Interactive visualization for spatial data exploration

By

Tatiana K. Noboa

Department of Electrical and Computer Engineering
University of Queensland

Submitted for the degree of Master in Computer Sciences in the division of Information Technology
June 2014
Dear Professor Strooper,

In accordance with the requirements of the degree of Master in Computer Sciences in the division of Computer Systems Engineering, I present the following thesis entitled "Interactive Visualization for Spatial Data Exploration". This work was performed under the supervision of Dr. Mohamed A. Sharaf.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

Tatiana Noboa
AUTHOR'S NAME.
To my daughter, Alicia.
Acknowledgements

I would like to express my deep gratitude to Professor Mohamed Sharaf my research supervisor, for his patient guidance, enthusiastic encouragement and accurate observations about this research work. I would also like to thank Abdullah Albarrak and Hina Khan for their advice and assistance for a better comprehension of the techniques of diversification and query refinement involved in this project. My grateful thanks are also extended to the professors and tutors in the computer department for their help in sharing with me their knowledge and explanations.

I would like to emphasize my thanks to the Ecuadorian government which trough SENECYT granted me the scholarship that made possible my studies in Australia.

Additionally, I would like to extend my thanks to Carlos de Oliveira for his help, and useful explanations and aid about Spring framework, Maven and other useful tools functioning and usage.

My gratitude to my grand aunt Alicia Mayorga who traveled from Ecuador to help me with my daily issues at the end of first and second semester and to Richard Green who made it possible the last phase of this studies with his constant aid and support.

Finally, I wish to thank my daughter for her support, patience, affection and responsible behavior throughout my study and our stay in this country.
Abstract

The present work has the objective of constructing a software platform that combines appropriate libraries and technologies to interactively apply visual perception of geographical information, which will encourage the exploration, observation and validation of hypotheses with the objective of gaining an intuitive knowledge of a given subject when examining sets of data and techniques over this groups, besides, the product of the present work aim is to aid the comprehension of data exploration techniques as data diversification and query refinement techniques.

To achieve this goal the stated platform combines various web technologies and services that allows the displaying of data exploration results in a spatial context. The functioning of the software environment is demonstrated in a web interactive platform that acting over two databases of geo-spatial information shows the behavior of max-min and max-sum dispersion methods for data diversification, and similarity aware query refinement SAQR techniques.

Finally, the mentioned techniques are show-cased by solving specified real life scenarios, the first one illustrates the evenly distribution of cities among a rectangular area or country according to max-min diversification technique and the boundary distribution according to max-sum diversification technique, this could guide a tourist on visiting diverse cities on a determined country for example. The second one illustrates SAQR method of query refinement, which specifically aims to guide police chiefs in allocating zones of operation for their under command officers using a real, historical dataset of crime incidents of the city of San Diego, CA in USA.
# Table of Contents

1. **Introduction** .................................................................................................................. 1

2. Literature review / prior art. ................................................................................................ 2
   2.1 Data exploration. ........................................................................................................... 2
       2.1.1 Search result diversification. .............................................................................. 3
       2.1.2 Query refinement. ............................................................................................... 4
   2.2 Data visualization. ........................................................................................................... 6
       2.2.1 Web Mapping APIs ............................................................................................. 7
       2.2.2 Information visualization toolkits for web applications. .................................... 9
       2.2.3 Development tools for integration of software components............................... 11

3. **Visualization for data exploration platform.** ................................................................. 18
   3.1 Problem Overview (application scenarios). ................................................................. 18
   3.2 Application functionality. ............................................................................................. 19
       3.2.1 DiverVisual functionality. ............................................................................... 19
       3.2.2 RefineVisual functionality. ............................................................................... 21
   3.3 System Architecture. .................................................................................................... 22
       3.3.1 User Interface ..................................................................................................... 24
       3.3.2 Google Maps Service and D3.js library. ............................................................. 27
       3.3.3 Data diversification and query refinement libraries. ......................................... 31
       3.3.4 Database ........................................................................................................... 37

4. **Conclusions** .................................................................................................................... 40
   4.1 Summary and conclusion ............................................................................................... 40

5. **Companion disk.** ............................................................................................................. 42

6. **Bibliography** .................................................................................................................... 43
List of Figures

Figure 1: Query refinement in two dimensional space ................................................................. 6
Figure 2: Google Maps API Drawing Library. .................................................................................. 8
Figure 3: Google Maps Drawing Manager adaptation to obtain rectangular area coordinates when
finishing drawing. ............................................................................................................................ 8
Figure 4: D3 scatter plot, data obtained from csv file, graphic includes html elements. ............. 11
Figure 5: Selecting a set of cities to be diversified by country or by rectangular area on a map. . 20
Figure 6: DiverVisual max-sum, max-min, by map, by country vizualization sample combinations........................................................................................................................... 21
Figure 7: Query refinement initial query drawing. ................................................................. 22
Figure 8: Query refinement, resulting