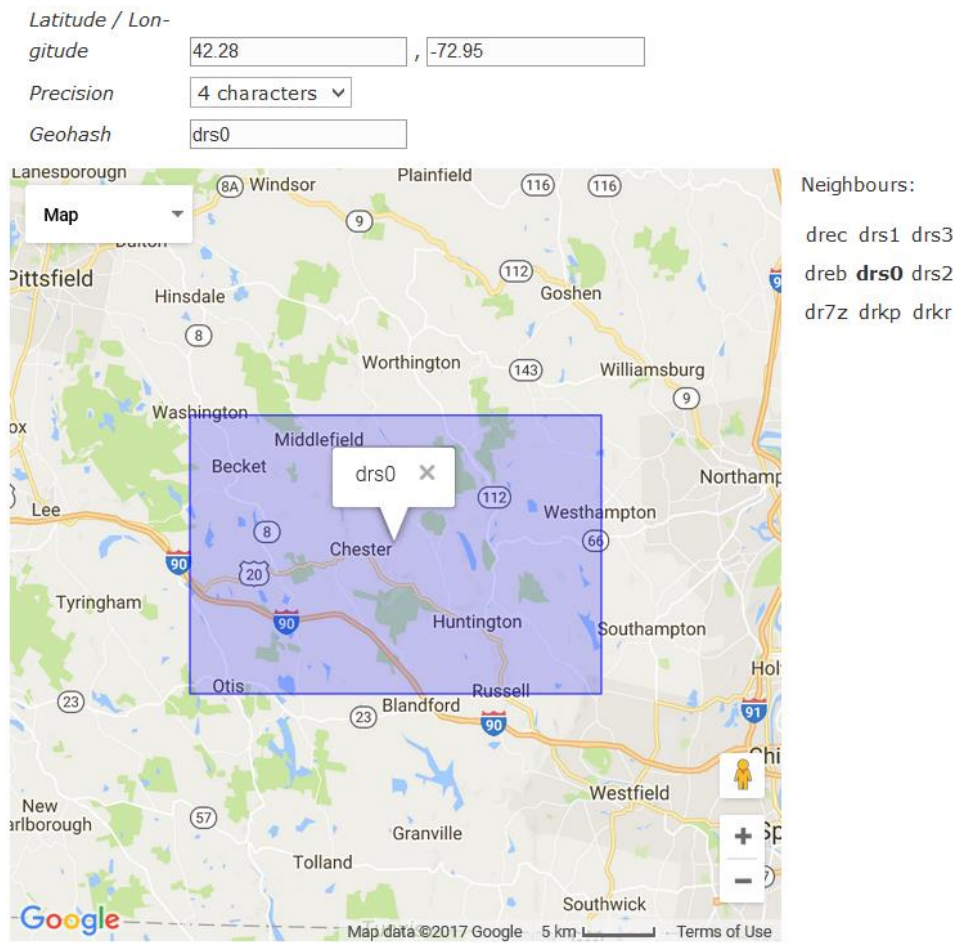


Figure 6-3: Map view of aggregated geo-cluster with location string of 4 character and precise location of 39.1 km * 19.5 km.

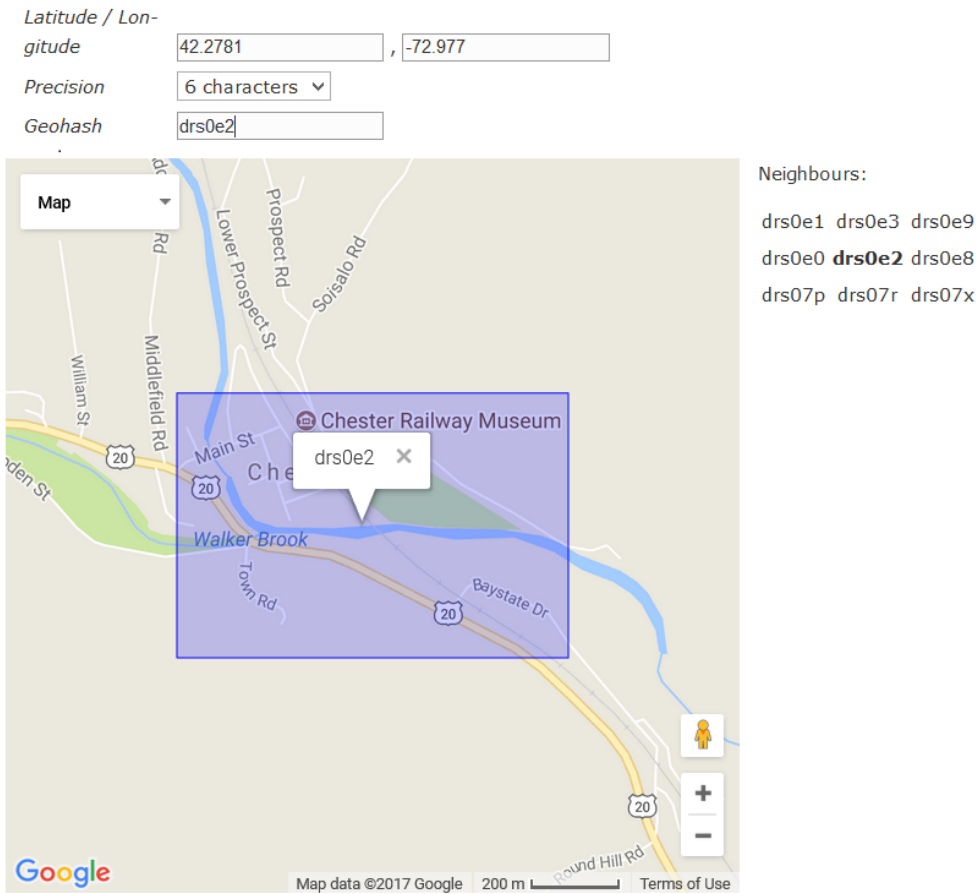


Figure 6-4: Map view of aggregated geo-cluster with location string of 6 character and precise location of 1.22km * 0.61km.

- **Big Data:** Summarizing data into squeezed information is very important as computation of high-performance geospatial data requires a lot of data operations. Therefore, it is mandatory to take advantage of a distributed processing environment while maintaining an in-memory indexing scheme to build an efficient and scalable system. Aggregating geospatial data is an effective summarization technique for visualizing geospatial data on maps. Summarization will not only reduce the overall processing cost, but will also effectively handle the bandwidth because of a huge data transmission.
- **Geo-Spatial Data aggregation:** The Grid-based and Distance-based clustering represent the two commonly used techniques for geo-clustering in online maps [114]. The Grid-based clustering works on the principle of a latitude, longitude geocode system, referred to as 'Geohash', which is designed by Gustavo Niemeyer. Therefore, the principle of grid-based clustering is dividing world map into rectangular cells, and then grouping data points within each cell hierarchically. Geohash has a character value that helps in postulating the accuracy of the hash value and determining the location. For example, the latitude and longitude coordinates of 42.2, -73 falls within the geohash box of 'DR'

and it is a part of New York City, USA. Adding a character to the string 'DR' will lead to more specified geographical subsets of the original string [115]. Figure 6-1 shows view of aggregated geo-clusters with 3 character 'DR5' and 5 character 'DR5R' of indexed tree. Figures (Figure 6-2, Figure 6-3, and Figure 6-4) further explains different map views with 2, 4 and 6 characters of geohash strings and their corresponding zones on map. Therefore, different levels of the geohash value are used by the grid-based clustering technique to group data points located in nearby cells.

One of the advantages of the geohash technique is that it translates two-dimensional spatial queries into one-dimensional string search. Therefore, it can solve search queries with $O(1)$ time complexity. The length of the geohash string is considered as the precision level for a specified zone. As the geohash strings are shortened, less precise zones are covered.

For real time processing and analytics, Elasticsearch is used to index and store the different clusters of summarized data at different zoom levels. The data summarization layer processes data points and classifies them into geohash clusters. Using the live data streams, aggregated live geohash clusters are prepared for each zoom level.

GeoHash length	Area width x height
1	5,009.4km x 4,992.6km
2	1,252.3km x 624.1km
3	156.5km x 156km
4	39.1km x 19.5km
5	4.9km x 4.9km
6	1.2km x 609.4m
7	152.9m x 152.4m
8	38.2m x 19m
9	4.8m x 4.8m
10	1.2m x 59.5cm
11	14.9cm x 14.9cm
12	3.7cm x 1.9cm

Figure 6-5: Size of Cells at Different Level of Spatially Indexed Tree

-
- **In-Memory Indexing and Storage:** On the top layer, Elasticsearch is used for in-memory data storage. The repository contains information related to real-time data. Elasticsearch has strong indexing model which makes it efficient for data retrieval. It is also used for in-memory storage for most recent live data, for fast access. We use Elasticsearch for both indexing and storing the most recent zoom level summarised data for real-time analytics.
 - **Persistence Memory:** Second level of repository layer don't have any role in real-time processing and persistence as it contains historical records in persistence memory which later used for offline analytics on historical data. It contains most recent records of defined set of the period (i.e. last one month). The most recent historical records are retrieved efficiently at this level of the repository. In real scenarios, most of the queries are related to recent information, so, having most recent historical records makes the system most robust.
 - **Archived Data Storage:** Hadoop/HDFS is the second level of repository, which archives historical and recent data sets. Unlike real-time processing, the second repository layer stores historical records in persistent memory and is used for statistical analytics over a specified spatio-temporal window in the past. It is effectively used for huge volume information retrieval. The Hadoop/HDFS ecosystem has proved to be efficient when the volume of historical data is huge [116]. Moreover, the recovery of data is very efficient because of the distributed storage and processing mechanism.
 - **RabbitMQ Server 3.4.4:** We deployed the RabbitMQ server as RAM node for large in-memory operations and to achieve efficiency for both storage and retrieval of data streams. On the other hand, there is a probability of data loss in case of node crash. We used Dell XPS 8700 4th Generation Intel® Core™ i7-4790 processor (8M Cache, up to 4.00 GHz) with 32GB Dual Channel DDR3 1600MHz - 4 DIMMs and 1TB 7200 RPM SATA Hard Drive 6.0 Gb/s + 256GB SSD.
 - **ES-Hadoop 2.4:** We set up a special version of Hadoop for Elasticsearch named as ES-Hadoop. Elasticsearch for Apache Hadoop (ES-Hadoop) is the two-way connector that solves a top wishlist item for any Hadoop user using real-time search. ES-Hadoop bridges that gap using Hadoop's big data analytics and the real-time search of Elasticsearch. We deployed two nodes: DataNode (for data storage) and TaskTraker (for data processing). We used following machine specification for Hadoop deployment. 2x1TB hard disks in a JBOD (Just a Bunch Of Disks) configuration and 2 quad-core CPUs, running at 2GHz and 32GB of RAM.
 - **Elasticsearch 2.4:** We deployed three nodes for elasticsearch, one as a server node and two data nodes. We also maintained good RAM oriented machines because elasticsearch operations for both sorting and aggregation are processed in memory. We used Dell XPS x8900 machine as Master node consists of 6th Generation Intel Core i7 processor with 3TB 7200 RPM SATA Hard Drive 6.0 Gb/s + 256GB SSD and 64GB Dual Channel DDR3 1600MHz - 4 DIMMs. For data nodes we used OptiPlex 790, 4th Generation Intel® Core™ i7-4790 processor (8M Cache, up to 4.00 GHz) with 1TB 7200 RPM SATA Hard Drive 6.0 Gb/s + 256GB SSD.
 - **Logstash:** Logstash works with elasticsearch and used along with each elasticsearch node as data in and out pipes.

6.3 Results

Type	Tweet ID	Potential EoI	Tweet
Specified	744497060884414465	Sports/ Badminton	I just finished 7m:23s of playing badminton with #Endomondo #endorphins https://t.co/Yc7ZK3roPj
Specified	744497085592928256	Social Event/ Birthday	Happy Birthday my omet partner and my team Imat MDC.. (w/ Pradita, imat, 2 others at Cafe Just Update) [pic] https://t.co/EqZmhrcHTE
Unspecified	744497121680711681	Father's Day	Good morning everyone Happy Father's Day to all the deserving https://t.co/W8dJJBZtDx
Unspecified	744497139343044608	Iowa Craft Brew Festival	3.7 #ICBF Actually not bad. - Drinking a Biergarten Tart by @Leinenkugels @ Iowa Craft Brew Festival (2016) https://t.co/m0WS5qDHQk
Specified	744497356624781312	Public places/ camping	So our next camping trip aint gt nuthin on us.....woza Swaziland @ https://t.co/CtcQrShc5J
Unspecified	744497419677667328	Closed services	Closed for services today, so this is as close as I got. Top of 2026 https://t.co/NE0kpnYnww
Specified	744497396084727808	Natural locations/ beaches	White sand beaches are :-) @ Bolod Beach Resort, Panglao Bohol https://t.co/X1umck6sBG
Specified	744497437721538561	Commercial places/ music	Live music in Temple Bar with @kevv_f @ The Temple Bar Pub https://t.co/C1X8airMVM
Specified	744497474383929344	Jobs	Jobs Want to work in #Alpharetta, GA? View our latest opening: https://t.co/HTHMTTKp8b #Construction #Job #Jobs #Hiring #CareerArc
Specified	744501233507934208	Weather	Storm in Bastrop CO moving towards 35 w/heavy rain, gusty winds, lots of lightning. atwxw https://t.co/XRCkjg41RW
Unspecified	744497581787537408	Street	Just posted a photo @ Maginhawa Street https://t.co/Y16WVIJrQ5
Unspecified	744497673479290880	Public Places/ Town	I'm at @GurneyParagon in George Town, Penang https://t.co/VUzExmps1I

Specified	744497906690854912	Commercial Places/ Hotels	Discover hotels around Tawau, Malaysia from 13 EUR per night: https://t.co/nt76ZX2IBg https://t.co/Zocxo2m9gs
Specified	744499301133455361	Incident/ construction	Construction on #5Line Both directions from Bowling Green Station to 42nd Street-Grand Central Station https://t.co/vereT4w2Ux
Specified	744499906040078338	Social events/ concert	Today is the WI Brass Quintet's concert! Come on down to Saint 2026 https://t.co/VQ8y6RCMTQ
Specified	744499995710210048	Social events/ Forum	Checking out the Norwich Food; Drink Festival. (@ The Forum in Norwich, Norfolk) https://t.co/OPmsq1S8n2
Specified	744500326665977856	Social events/ concert	Special Event on #NY23A Both directions from Old Kings Road; CR 47 to NY 32A; Palenville https://t.co/zGEP38g1Ug
Specified	744500881404506112	Social events/ wedding	Happy wedding :) (with Daniel, David, and 4 others at Ball Room Hotel Istana Nelayan) [pic] 2014 https://t.co/flRjReJyiA
Specified	744501200742080513	Killed	JUST IN: Teen killed in Rowan County crash, officials say. Latest on Eyewitness News. @wsocvtv https://t.co/FnbesqYCIU
Specified	744501256865996800	Opening	So we're opening up the studio next Saturday! Lots of stuff going on, live screen printing, Riso2026 https://t.co/qIoBaghVBx
Specified	744504007406690304	Earthquake	USGS reports a M1.15 #earthquake 13km W of Salton City, California on 6/19/16 @ 12:13:15 UTC https://t.co/uuKMTvhZcl #quake

Table 6-1: Some potential EoIs that are detected from window-based bulk of tweets of 19-June-2017 (11:48 am to 12:14 pm)

Table 6-1 shows list of some specified and unspecified potential EoIs that are extracted from one window-based bulk of tweets of 19 June 2016 between 11:48am to 12:14pm. Specified are the ones that are matched with our existing EoI corpus and unspecified are the ones that are detected due to high peak of words that people are talking more than specific threshold in a given window-based bulk of tweets as shown in Figure 6-6. These are the outcomes of data cleaner and wrapper module by filtering and removing noisy data.

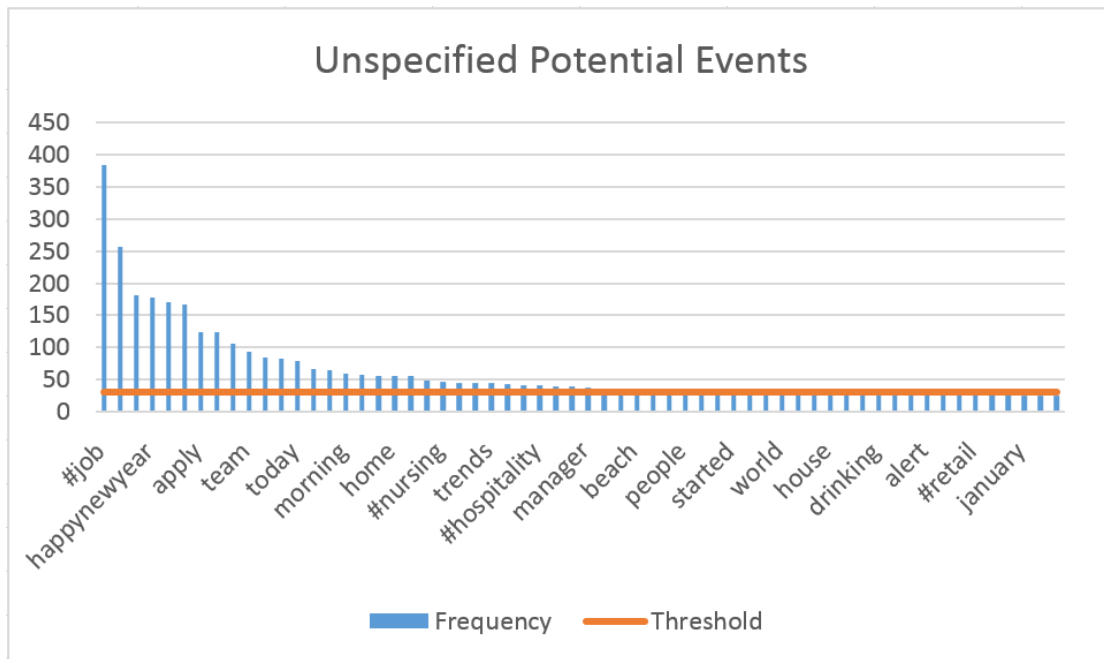


Figure 6-6: List of Unspecified Words with Number of Occurrence in One Batch of Tweets

Figure 6-7 shows a query to fetch EoIs at zoom level 17 and a sample result to demonstrate structure of EoI cluster. In query result, we can see that it took 179 milliseconds to return 1,35,681 EoIs using 28 shards with 100% success rate. Each EoI contains, name of index as ‘_index’, properties of events as ‘eventType’, total number of packets and their identifiers involves in clustering as ‘packetcount’ and ‘packets’ respectively, time stamp of EoI as ‘@timestamp’, identifier of clustered EoI as ‘id’, zoom level details of map as ‘zoomStart’ and ‘zoomEnd’, geohashing index of cell as ‘cellkey’, centroid location of a cluster by taking into consideration location of all packets in clusters as ‘location’ and a flag that is use to calculate vertical spatial scope as ‘visited’.

```
GET /eventtwitter*/_search { "query" : { "match":{"zoomStart":17} } }
{ "took": 179, "timed_out": false, "_shards": { "total": 28, "successful": 28,
"failed": 0 }, "hits": { "total": 135681,
"hits": [ { "_index": "eventtwitter-2017.01.03", "_source": {
"eventType": {"ogun": 9, "wife": 9, "husband": 9, "accident": 9},
"packetcount": 9, "title": "Incident/accident/",
"packets": "[\"816030466696310784\", \"816030470894796800\"...]",
"content": "[\"Husband, wife, son die in Ogun auto accident\"...]",
"@timestamp": 1483391799000, "id": "816030466696310784",
"zoomStart": "17", "zoomEnd": "18", "cellkey": "s14mh",
"location": { "lon": 3.3792057, "lat": 6.5243793}, "visited": false
}, ... , ] } }
```

Figure 6-7: Query and a sample EoI of zoom level 17

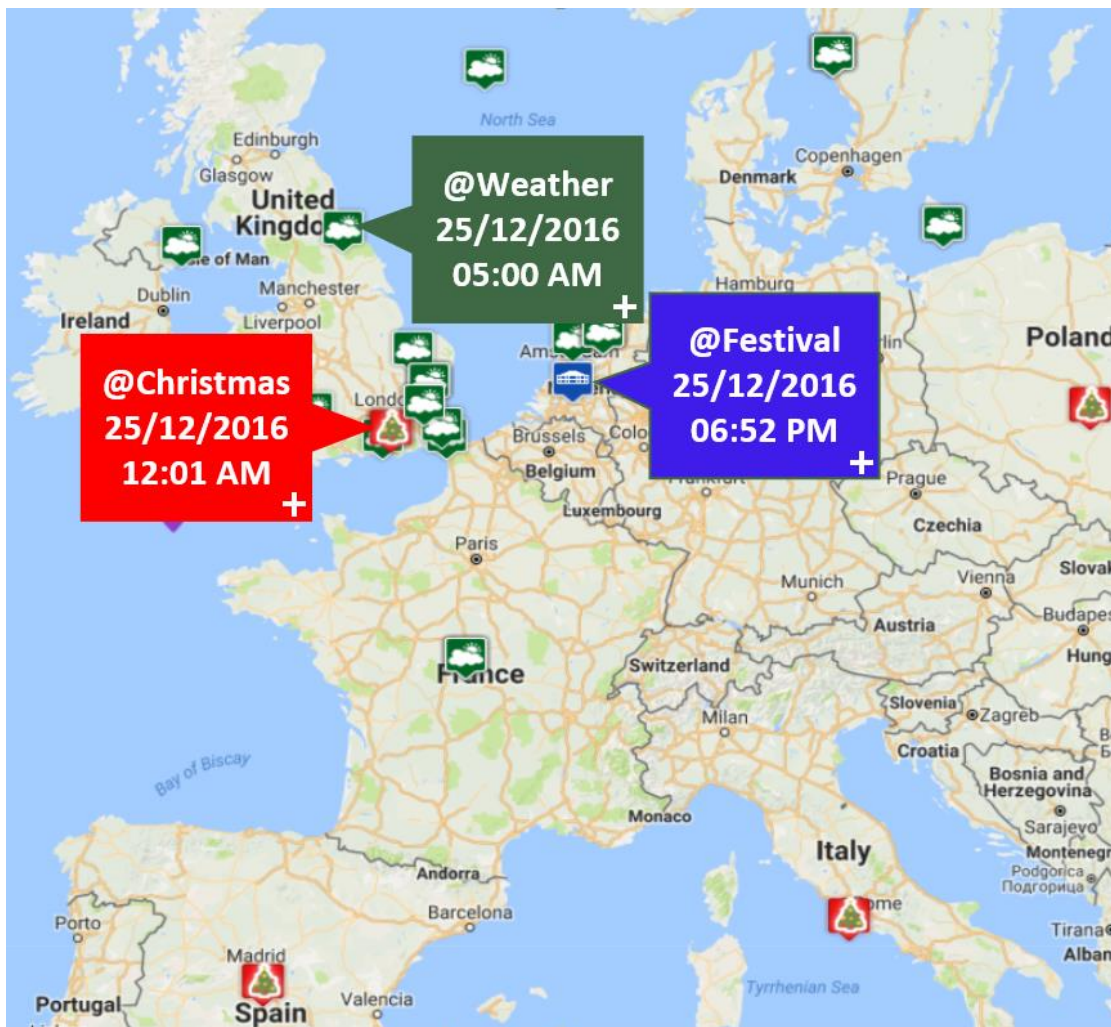


Figure 6-9: High Abstraction Map View with Country Level Social Events

In Figure 6-9, we can see map of Europe from low zoom level which shows Christmas and Festival cluster of EoIs at country level apart from weather updates that are from hundreds of bots in Europe tweeting about weather update within fixed interval of time.



Figure 6-10: Detailed Map View for the City of New York

Figure 6-10 shows high zoom level of New York city of neighbourhood level, where one can find local EoIs such as traffic accident on '#1 Line NB 50th street', jobs hiring in 'Barista', a day party at 'attic roof top lounge', event in 'gershawin theater' and one unspecified event 'Carolines Broadway Video' which is detected based on keyword peaks as users' are posting too many geotagged tweets. 'Carolines Broadway Video' is not in our event corpus but based on the frequency of the words, our hadath assumes it as EoI.

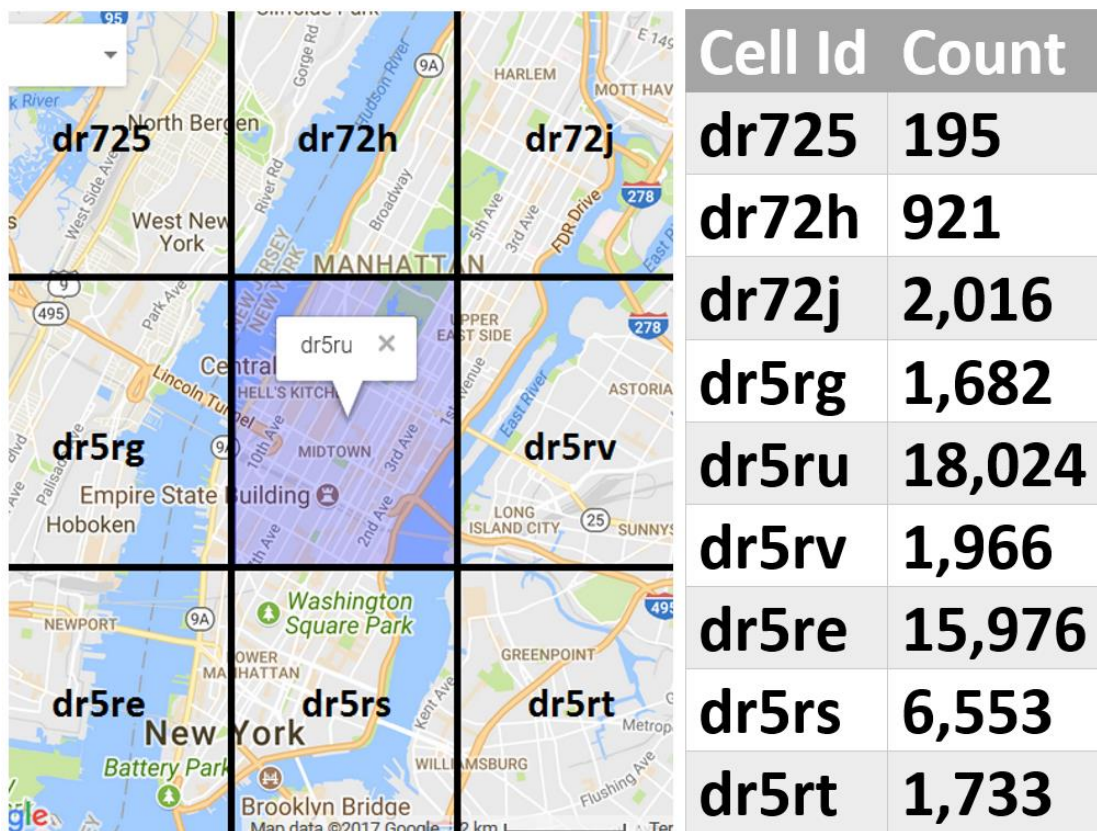


Figure 6-11: Number of potential indexed packet at precision 5 level

6.4 Evaluation

Data Wrapping and Cleaning: Major tasks for the data cleaner and wrapper are: 1) to clean irrelevant fields and digest incoming streams into a unique data packet format; 2) to use specified string matching technique that detect and match candidate packets with our event classifier corpus in order to identify potential event classes and properties. We used Ark-tweet-NLP [107] library for part-of-speech and annotation. This library is trained for Twitter and produce better results than Stanford NLP. It takes care of the out-of-vocabulary words, and stop words used in Twitter; and 3) to apply unspecified topic detection method that extract spatio-temporal peaks and unusual happenings based on the top frequent words. This approach helps us to detect unknown events that are not a part of corpus but their value are more than threshold

at given time. Figure 6-12 shows processing time for generating data packets after cleaning and wrapping of 15 batch files with each file containing 100,000 tweets.

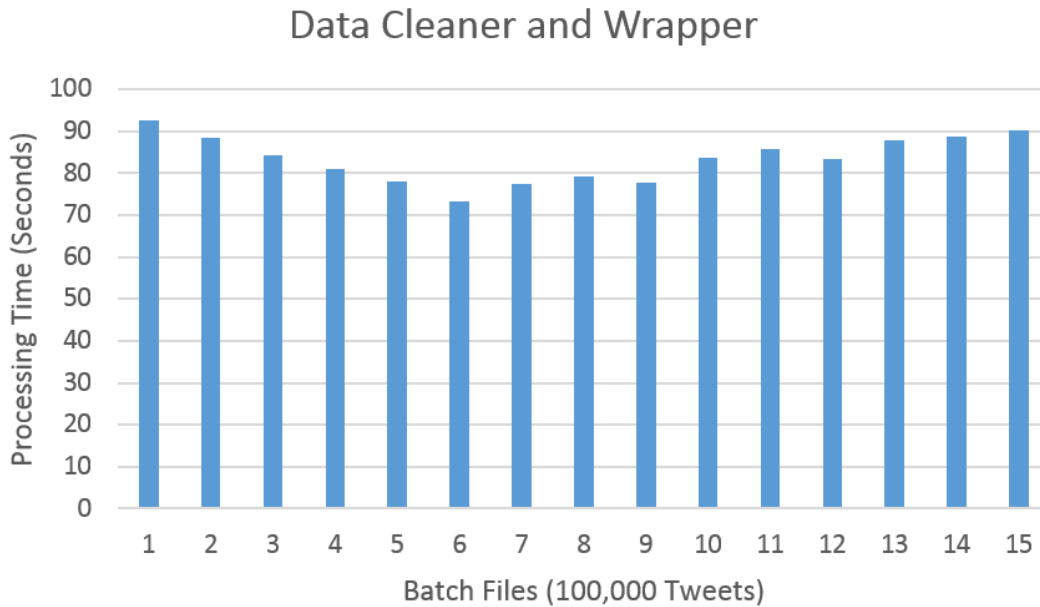


Figure 6-12: Processing time for generating data packets from 100,000 tweets

Indexing: Sequential and Bulk are two modes of indexing in elastic search. We indexed 15,000 data packets in both the approaches on a single node with 1 shard. Sequential took 28 minute and 52 seconds where bulk indexing with error check took 8.24 seconds and without error check took 2.05 seconds as shown in Table 6-2. Error code checks acknowledgement of each bulk indexing data and in case of any error it will resend the bulk data again.

Figure 6-11 shows list of indexed packets in different cells of New York at precision 5.

Method	Time in Seconds
Sequential	1732
Bulk (with error check)	8.24

Table 6-2: Indexing time for 15,000 packets

EoIs: Local EoIs detection are done at leaf level, potential event data packets are considered as a nodes and the value of 'text similarity (TF-IDF)' between data packets as a weight of the bidirectional edge. Data packets with a high text similarity value are clustered. The results of different type of EoIs are shown in Table 6-1 which are extracted from one window based bulk of tweets of 19th June 2016 between 11:48am - 12:14pm. The greedy optimized cluster method runs in time $\sum_{i=1}^C (n_i \log n_i)$ where 'n_i' is the total number of old data packets, new data packets and existing clusters in ith leaf cell and 'C' is the number of updated leaf cell. First time execution of leaf cells does not contain any old data packets and existing clusters and value of the 'C' is total number of leaf cells in spatio-temporal indexed hierarchical tree.

Spatial Scope: Horizontal and vertical spatial scope are done in bottom-up approach, starting from parent of leaf nodes to root of the tree. The horizontal spatial scope start from leaf node and merge same EOIs in the sibling cells. The process continues from leaf node to the root of the tree. Based on clustering of EOIs at different level of tree, system updates the significance of zoom level of a particular EoI. The vertical spatial scope also works in a similar manner but it is not continues as compared to horizontal spatial scope. Vertical scope updated based on importance of event from zoom level k (e.g., corresponds to neighbourhood level on map) to zoom level $k - n$ (e.g., corresponds to city level) where value of n decided based on total number of tweets related to event and number of unique users . The horizontal spatial scope runs in time

$$\sum_{l=leaf-1}^R \sum_{i=1}^M (h_i \log h_i)$$

where h_i is the number of EOIs in child cells of i^{th} cell, ' l ' is depth, ' R ' is root of the tree, and ' M ' is the number of cell at depth ' l '.

The vertical spatial scope runs in time

$$\sum_{l=leaf-1}^R \sum_{i=1}^M (v_i \log v_i)$$

where v_i is the number of EOIs in child cells of i^{th} cell that are not updated in horizontal scope and value of ' $v_i \leq h_i$ ', ' l ' is depth, ' R ' is root of the tree, and ' M ' is the number of cell at depth ' l '. Both horizontal and vertical scopes are calculated from parent of leaf cell at depth ' $leaf-1$ ' upto to the root ' R ' of the tree.

Querying EoIs: For visualization of EOIs on the top of Maps, we need to fetch data based on maps zoom level. At low zoom level, the number of EOIs are less due to few big events at country level. As we zoomed in, the number of EOIs increases. Figure 6-7 shows a query to fetch EOIs at zoom level 17. The query returns 135681 EOIs cluster at zoom 17 whereas Figure 6-13 shows average snapshot of processing time of five different runs in milliseconds for querying EOIs from zoom level 6 to zoom level 17. Zoom levels 18 shows very limited area in maps so we didn't take into consideration for visualizing EOIs rather we display potential indexed packets only.

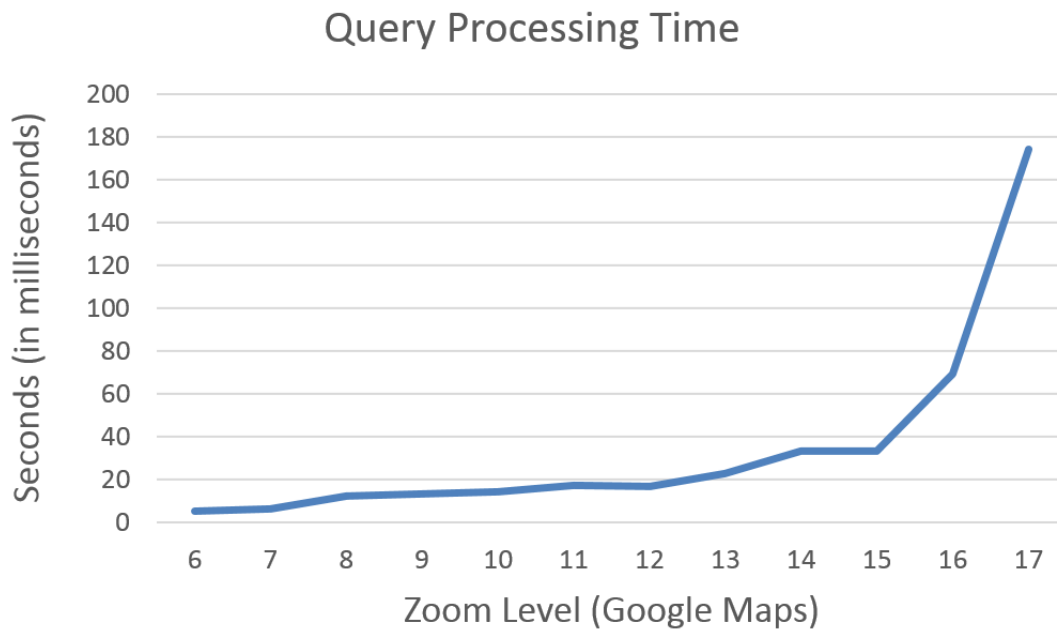


Figure 6-13: Processing time for event discovery at the different map zoom levels

6.5 Conclusion

Hadath is a proof-of-concept implementation to demonstrate the feasibility and adaptability of building *Event-enriched Maps* by displaying real-time events and findings. This chapter showed details about implementation of *Hadath* with their results. Furthermore, *Hadath* provides new dimension to existing maps by displaying EoIs at different scales. The spatio-temporal scope of EoIs so created leads to a unique and dynamic map browsing experience: such a mapping system has the potential to become a city search engine or other spatial queries, where EoIs can be identified at different scales of relevance for city dwellers as well as visitors/tourists.

Chapter 7

Conclusion and Future Research

7.1 Conclusion

In today's busy world, users and authorities require better services to achieve their daily activities and tasks in a smart way by using available resources in an optimum manner. The variety of data sources available today—crowd-sourced data, open governmental data, and other online sources—can provide users smart tools to better manage their daily activities. However, collecting and integrating this multitude of overlapping data sources is a challenging task. Specifically, digital maps are being used extensively to browse and share information about points of interest, for planning trips, and for finding optimum paths. Suggesting an optimum path for large crowds poses a unique challenge to existing routing algorithms due to dynamic changes in road networks. In Chapter 3, we presented a constraint-aware framework that collects data from large crowds by offering location-based services, incentive-based traffic update modules, and geo-tagged social networks. The proposed framework extracts knowledge from the data of social networks and mobile applications in terms of constraints, traffic flow, and average crossing time of an intersection. Then it recommends the best optimum path over such dynamic road networks. We also developed a grid-based algorithm to recommend optimum paths and to minimize the response time to identify affected users.

In addition to social data, other type of data available for extracting information includes open government data, reviews from Booking.com and Yelp, and weather updates through sensors: incorporating these on maps has become an exciting challenge. There is a real opportunity to enrich traditional maps with different knowledge-based layers extracted from this variety of available data sources. Thus in Chapter 4, we proposed the new concept of enriched maps named it as “smart maps”: these are premised on collecting, managing, and integrating heterogeneous data sources in order to infer relevant knowledge-based layers. Unlike conventional maps, smart maps extract live events (e.g., concerts, competitions, and incidents),

online offers, and statistical analyses (e.g., dangerous areas) by encapsulating incoming semi and unstructured data into structured generic packets. These packets are processed to extract statistical knowledge on accident-prone and safe areas, and to detect Events of Interest (EoI) based on a multi-dimensional clustering technique. This approach sets the ground for delivering different intelligent services and applications, such as: 1) *city explorer* that provides the latest information collected from multiple sources about places and events; 2) *route and trip planning* that leverage a smart map framework to recommend safe routes; and 3) *multimedia routing* that enhances routing frameworks by leveraging geo-tagged multimedia data—such as images, audio, video, and text—in order to add semantics to conventional spatial queries. The framework collects, stores, and spatially tags multimedia data shared by users through social networks or by the mobile application developed by us. The system then uses this data to enhance traditional routing services by resolving existing usability issues and by providing semantics to the routes in terms of enriched points of interest.

However, augmenting the concept of smart maps by designing an efficient and scalable system of the world in near real-time remains a challenge: such a system should be able to extract live events and infer their spatial and temporal scopes so that they can be displayed in a clear, non-cluttered manner. This challenge is of critical importance, since events are naturally happening at different scales and with different significance levels. In mapping applications, the context of what to display, when, and on which level(s) of granularity is set based on the spatio-temporal extents of the corresponding items. In the remaining chapters (5 and 6) of the thesis, we introduced a system: called Hadath, it builds multi-resolution, event-enriched maps by handling social data streams. It also develops different algorithms for the efficient extraction, clustering, and mapping of live events. Hadath wraps incoming unstructured data streams into data packets, i.e. a generic structured format of a potential event. These packets are then processed to extract EoIs based on a hierarchical clustering technique, which defines the spatio-temporal scope for each event. The system can provide valuable knowledge from crowd-sourced data to authorities, market firms, event organisers, and end-users to help in decision making. We consider Hadath as a next generation map platform which will intelligently extract relevant knowledge from crowd-sourced data in real-time.

7.2 Future Research

Many perspectives may be taken into consideration in future research in this field.

Primarily, we are planning to include more sensors—such as physical sensors including inductive loop, camera, and drone—to acquire dynamic road conditions. The trajectory flow of data with the time stamp and date of location-based services can be used for statistical analysis of dynamic road conditions in the future. We plan to work on the quality of the generated path before recommending it to users: this will be through mining and validating social sensor data on one hand and improving the system through the learning process on the other. We will also implement a calibration model to find the best-suited frequency for the acquisition of streaming data.

Furthermore, we plan to merge more data sources in our Hadath system (e.g., Flickr, online newspapers) to increase the correctness and conciseness of detected events. We can use machine learning and deep learning techniques to detect EoIs. We also need to handle unspecified events more intelligently. We can use different sizes and shapes of cells based on

city, state, and country maps instead of using fixed size sibling cells at multilevel indexing. Additionally, an extensive performance evaluation of the different solutions needs to be conducted with respect to closely-related systems.

Moreover, we plan to develop a query language that explores the knowledge of detected EoIs in Hadath to enhance existing spatio-temporal queries. It will allow users to post complex queries that need different EoIs—such as dynamic road conditions, musical events, and discounts on pizza—with a safe, multimedia path which can otherwise not be answered in the existing map system.

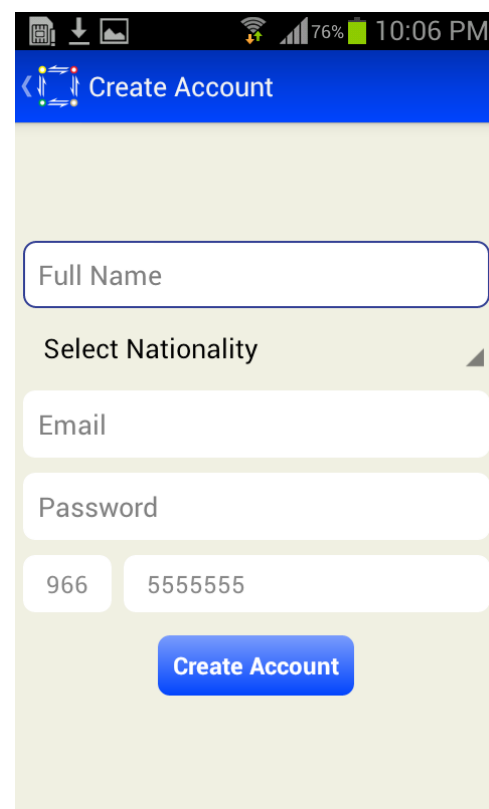
Appendix A

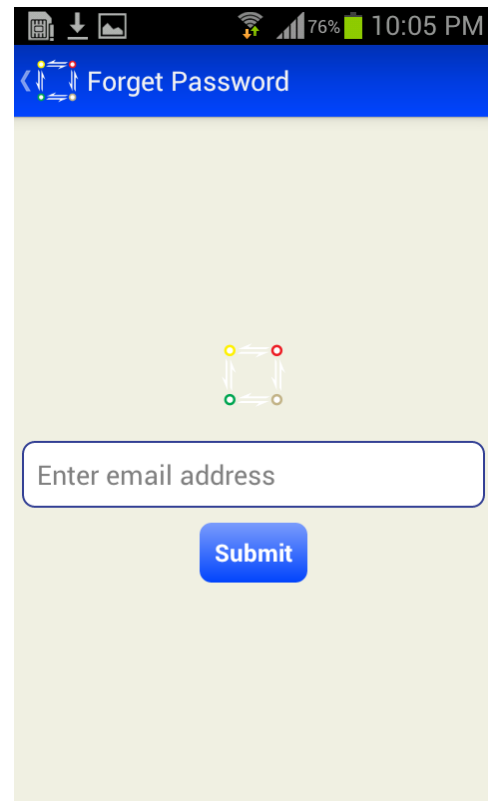
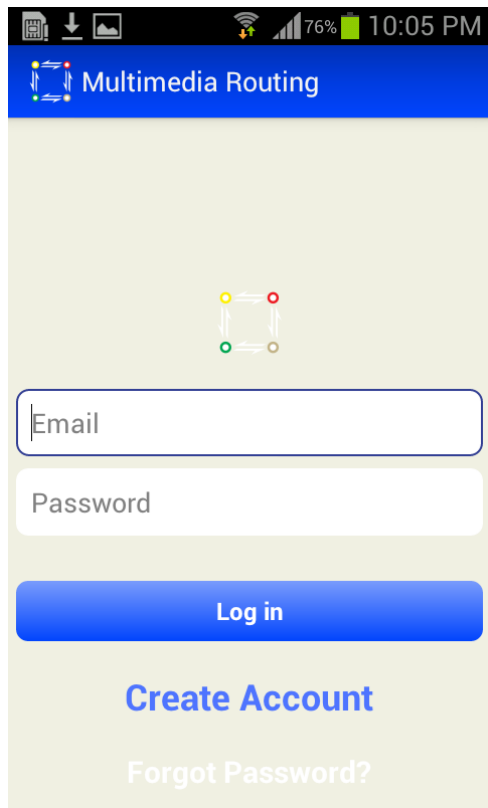
Multimedia Routing

A. Salient Features

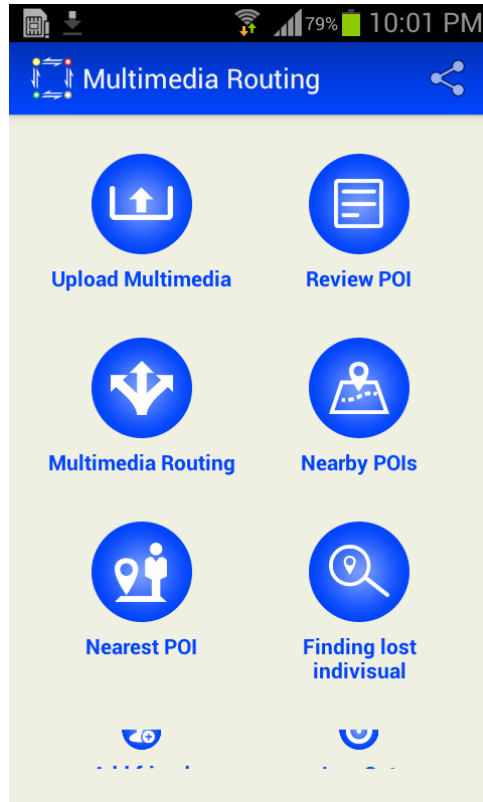
- A framework to environment to answer multimedia spatiotemporal queries in real-time.
- Multimedia routing to enhance traditional routing and increase usability.
- Nearest POIs with multimedia information.
- Finding lost individual in a large crowd using multimedia.
- Add multimedia POIs.
- Update multimedia POIs.
- Add Friends, accept or reject friend request.
- Create account, forget password, login and logout.
- Traffic update with multimedia information.
- Filter to select type of POIs.
- Filter to select type of multimedia.
- Customized results as per the users' smartphone bandwidth and resolution requirements.

B. Icon and Login

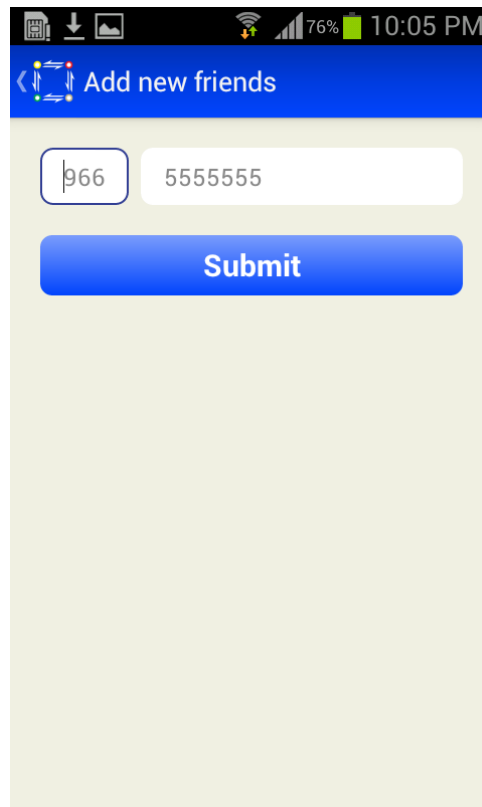




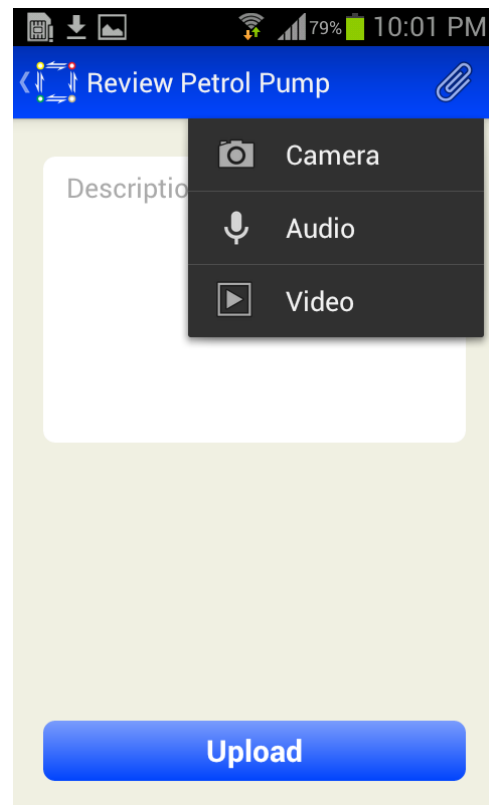
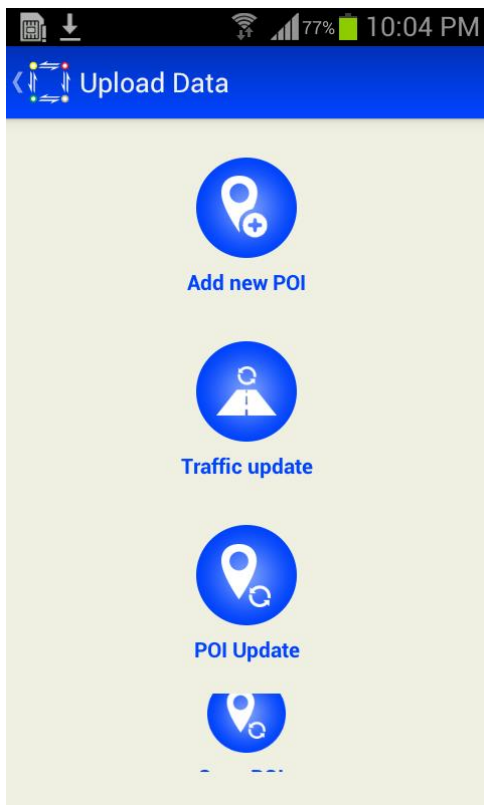
C. Main Screen



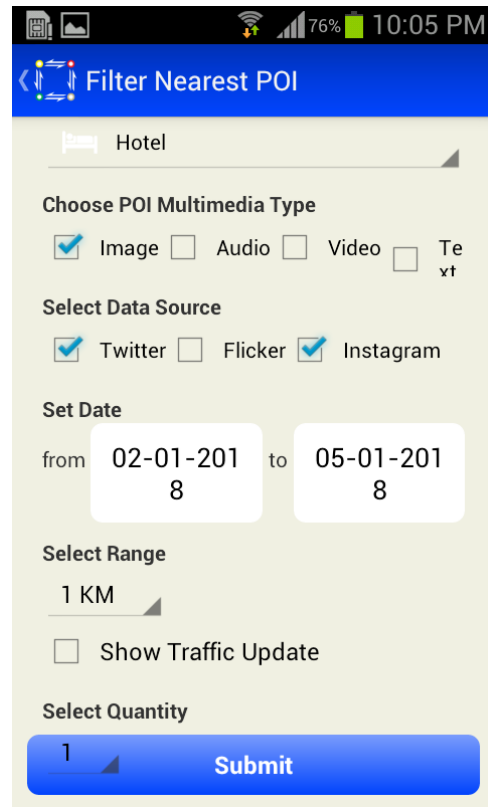
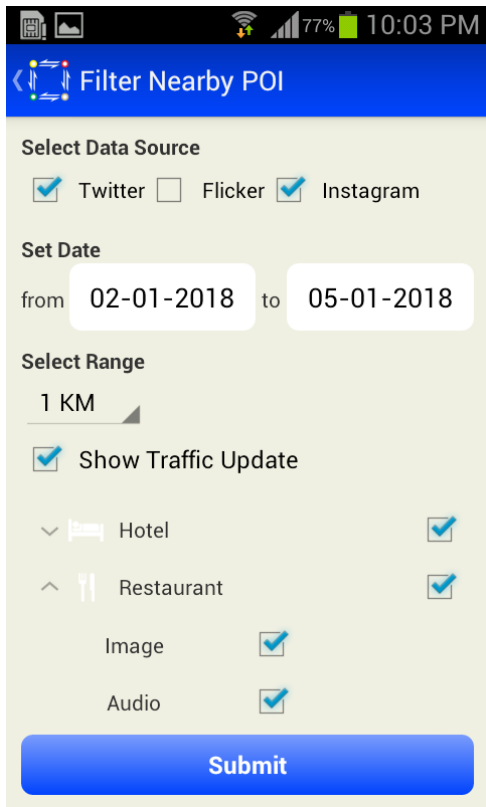
D. Add Friend



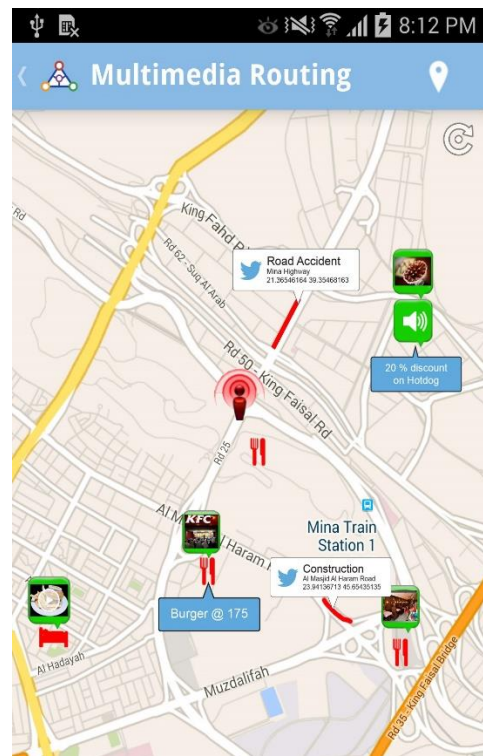
E. Add and Review POIs



F. Filter Conditions



G. Multimedia Routing



Appendix B

Hadath

A. *Salient Features*

- New places of interest
- Traffic Updates
- Statistical knowledge such as crime zone, accident area.
- Dealing with heterogeneous data sources
- Specified and unspecified event detection
- Efficient Indexing
- Clustering of an events based on text-similarity, spatial and temporal parameters
- Understanding spatio-temporal scope of an Eols
- Visualize Eols on different zoom levels based on scope
- Scalable system
- Framework to enhance traditional spatial queries
- Smart applications
 - City explorer
 - Safe Routing

B. *Code snapshot*

a. *Horizontal Scope*

```

private static String geohash1 = "01234567";
private static String geohash2 = "89bcdefg";
private static String geohash3 = "hjkmpqrs";
private static String geohash4 = "stuvwxyz";

public static void main(String[] args) {
    int totalIndexCluster = 0;
    int totalFailedCluster = 0;
    try {
        Settings settings = Settings.settingsBuilder().put("cluster.name", "elasticsearch").build();

        Client client = TransportClient.builder().settings(settings).build().addTransportAddress(new InetSocketAddress(InetAddress.getByAddress("localhost"), 9300));

        // Sample Query
        String queryString = "{\"query\": {\"term\": {\"zoomStart\": \"17\"}}, \"aggregations\": {\"myLarge-GrainGeoHashGrid\": {\"geohash_grid\": \""
            + \"(\field\": \"location\", \"precision\": 5, \"size\": 3000)}}\"";

        // Elasticsearch Response
        SearchResponse response = client.prepareSearch("eventtwitter").setTypes("tweet").setSource(queryString).execute().actionGet();

        // Elasticsearch Response Hits
        GeoHashGrid agg = response.getAggregations().get("myLarge-GrainGeoHashGrid");
        long totalCount = 0;
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss.SSS");
        System.out.println("Start Time:" + sdf.format(cal.getTime()));
        // For each entry
        for (GeoHashGrid.Bucket entry : agg.getBuckets()) {
            String keyAsString = entry.getKeyAsString(); // key as String
            GeoPoint key = (GeoPoint) entry.getKey(); // key as geo point
            long docCount = entry.getDocCount(); // Doc count

            System.out.println("Key:" + keyAsString + " docs:" + docCount);
            String packetList = " ";
            String bucketQueryString = "{\"query\": {\"bool\": { \"must\": [{\"term\": {\"location1.geohash\": \"\" + keyAsString
                + \"\"}}, {\"term\": {\"zoomStart\": \"17\"}}]}}, \"size\": \" + docCount + \"\"";
            SearchResponse bucketResponse = client.prepareSearch("eventtwitter").setTypes("tweet").setSource(bucketQueryString).execute().actionGet();
            BulkRequestBuilder bulkRequestBuilder = client.prepareBulk().setRefresh(true);

            for (SearchHit hit : bucketResponse.getHits()) {
                //Initialize here
                Map<String, Object> sourceData = hit.getSource();
                //to update Flag
                String elasticIndex = hit.getIndex();
                String elasticId = hit.getId();
                id = sourceData.get("id").toString();
                if (packetList.contains(id))
                    continue;
                packetList += " " + id;
                if (title.isEmpty() && sourceData.get("title") != null)
                    title = sourceData.get("title").toString();

                url = sourceData.get("url").toString();

                location = sourceData.get("location");
                timestamp = sourceData.get("@timestamp");
                HashMap<String, Double> sourceLocation = (HashMap<String, Double>) sourceData.get("location");
                lat1 = (double) sourceLocation.get("lat");
                lon1 = (double) sourceLocation.get("lon");
                String cellkeyParent = sourceData.get("cellkey").toString();
                String clusterCellkeyParentCell = cellkeyParent
                    .substring(cellkeyParent.length() - 1);

                int len = sourceData.get("packets").toString().split(",").length;
                latlon.add(new CoordinatesIndex(id, lat1, lon1, len,
                    cellkeyParent, elasticId, elasticIndex));

                if (sourceData.get("eventType") != null) {
                    HashMap<String, Integer> events = (HashMap<String, Integer>) sourceData.get("eventType");
                    // Faizan:Commented
                    for (String event : events.keySet()) {
                        if (eventType.containsKey(event)) {
                            Integer val = eventType.get(event);
                            eventType.put(event, val + 1);
                        } else
                            eventType.put(event, events.get(event));
                    }

                    for (Entry<String, Integer> entry1 : events.entrySet()) {
                        String key1 = entry1.getKey();
                        tagsData += key1 + " ";
                    }
                }
            }
        }
    }
}

```

```

data.put("id", id);
data.put("location", location);
data.put("cellkey", keyAsString);
data.put("@timestamp", timestamp);
packets.put(id);

String bucketTargetQueryString = "{\"query\": {\"bool\": { \"must\": [{\"term\": {\"location1.geohash\": \"\"+ keyAsString + \"\"}}, \"
+ \"{\"term\": {\"zoomStart\": \"17\"}}]}}, \"size\": \"+ docCount + \"\"};
SearchResponse bucketTargetResponse = client
.prepareSearch("eventtwitter").setTypes("tweet")
.setSource(bucketTargetQueryString).execute()
.actionGet();
for (SearchHit hitt : bucketTargetResponse.getHits()) {
Map<String, Object> targetData = hitt.getSource();
String cellkey = targetData.get("cellkey").toString();
String clusterCell = cellkey
.substring(cellkey.length() - 1);
// if cellkey not contain in list
// ignoring check in same cell
if (cellkey.equalsIgnoreCase(cellkeyParent) // &&
// !bigCells.contains(cellkey))

continue;

String idsub = targetData.get("id").toString();
if (id.equals(idsub))
continue;
if (packetList.contains(idsub))
continue;
Boolean flag=false;

if ((geohash1.contains(clusterCell)) && (geohash1.contains(clusterCellkeyParentCell))) {
flag=true;
}
else if ((geohash2.contains(clusterCell)) && (geohash2.contains(clusterCellkeyParentCell))) {
flag=true;
}
else if ((geohash3.contains(clusterCell)) && (geohash3.contains(clusterCellkeyParentCell))) {
flag=true;
}
else if ((geohash4.contains(clusterCell)) && (geohash4.contains(clusterCellkeyParentCell))) {
flag=true;
}
}
if (!flag)
continue;
//to update Flag
String elasticIndex1 = hitt.getIndex();
String elasticId1 = hitt.getId();
String pack = targetData.get("packets").toString();
pack = pack.replaceAll("\\n", "");
String targetPackets = pack.trim().substring(1,
pack.length() - 1);

HashMap<String, Double> destinationLocation = (HashMap<String, Double>) targetData.get("location");
lat2 = (double) destinationLocation.get("lat");
lon2 = (double) destinationLocation.get("lon");

if (title.isEmpty() && targetData.get("title") != null)
title = targetData.get("title").toString();

tagsSubContent = targetData.get("content").toString();
String tagsSubData = "";
if (targetData.get("eventType") != null) {
HashMap<String, Integer> events = (HashMap<String, Integer>) targetData.get("eventType");

for (Entry<String, Integer> entry1 : events
.entrySet()) {
String key1 = entry1.getKey();
Integer value = entry1.getValue();
tagsSubData += key1 + " ";
}
}

Cosine_Similarity csl = new Cosine_Similarity();
if (tagsData.split(" ").length >= DATASPLIT && tagsSubData.split(" ").length >= DATASPLIT) {
double sim_score = csl.Cosine_Similarity_Score(tagsData, tagsSubData);
if (sim_score > SIM_THRESHOLD) {
packetList += " " + idsub;
int sublen = targetData.get("packets")
.toString().split(",").length;
latlon.add(new CoordinatesIndex(idsub, lat2, lon2,
sublen, cellkey, elasticId1, elasticIndex1));
packetCount++;
if (range < 32)
range++;
}
}

```



```

try {
    Settings settings = Settings.settingsBuilder()
        .put("cluster.name", "elasticsearch").build();

    Client client = TransportClient.builder().settings(settings).build().addTransportAddress(new InetSocketAddress(InetAddress
        .getByName("localhost"), 9300));

    // Elastic Query
    String queryString = "{\"query\": { \"bool\": { \"must\": [{ \"terms\": { \"zoomStart\": [ \"15\", \"16\", \"17\" ] } } ] }, \"aggregations\": \" \"
        + \"myLarge-GrainGeoHashGrid\": { \"geohash_grid\": { \"field\": \"location\", \"precision\": 5, \"size\": 10000 } } } }";

    // Elasticsearch Response
    SearchResponse response = client.prepareSearch("eventtwitter*").setTypes("tweet").setSource(queryString).execute().actionGet();

    // Elasticsearch Response Hits
    GeoHashGrid agg = response.getAggregations().get("myLarge-GrainGeoHashGrid");
    long totalCount = 0;
    Calendar cal = Calendar.getInstance();
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss.SSS");
    System.out.println("Start Time:" + sdf.format(cal.getTime()));
    // For each entry
    for (GeoHashGrid.Bucket entry : agg.getBuckets()) {
        String keyAsString = entry.getKeyAsString(); // key as String
        GeoPoint key = (GeoPoint) entry.getKey(); // key as geo point
        long docCount = entry.getDocCount(); // Doc count

        System.out.println("Key:" + keyAsString + " docs:" + docCount);
        String packetList = " ";
        String bucketQueryString = "{\"query\": { \"bool\": { \"must\": [ { \"terms\": { \"zoomStart\": [ \"15\", \"16\", \"17\" ] }, { \"range\": { \"packetscount\": \" \"
            + \">19\" } } ] }, { \"term\": { \"visited\": false } }, { \"term\": { \"location1.geohash\": \"\" + keyAsString + \"\" } } ] }, \"size\": \" + docCount + \" } }";

        SearchResponse bucketResponse = client.prepareSearch("eventtwitter*").setTypes("tweet").setSource(bucketQueryString).execute().actionGet();
        BulkRequestBuilder bulkRequestBuilder = client.prepareBulk().setRefresh(true);

        for (SearchHit hit : bucketResponse.getHits()) {

            //initialize variables
            String id;
            int zoomUpdatedValue = 0;
            int packetCount = 0;
            ArrayList<CoordinatesIndex> latlon = new ArrayList<CoordinatesIndex>();
            Map<String, Object> data = new HashMap<String, Object>();

            Map<String, Object> sourceData = hit.getSource();
            //to update flag
            String elasticIndex = hit.getIndex();
            String elasticId = hit.getId();
            id = sourceData.get("id").toString();
            if (packetList.contains(id))
                continue;
            packetList += " " + id;

            packetCount = (int)sourceData.get("packetscount");
            if (packetCount >= getZoom14ThresholdValue()) {
                zoomUpdatedValue = ZOOM14;
                //flag=true;
                data.put("id", id);
                data.put("location", sourceData.get("location"));
                data.put("cellkey", keyAsString);
                data.put("@timestamp", sourceData.get("@timestamp"));
                data.put("title", sourceData.get("title").toString());
                data.put("eventType", sourceData.get("eventType"));
                data.put("zoomStart", ZOOM14);
                data.put("zoomEnd", ZOOM17);
                //data.put("location", location);
                data.put("location1", sourceData.get("location"));
                data.put("visited", false);
                data.put("packetscount", sourceData.get("packetscount"));
                data.put("url", sourceData.get("url"));
                data.put("content", sourceData.get("content"));
                data.put("packets", sourceData.get("packets"));
                data.put("percentage", sourceData.get("percentage"));
                Date date = new Date(
                    Long.parseLong(data.get(
                        "@timestamp")
                            .toString());
                );
                bulkRequestBuilder.add(client.prepareIndex("eventtwitter-"+ outFormatter.format(date), "tweet", id + ZOOM14).setSource(data));
            }

            if (packetCount >= getZoom14ThresholdValue()) {
                zoomUpdatedValue = ZOOM15;
                data.put("id", id);
                data.put("location", sourceData.get("location"));
                data.put("cellkey", keyAsString);
            }

```


- b. Query to fetch all data packets with potential flag value as false (potentially unspecified Eols). The results took '2396' ms to answer the query and return 9,80,318 number of data packets with '28' successful shards.

```
Server localhost:9200
247 GET /twitter/_search
248 {"query": {"term": {"location.geohash": "s14mh"}}, "size":
249 2000}
250 {"query": {"term": {"location.geohash": "s14mh"}}}
251 {"query": {}}
252
253 "wildcard": {"header.eventname": "graduation" }
254
255
256 }
257
258 GET /twitter/_search
259 {"query": {
260 "term": {
261 "header.potentialEvent": "false"
262 }
263 }
264 }
265
266 GET /eventtwitter/_search
267 {"query": {"bool": {"must": [{"term": {"location.geohash":
268 "s14mh"}}, {"term": {"zoomStart": "17"}]}], "size": 8000}
269 {"query": {"bool": {"must": [{"match": {"_id": "806258571482
270 632192"}}, {"match": {"cellkey": "d-r-5"}]}]}]}
271 GET /eventtwitter/_search
272 {"query": {"bool": {"must": [{"match": {"_id": "806258571482
273 632192"}}, {"match": {"cellkey": "d-r-5"}]}]}]}
274 GET /eventtwitter/_search
275 {"query": {"terms": {"zoomStart": ["15", "16", "17"]},
276 "aggregations": {"myLarge-GeoHashField": {"geohash_fid
277 "": {"field": "location", "precision": 5}}}}
278
1 {
2 "took": 2396,
3 "land_out": false,
4 "_shards": {
5 "total": 28,
6 "successful": 28,
7 "failed": 0
8 }},
9 "hits": {
10 "total": 980318,
11 "max_score": 2.18234,
12 "hits": [
13 {
14 "_index": "twitter-2016_12_25",
15 "_type": "tweet",
16 "_id": "APVPMZeholwrezMh17",
17 "score": 2.18234,
18 "_source": {
19 "id": "813083905221889952",
20 "visited": "false",
21 "timestamp": 1482689284089,
22 "location": {
23 "lat": 4.85345709,
24 "lon": 100.73864488
25 },
26 "location": {
27 "lat": 4.85345709,
28 "lon": 100.73864488
29 },
30 "latitude": 4.85345709,
31 "longitude": 100.73864488,
32 "payload": {
33 "tags": [
34 "foodstagram",
35 "food",
```


- f. Query to fetch Eols at Zoom level '17' from a cell "dr5rv". The results took '9' ms time and return '119' records along with their details. It also shows one Eol in a results that includes clustering of '2' similar packets.

```

Server localhost:9200
178 "size":500
179 }
180 }
181
182 GET /event/twiter*/_search
183 {"query":{"bool":{"must":[{"range":{"percentage":{"lte":100}}},{"match":{"zoomStart":{"lte":11}}]}]},"size":9000}
184
185
186 GET /event/twiter*/_search
187 {"query":{"bool":{"must":[{"range":{"percentage":{"gte":0}}},{"match":{"zoomStart":{"lte":15}}]}]},"size":9000}
188
189 {"query":{"bool":{"must":[{"range":{"percentage":{"lte":100}}},{"match":{"zoomStart":{"lte":5}}]}]},"size":9000}
190
191 GET /event/twiter*/_search
192 {"query":{"bool":{"must":[{"range":{"percentage":{"lte":100}}},{"match":{"zoomStart":{"lte":17}}]}]},"size":20}
193
194 {"query":{"bool":{"must":[{"match":{"packets":{"8062585748262192"}}}},{"match":{"callkey":{"dr5rv"}}]}]}
195 GET /event*/_search
196 {
197   "query": {
198     "bool": { "must": [
199       { "term": { "location.geohash": "dr5rv" } }
200       , { "term": { "zoomStart": "17" } }
201     ] }
202   }
203 }
204 }
205 "size": 10
206 }

1 - {
2   "took": 9,
3   "timed_out": false,
4   "shards": {
5     "total": 28,
6     "successful": 28,
7     "failed": 0
8   },
9   "hits": {
10    "total": 119,
11    "max_score": 9.673063,
12    "hits": [
13      {
14        "_index": "event/twiter-2017.01.01",
15        "_type": "tweet",
16        "_id": "81564900271333122",
17        "_score": 9.673063,
18        "source": {
19          "event": {
20            "subway": 2,
21            "theyhaveballoon": 1,
22            "street": 2,
23            "7ndave": 1,
24            "7nd": 2,
25            "2ndavenuesubway": 2,
26            "construction": 1,
27            "avenue": 2,
28            "mta": 1
29          }
30        },
31        "packetscount": 2,
32        "title": "Incident/street/construction",
33        "packets": "[\n      #1ma #2ndavenuesubway #2ndave #construction @ 7nd Street (Second Avenue Subway) https://t.co/9vz0hprnV",
34        "url": "https://t.co/9vz0hprnV",
35        "@latestamp": 1483300850000,

```


D. Visualization on Kibana

Kibana²⁷ is an open source tool to visualize elasticsearch indexed data. It allows users to create pie chart, line, plot and maps on large volume of indexed data.

- a. Visualization of back-end indexed data in list format.*

²⁷ <https://www.elastic.co/products/kibana>

b. Visualization of a sample indexed data (part-1).

```

    id: 811731788804464640 visited: false @timestamp: 1,482,366,914,000 location: { "lat": 40.2516871, "lon": -111.6675215 } location1: { "lat": 40.2516871, "lon": -111.6675215 } latitude: 40.252 longitude: -111.668 payload.tags: Provo, Job, SONIC, Hospitality, Hiring, CareerArc payload.followers: 346 payload.viewers: 0 payload.title: Payload text: If you're looking for work in #Provo, UT, check out this #job: https://t.co/1l0tUeZVs #SONIC #Hospitality #Hiring #CareerArc payload.screenname: tmj_slc_hrt payload.language: en payload.displayName: TMJ-SLC HRTA Jobs payload.url: https://t.co/1l0tUeZVs header.time: Thu Dec 22 00:35:14 +0000 2016 header.timestamp: Thu Dec 22 00:35:14 +0000 2016 header.geo.coordinates: 40.252, -111.668 header.source: twitter header.eventType: looking, work, provo, c heck, job, sonic, hospitality, hiring, careerarc header.types: text header.potentialEvent: true header.eventname: Hiring/Hiring/ id: AVpDm1yeb0W-eZLdW type: tweet index: twitter-2016.12.22 score: 1

Table JSON
# @timestamp @ @ 1,482,366,914,000
# _id @ @ AVpDm1yeb0W-eZLdW
# _index @ @ twitter-2016.12.22
# _score @ @ 1
# _type @ @ tweet
# header.eventType @ @ looking, work, provo, check, job, sonic, hospitality, hiring, careerarc
# header.eventname @ @ Hiring/Hiring/
# header.geo.coordinates @ @ 40.252, -111.668
# header.geo.type @ @ point
# header.potentialEvent @ @ true
# header.source @ @ twitter
# header.time @ @ Thu Dec 22 00:35:14 +0000 2016
# header.timestamp @ @ Thu Dec 22 00:35:14 +0000 2016
# header.type @ @ text
# id @ @ 811731788804464640
# latitude @ @ 40.252
# location @ @ {
  "lat": 40.2516871,
  "lon": -111.6675215
}
# location1 @ @ {
  "lat": 40.2516871,
  "lon": -111.6675215
}
# longitude @ @ -111.668
# payload.displayName @ @ TMJ-SLC HRTA Jobs
# payload.followers @ @ 346
  
```

[Link to Twitter-2016.12.22/Tweet/AVpDm1yeb0W-eZLdW](#)

c. Visualization of a sample indexed data (part-2).

# @timestamp	Q Q Q 1,492,366,914,000
t _id	Q Q Q Alpv0m1Jyeh0Ww-etz6qW
t _index	Q Q Q twitter-2016.12.22
# _score	Q Q Q 1
t _type	Q Q Q tweet
t header.eventType	Q Q Q Looking, work, provo, check, job, sonic, hospitality, hiring, careerarc
t header.eventname	Q Q Q Hiring/hiring/
# header.geo.coordinates	Q Q Q 40.252, -111.668
t header.geo.type	Q Q Q point
header.potentialEvent	Q Q Q true
t header.source	Q Q Q twitter
t header.time	Q Q Q Thu Dec 22 00:35:14 +0000 2016
t header.timestamp	Q Q Q Thu Dec 22 00:35:14 +0000 2016
t header.type	Q Q Q text
t id	Q Q Q 811731788804464640
# latitude	Q Q Q 40.252
location	Q Q Q { "lat": 40.2516871, "lon": -111.6675215 }
location1	Q Q Q { "lat": 40.2516871, "lon": -111.6675215 }
# longitude	Q Q Q -111.668
t payload.displayName	Q Q Q TMJ-SLC HRTA Jobs
# payload.followers	Q Q Q 346
t payload.language	Q Q Q en
t payload.screenName	Q Q Q tmj_slc_hrta
t payload.tags	Q Q Q Provo, job, SONIC, hospitality, Hiring, CareerArc
t payload.text	Q Q Q If you re Looking for work in #Provo, UT, check out this #job: https://t.co/Lt0UAEZv5 #SONIC #Hospitality #Hiring #CareerArc
t payload.title	Q Q Q
t payload.url	Q Q Q https://t.co/Lt0UAEZv5

d. Visualization of kibana status and memory usage.

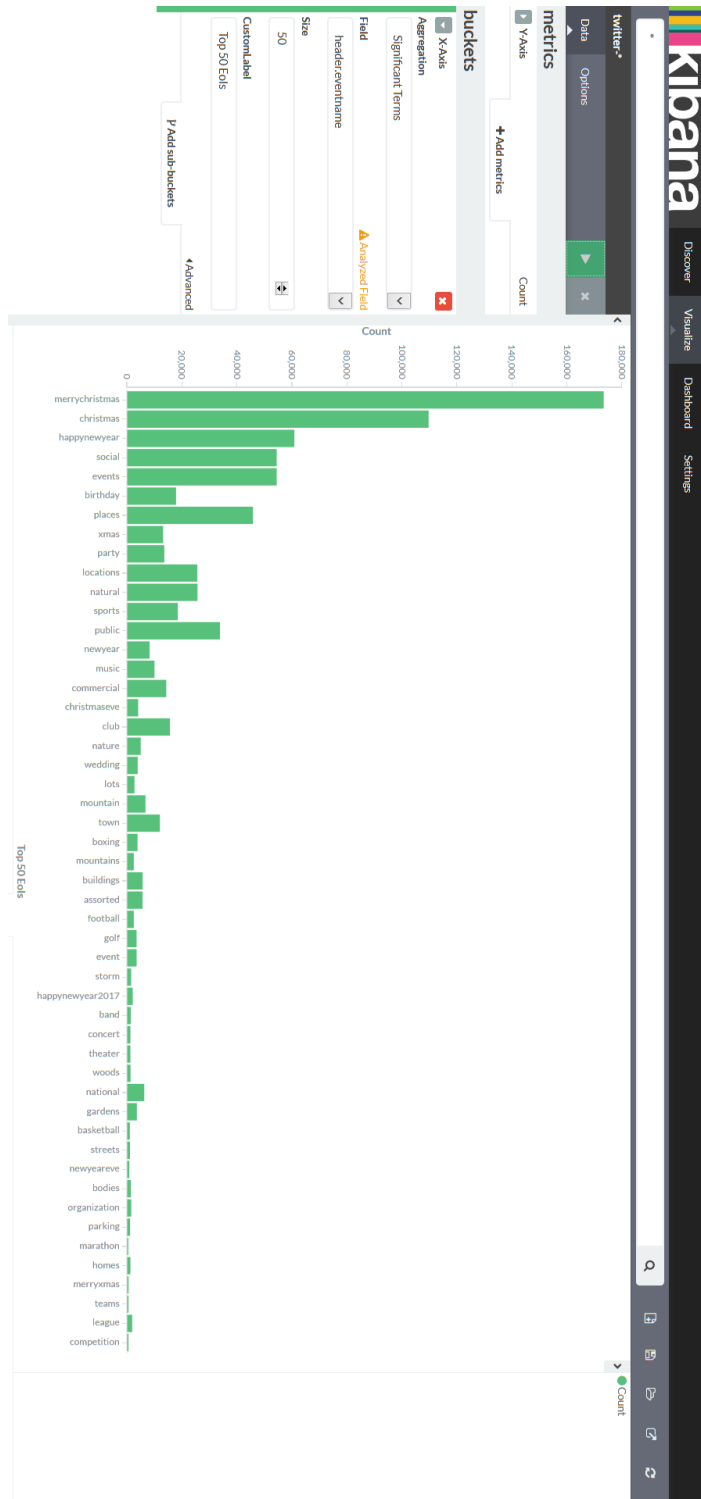
Status: **Green** ✓

Heap Total (MB) 413.88	Heap Used (MB) 405.95	Load 0.00, 0.00, 0.00
Response Time Avg (ms) 1.06	Response Time Max (ms) 1.50	Requests Per Second 0.47

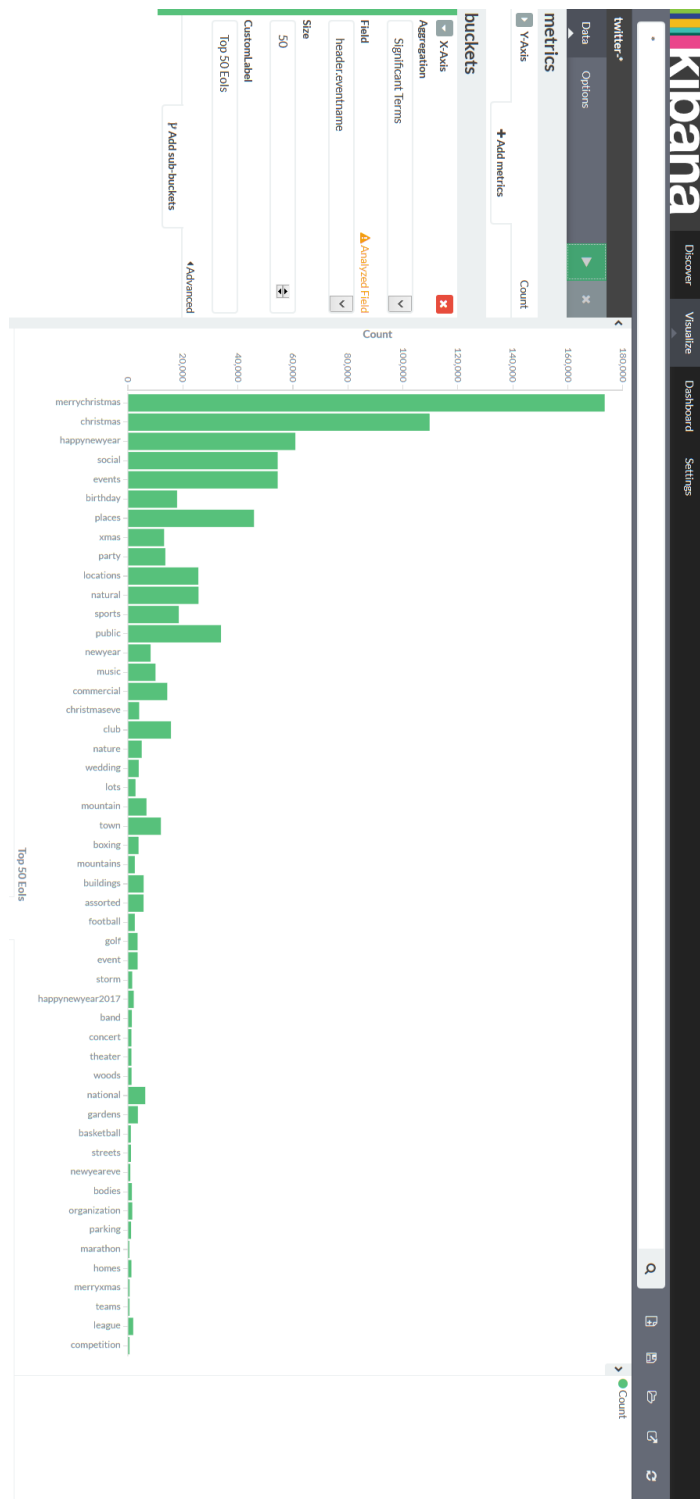
Installed Plugins

Name	Status
plugin:kibana	✓ Ready
plugin:elasticsearch	✓ Kibana index ready
plugin:kbn_vislib_vis_types	✓ Ready
plugin:markdown_vis	✓ Ready
plugin:metric_vis	✓ Ready
plugin:spyModes	✓ Ready
plugin:statusPage	✓ Ready
plugin:table_vis	✓ Ready

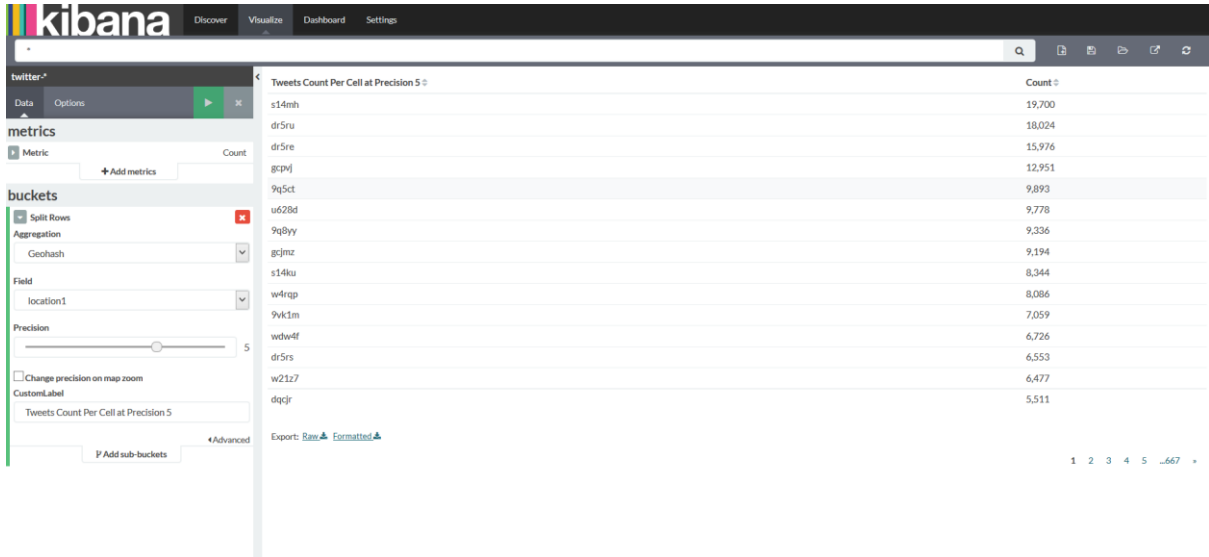
e. Visualization of top 50 Eols.



f. Visualization of top 50 active users.



g. Visualization of list of cells with number of indexed data packets in each cells.



List of Publications

Journal Papers

- 1 **Faizan Ur Rehman**, Imad Afyouni, Ahmed Lbath, Sohaib Khan, Saleh Basalamah, [Building Socially-Enabled Event-Enriched Maps](#), *GeoInformatica*, (Under Review-submitted on September, 2018).
- 2 Akhlaq Ahmad, Md. Abdur Rahman, Mohamed Ridza Wahiddin, **Faizan Ur Rehman**, Abdelmajid Khelil, Ahmed Lbath, “[Context-Aware Services based on Spatio-Temporal Zoning and Crowdsourcing](#)”, *Journal of Behavior and Information Technology, Taylor Francis Series*, Volume 37, Pages 736-760, 2018.
- 3 Amgad Madkour, Walid G. Aref, **Faizan Ur Rehman**, Mohamed Abdur Rahman, Saleh Basalamah, [A Survey of Shortest-Path Algorithms](#). *ACM Transactions on Spatial Algorithms and Systems* (Under Review-submitted on February, 2018), *CoRR abs/1705.02044* (2017)
- 4 KwangSoo Yang, Apurv Hirsh Shekhar, **Faizan Ur Rehman**, Hassan F. Lahza, Saleh M. Basalamah, Shashi Shekhar, Imtiaz Ahmed, Arif Ghafoor, [Intelligent Shelter Allotment for Emergency Evacuation Planning: A Case Study of Makkah](#). *IEEE Intelligent Systems* 30(5): 66-76 (2015)

Conference Papers

- 1 **Faizan Ur Rehman**, Imad Afyouni, Ahmed Lbath, Saleh Basalamah, [Towards Building Smart Maps from Heterogeneous Data Sources](#), *The Ninth International Conference on Advanced Geographic Information Systems, Applications, and Services, GEOprocessing, Nice, France, March 2017*.
- 2 **Faizan Ur Rehman**, Imad Afyouni, Ahmed Lbath, Saleh M. Basalamah, [Understanding the Spatio-Temporal Scope of Multi-scale Social Events](#), *1st ACM SIGSPATIAL Workshop on Analytics for Local Events and News (LENS 2017), Redondo Beach, California, USA, ACM Sigspatial 2017, November, 2017*
- 3 **Faizan Ur Rehman**, Imad Afyouni, Ahmed Lbath, Sohaib Khan, Saleh M. Basalamah, Mohamed F. Mokbel, [Building Multi-Resolution Event-Enriched Maps From Social Data](#). *The 20th International Conference on Extending Database Technology (EDBT'17) 594-597, Venice, Italy, March 2017*

-
- 4 **Faizan Ur Rehman**, Ahmed Lbath, Bilal Sadiq, Akhlaq Ahmad, Abdullah Murad, Imad Afyouni, Md. Abdur Rahman, Saleh Basalamah, "[Constraint-Aware Optimized Path Recommender in Crowdsourced Environment](#)". *12th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2015)*, November 17-20, 2015, Marrakech, Morocco.
 - 5 Bilal Sadiq, **Faizan Ur Rehman**, Akhlaq Ahmad, Abdullah Murad, Ahmad Lbath, Md. Abdur Rahman, Sohaib Ghani, "[A Spatio-temporal Multimedia Big Data Framework for a Large Crowd](#)". *Proceeding of IEEE International Conference on Big Data (IEEE Big Data 2015)*, October 29-November 1, 2015, Santa Clara, CA, USA
 - 6 Akhlaq Ahmad, Imad Afyouni, Abdullah Murad, Md. Abdur Rahman, **Faizan Ur Rehman**, Bilal Sadiq, Saleh M. Basalamah, Mohamed Ridza Wahiddin, [ST-Diary: A Multimedia Authoring Environment for Crowdsourced Spatio-Temporal Events](#). *LBSN@SIGSPATIAL/GIS ACM Sigspatial 2015: 2:1-2:7*, Seattle, Washington, USA, November 2015
 - 7 **Faizan Ur Rehman**, Ahmed Lbath, Imad Afyouni, Abdullah Murad, Md. Abdur Rahman, Bilal Sadiq, Akhlaq Ahmad, Saleh M. Basalamah, [Semantic multimedia-enhanced spatio-temporal queries in a crowdsourced environment](#). *SIGSPATIAL/GIS (ACM Sigspatial) 2015: 100:1-100:4*, Seattle, Washington, USA, November 2015
 - 8 **Faizan Ur Rehman**, Ahmed Lbath, Abdullah Murad, Md. Abdur Rahman, Bilal Sadiq, Akhlaq Ahmad, Ahmad M. Qamar, Saleh M. Basalamah, [A Semantic Geo-Tagged Multimedia-Based Routing in a Crowdsourced Big Data Environment](#). *Proceeding of ACM Multimedia (demo paper), ACM MM'15 October 2015, Brisbane, Australia*
 - 9 Akhlaq Ahmad, **Faizan Ur Rehman**, Md. Abdur Rahman, Abdullah Murad, Bilal Sadiq, Ahmad Qamar, Saleh Basalamah, Mohamed Ridza Wahiddin, "[i-Diary: A Crowdsourced Spatio-Temporal Multimedia Enhanced Points of Interest Authoring Tool](#)", *Proceeding of ACM Multimedia (demo paper), ACM MM'15 October 2015, Brisbane, Australia*.
 - 10 Akhlaq Ahmad, Imad Afyouni, Abdullah Murad, Md. Abdur Rahman, **Faizan Ur Rehman**, Bilal Sadiq, Mohamed Ridza Wahiddin, "[Quality and Context-Aware Data Collection Architecture from Crowd-Sourced Data](#)", in *Proceedings of the Fourth International Multi-topic Conference (IMTIC'15)*, Feb 2015, Pakistan
 - 11 Akhlaq Ahmad, Md. Abdur Rahman, **Faizan Ur Rehman**, Imad Afyouni, Bilal Sadiq, Mohamed Ridza Wahiddin, "[Towards a Mobile and Context-Aware Framework from Crowdsourced Data](#)", *The 5th International Conference on Information and*

Communication Technology for The Muslim World (ICT4M'14), Nov. 2014, Kuching Malaysia).

- 12 **Faizan Ur Rehman**, Ahmed Lbath, Md. Abdur Rahman, Saleh M. Basalamah, Imad Afyouni, Akhlaq Ahmad, Syed Osama Hussain, [Toward dynamic path recommender system based on social network data](#). *IWCTS@SIGSPATIAL 2014: 64-69 (ACM Sigspatial)*, Dallas, Texax, USA, November, 2014.
- 13 Ahmad M. Qamar, Imad Afyouni, Md. Abdur Rahman, **Faizan Ur Rehman**, Delwar Hussain, Saleh M. Basalamah, Ahmed Lbath, [A GIS-based serious game interface for therapy monitoring](#). *SIGSPATIAL/GIS 2014: 589-592 (ACM Sigspatial)*, Dallas, Texax, USA, November, 2014.
- 14 Akhlaq Ahmad, Md. Abdur Rahman, **Faizan Ur Rehman**, Ahmed Lbath, Imad Afyouni, Abdelmajid Khelil, Syed Osama Hussain, Bilal Sadiq, Mohamed Ridza Wahiddin, "[A framework for crowd-sourced data collection and context-aware services in Hajj and Umrah](#)". *12th ACS/IEEE International Conference on Computer Systems and Applications AICCSA'14*, Nov. 2014, Doha Qatar.

Bibliography

- [1] M. F. Goodchild, "Citizens as sensors: the world of volunteered geography," *GeoJournal*, vol. 69, no. 4, pp. 211–221, Aug. 2007.
- [2] C. Vega-Garcia, "Making Maps: A Visual Guide To Map Design For Gis (Second Edition)," *Photogramm. Rec.*, vol. 27, no. 139, pp. 389–390, 2012.
- [3] "Foursquare Map." [Online]. Available: <https://foursquare.com/>.
- [4] "Flickr." [Online]. Available: <https://www.flickr.com/map>.
- [5] A. Magdy *et al.*, "Taghreed : A System for Querying , Analyzing , and Visualizing Geotagged Microblogs," *ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, no. 22, pp. 163–172, 2014.
- [6] "MapD Tweet Demo." [Online]. Available: <https://www.mapd.com/>.
- [7] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling, "NewsStand: A new view on news," in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, 2008, p. 18.
- [8] J. Sankaranarayanan, B. E. Teitler, H. Samet, M. D. Lieberman, and J. Sperling, "TwitterStand : News in Tweets * ," 2009.
- [9] "World Health Organization." [Online]. Available: <http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html>.
- [10] S. Consoli *et al.*, "A Smart City Data Model Based on Semantics Best Practice and Principles," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1395–1400.
- [11] "IBM (Smarter Planet-Smarter Cities)." [Online]. Available: https://www.ibm.com/smarterplanet/us/en/smarter_cities/overview/.
- [12] "Microsoft (CityNext)." [Online]. Available: <https://www.microsoft.com/en-us/enterprise/citynext>.
- [13] X. L. Dong and F. Naumann, "Data Fusion: Resolving Data Conflicts for Integration," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1654–1655, Aug. 2009.
- [14] A. Halevy, A. Rajaraman, and J. Ordille, "Data Integration: The Teenage Years," in *Proceedings of the 32Nd International Conference on Very Large Data Bases*, 2006, pp. 9–16.
- [15] A. H. LE Fleming, "Data mashups: potential contribution to decision support on climate change and health," *Int. J. Environ. Res. Public Heal.*, pp. 1725–1746, 2014.
- [16] G. Lamprianidis, D. Skoutas, G. Papatheodorou, and D. Pfoser, "Extraction, Integration and Analysis of Crowdsourced Points of Interest from Multiple Web Sources," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, 2014, pp. 16–23.
- [17] G. Lamprianidis, D. Skoutas, G. Papatheodorou, and D. Pfoser, "Extraction, Integration and Exploration of Crowdsourced Geospatial Content from Multiple Web Sources," in *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in*

- Geographic Information Systems*, 2014, pp. 553–556.
- [18] A. Ghasemi, *Sustainability and Data Collection in the Smart City - A Case Study of the Wi-Fi and Bluetooth Tracking Project in Copenhagen*. 2015.
- [19] J. M. Schleicher, M. Vögler, C. Inzinger, and S. Dustdar, “Towards the Internet of Cities: A Research Roadmap for Next-Generation Smart Cities,” in *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics*, 2015, pp. 3–6.
- [20] S. Bischof, A. Karapantelakis, C.-S. Nechifor, A. P. Sheth, A. Mileo, and P. Barnaghi, “Semantic modelling of smart city data,” 2014.
- [21] X. L. Dong, L. Berti-Equille, and D. Srivastava, “Integrating Conflicting Data: The Role of Source Dependence,” *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 550–561, Aug. 2009.
- [22] X. L. Dong, L. Berti-Equille, and D. Srivastava, “Truth Discovery and Copying Detection in a Dynamic World,” *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 562–573, Aug. 2009.
- [23] X. Yin, J. Han, and P. S. Yu, “Truth Discovery with Multiple Conflicting Information Providers on the Web,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 796–808, Jun. 2008.
- [24] F. Atefeh and W. Khreich, “a Survey of Techniques for Event Detection in Twitter,” *Comput. Intell.*, vol. 31, no. 1, p. n/a-n/a, 2013.
- [25] F. Zarrinkalam and E. Bagheri, “Event identification in social networks,” *Encycl. with Semant. Comput. Robot. Intell.*, vol. 01, no. 01, p. 1630002, 2017.
- [26] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors,” *Proc. 19th Int. Conf. World Wide Web*, pp. 851–860, 2010.
- [27] S. Panem, M. Gupta, and V. Varma, “Structured Information Extraction from Natural Disaster Events on Twitter,” in *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, 2014, pp. 1–8.
- [28] H. Li, H. Ji, and L. Zhao, “Social event extraction: Task, challenges and techniques,” in *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, 2015, pp. 526–532.
- [29] E. Benson, A. Haghighi, and R. Barzilay, “Event discovery in social media feeds,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 389–398.
- [30] F. U. Rehman *et al.*, “A Constraint-Aware Optimized Path Recommender in a Crowdsourced Environment.”, AICCSA, Morocco, 2015.
- [31] F. U. Rehman *et al.*, “Toward dynamic path recommender system based on social network data,” *Proc. 7th ACM SIGSPATIAL Int. Work. Comput. Transp. Sci. - IWCTS '14*, pp. 64–69, 2014.
- [32] X. Zhang, X. Chen, Y. Chen, S. Wang, Z. Li, and J. Xia, “Event detection and popularity prediction in microblogging,” *Neurocomputing*, vol. 149, no. PC, pp. 1469–1480, 2015.
- [33] A. Magnuson, V. Dialani, and D. Mallela, “Event Recommendation using Twitter Activity,” *RecSys '15 Proc. 9th ACM Conf. Recomm. Syst.*, pp. 331–332, 2015.
- [34] H. et al., “Reading News with Maps by Exploiting Spatial Synonyms,” *Commun. ACM*, vol. 57, no. 10, pp. 64–77, Sep. 2014.
- [35] S. B. Kaleel and A. Abhari, “Cluster-discovery of twitter messages for event detection and trending,” *J. Comput. Sci.*, vol. 6, pp. 47–57, 2015.

-
- [36] N. Nitta, Y. Kumihashi, T. Kato, and N. Babaguchi, “Real-world event detection using flickr images,” in *International Conference on Multimedia Modeling*, 2014, pp. 307–314.
- [37] L. Chen and A. Roy, “Event detection from flickr data through wavelet-based spatial analysis,” in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 523–532.
- [38] S. Petrović, M. Osborne, and V. Lavrenko, “Streaming first story detection with application to twitter,” in *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, 2010, pp. 181–189.
- [39] D. T. Nguyen and J. E. Jung, “Real-time event detection for online behavioral analysis of big social data,” *Futur. Gener. Comput. Syst.*, vol. 66, pp. 137–145, 2017.
- [40] H. Becker, M. Naaman, and L. Gravano, “Beyond Trending Topics: Real-World Event Identification on Twitter,” in *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.
- [41] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu, “Parameter Free Bursty Events Detection in Text Streams,” in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 181–192.
- [42] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang, “TEDAS: A Twitter-based Event Detection and Analysis System,” in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, 2012, pp. 1273–1276.
- [43] M. Schinas, S. Papadopoulos, G. Petkos, Y. Kompatsiaris, and P. A. Mitkas, “Multimodal Graph-based Event Detection and Summarization in Social Media Streams,” in *Proceedings of the 23rd ACM International Conference on Multimedia*, 2015, pp. 189–192.
- [44] G. Petkos, S. Papadopoulos, and Y. Kompatsiaris, “Social Event Detection Using Multimodal Clustering and Integrating Supervisory Signals,” in *Proceedings of the 2Nd ACM International Conference on Multimedia Retrieval*, 2012, pp. 231–238.
- [45] J. Wang, M. Korayem, S. Blanco, and D. J. Crandall, “Tracking natural events through social media and computer vision,” in *Proceedings of the 2016 ACM on Multimedia Conference*, 2016, pp. 1097–1101.
- [46] X. Dong, D. Mavroeidis, F. Calabrese, and P. Frossard, “Multiscale Event Detection in Social Media,” *Data Min. Knowl. Discov.*, vol. 29, no. 5, pp. 1374–1405, Sep. 2015.
- [47] L. Zhao, F. Chen, C.-T. Lu, and N. Ramakrishnan, “Multi-resolution spatial event forecasting in social media,” in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 2016, pp. 689–698.
- [48] Y. Jiang, C. Perng, and T. Li, “META : Multi-resolution Framework for Event Summarization,” pp. 605–613.
- [49] B. M. Varghese, A. Unnikrishnan, and others, “SPATIAL CLUSTERING ALGORITHMS-AN OVERVIEW,” *ASIAN J. Comput. Sci. Inf. Technol.*, vol. 3, no. 1, 2013.
- [50] F. Murtagh and P. Contreras, “Algorithms for hierarchical clustering: an overview,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 2, no. 1, pp. 86–97, 2012.
- [51] L. Kaufman and P. J. Rousseeuw, “Clustering by Means of Medoids [w:] Statistical Data Analysis Based on The LI-Norm and Related Methods, red. Y. Dodge.” North-Holland, Amsterdam, 1987.
- [52] L. Kaufman and P. J. Rousseeuw, “Partitioning around medoids (program pam),” *Find. groups data an Introd. to Clust. Anal.*, pp. 68–125, 1990.

-
- [53] R. T. Ng and J. Han, “Efficient and Effective Clustering Methods for Spatial Data Mining”, Proc. 20th Int. Conf. On Very Large Data Bases, Santiago, Chile, Morgan Kaufmann Publishers,” 1994.
- [54] J. A. Hartigan and M. A. Wong, “Algorithm AS 136: A k-means clustering algorithm,” *J. R. Stat. Soc. Ser. C (Applied Stat.)*, vol. 28, no. 1, pp. 100–108, 1979.
- [55] R. T. Ng and J. Han, “CLARANS: A method for clustering objects for spatial data mining,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002.
- [56] S. Guha, R. Rastogi, and K. Shim, “CURE: an efficient clustering algorithm for large databases,” in *ACM Sigmod Record*, 1998, vol. 27, no. 2, pp. 73–84.
- [57] S. Guha, R. Rastogi, and K. Shim, “ROCK: A robust clustering algorithm for categorical attributes,” *Inf. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.
- [58] W. Wang, J. Yang, R. Muntz, and others, “STING: A statistical information grid approach to spatial data mining,” in *VLDB*, 1997, vol. 97, pp. 186–195.
- [59] T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: an efficient data clustering method for very large databases,” in *ACM Sigmod Record*, 1996, vol. 25, no. 2, pp. 103–114.
- [60] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, and others, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Kdd*, 1996, vol. 96, no. 34, pp. 226–231.
- [61] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, “Density-based clustering in spatial databases: The algorithm gdbscan and its applications,” *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, 1998.
- [62] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “OPTICS: ordering points to identify the clustering structure,” in *ACM Sigmod record*, 1999, vol. 28, no. 2, pp. 49–60.
- [63] “MapD.” [Online]. Available: <https://www.mapd.com/>.
- [64] T. M. Mitchell, “Machine learning (mcgraw-hill international editions computer science series),” 1997.
- [65] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.
- [66] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, “Community detection in social media,” *Data Min. Knowl. Discov.*, vol. 24, no. 3, pp. 515–554, 2012.
- [67] J. S. Alowibdi, S. Ghani, and M. F. Mokbel, “VacationFinder: A tool for collecting, analyzing, and visualizing geotagged Twitter data to find top vacation spots,” in *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, 2014, pp. 9–12.
- [68] M. Gupta, J. Gao, C. Zhai, and J. Han, “Predicting future popularity trend of events in microblogging platforms,” *Proc. Assoc. Inf. Sci. Technol.*, vol. 49, no. 1, pp. 1–10, 2012.
- [69] A. Madkour, W. G. Aref, F. U. Rehman, and M. A. Rahman, “A Survey of Shortest-Path Algorithms,” pp. 1–26, 2017.
- [70] C. S. T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to algorithms*, vol. 53, no. 9. 2001.
- [71] E. W. Dijkstra, “A note on two problems in connection with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.
- [72] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

-
- [73] E. V. Denardo, "Dynamic Programming: Models and Applications, 2003." Dover Publications: New York.
- [74] M. Sniedovich, *Dynamic programming: foundations and principles*. CRC press, 2010.
- [75] M. Sniedovich, "Dijkstra's algorithm revisited: The dynamic programming connexion," *Control Cybern.*, vol. 35, no. 3, pp. 599–620, 2006.
- [76] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM*, vol. 34, no. 3, pp. 596–615, 1987.
- [77] M. L. Fredman and D. E. Willard, "BLASTING Through the Information Theoretic Barrier with FUSION TREES," in *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, 1990, pp. 1–7.
- [78] M. L. Fredman, "Trans-dichotomous algorithms for minimum spanning trees and shortest paths," *J. Comput. Syst. Sci.*, vol. 48, no. 3, pp. 533–551, 1994.
- [79] P. van Emde Boas, "Preserving Order in a Forest in Less Than Logarithmic Time," in *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, 1975, pp. 75–84.
- [80] M. Thorup, "On RAM Priority Queues," in *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 1996, pp. 59–67.
- [81] A. Bernstein and L. Roditty, "Improved Dynamic Algorithms for Maintaining Approximate Shortest Paths Under Deletions," in *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, 2011, pp. 1355–1365.
- [82] X. Wang, Y. Zhao, L. Nie, Y. Gao, and S. Member, "Semantic-Based Location Recommendation With Multimodal Venue Semantics," vol. 17, no. 3, pp. 409–419, 2015.
- [83] M. Raubal and S. Winter, "Enriching Wayfinding Instructions with Local Landmarks."
- [84] S. Wakamiya *et al.*, "Route Recommendation System by Extracting Popular and Visible Landmarks," *Inf. Process. Soc. Japan*, vol. 162C50, pp. 683–686, 2016.
- [85] N. Powdthavee, "Unhappiness and crime: evidence from South Africa," *Economica*, vol. 72, no. 287, pp. 531–547, 2005.
- [86] P. Salesses, K. Schechtner, and C. A. Hidalgo, "The collaborative image of the city: mapping the inequality of urban perception," *PLoS One*, vol. 8, no. 7, p. e68400, 2013.
- [87] "Open Government Data USA." [Online]. Available: <https://www.data.gov/%0A>.
- [88] "Open Government Data Australia." [Online]. Available: <https://data.gov.au/>.
- [89] "Open Government Data Canada." [Online]. Available: <https://open.canada.ca/en/open-data>.
- [90] G. Schoier and G. Borruoso, "Spatial data mining for highlighting hotspots in personal navigation routes," *Int. J. Data Warehous. Min.*, vol. 8, no. 3, pp. 45–61, 2012.
- [91] G. Schoier and G. Borruoso, "Individual movements and geographical data mining. Clustering algorithms for highlighting hotspots in personal navigation routes," in *International Conference on Computational Science and Its Applications*, 2011, pp. 454–465.
- [92] J. Kim and T. Sandholm, "SocRoutes : Safe Routes Based on Tweet Sentiments," pp. 179–182.
- [93] S. Shah, F. Bao, C.-T. Lu, and I.-R. Chen, "CROWDSAFE: Crowd Sourcing of Crime Incidents and Safe Routing on Mobile Devices (Demo Paper)," *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst. - GIS '11*, p. 521, 2011.
- [94] J. Kortge and J. Zhang, "System and method for providing safety-optimized navigation route planning." Google Patents, 2006.
- [95] J. Van Den Berg and M. Overmars, "Planning the shortest safe path amidst

- unpredictably moving obstacles,” in *Algorithmic Foundation of Robotics VII*, Springer, 2008, pp. 103–118.
- [96] S. Suh and K. G. Shin, “Robot path planning with distance-safety criterion,” in *Decision and Control, 1987. 26th IEEE Conference on*, 1987, vol. 26, pp. 634–641.
- [97] A. Lambert and D. Gruyer, “Safe path planning in an uncertain-configuration space,” in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 2003, vol. 3, pp. 4185–4190.
- [98] L. Leenen, A. Terlunen, and H. Le Roux, “A constraint programming solution for the military unit path finding problem,” Taylor & Francis Group, 2012.
- [99] R. V Helgason, J. L. Kennington, and K. H. Lewis, “Shortest Path Algorithms on Grid Graphs with Applications to Strike Planning,” 1997.
- [100] S. Mittal and K. Deb, “Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms,” in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 3195–3202.
- [101] S. A. Bortoff, “Path planning for UAVs,” in *American Control Conference, 2000. Proceedings of the 2000*, 2000, vol. 1, no. 6, pp. 364–368.
- [102] A. Ahmad *et al.*, “A Framework for Crowd-Sourced Data Collection and Context-Aware Services in Hajj and Umrah,” 2014.
- [103] A. Ahmad, M. A. Rahman, I. Afyouni, F. Ur Rehman, B. Sadiq, and M. R. Wahiddin, “Towards a mobile and context-aware framework from crowdsourced data,” *5th Int. Conf. Inf. Commun. Technol. Muslim World*, pp. 1–6, Nov. 2014.
- [104] B. Krogh, E. Lewis-kelham, N. Pelekis, and K. Torp, “Trajectory Based Traffic Analysis,” pp. 546–549, 1861.
- [105] H. Samet, “The Quadtree and Related Hierarchical Data Structures,” *ACM Comput. Surv.*, vol. 16, no. 2, pp. 187–260, 1984.
- [106] “Osm2pgsql.” [Online]. Available: <https://github.com/openstreetmap/osm2pgsql>.
- [107] K. Gimpel *et al.*, “Part-of-speech tagging for twitter: Annotation, features, and experiments,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, 2011, pp. 42–47.
- [108] “OSMPOISPBF.” [Online]. Available: <https://github.com/MorbZ/OsmPoisPbf>.
- [109] “OSMPgRouting.” [Online]. Available: <https://github.com/pgRouting/osm2pgrouting>.
- [110] J. B. Harley, *The new nature of maps: essays in the history of cartography*, no. 2002. JHU Press, 2002.
- [111] M. Avvenuti, S. Cresci, A. Marchetti, C. Meletti, and M. Tesconi, “EARS (Earthquake Alert and Report System): A Real Time Decision Support System for Earthquake Crisis Management,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 1749–1758.
- [112] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks.” 2008.
- [113] F. U. Rehman, I. Afyouni, A. Lbath, S. Khan, S. M. Basalamah, and M. F. Mokbel, “Building Multi-Resolution Event-Enriched Maps From Social Data,” in *Proceedings of the 20th International Conference on Extending Database Technology, {EDBT} 2017, Venice, Italy, March 21-24, 2017.*, 2017, pp. 594–597.
- [114] S. Atta, B. Sadiq, A. Ahmad, S. N. Saeed, and E. Felemban, “Spatial-crowd: A big data framework for efficient data visualization,” in *Big Data (Big Data), 2016 IEEE International Conference on*, 2016, pp. 2130–2138.
- [115] J. Liu, H. Li, Y. Gao, H. Yu, and D. Jiang, “A geohash-based index for spatial data

- management in distributed memory,” in *Geoinformatics (GeoInformatics), 2014 22nd International Conference on*, 2014, pp. 1–4.
- [116] A. K. Karun and K. Chitharanjan, “A review on hadoop - HDFS infrastructure extensions,” in *2013 IEEE Conference on Information Communication Technologies*, 2013, pp. 132–137.
- [117] “Map-icons.” [Online]. Available: map-icons.com/.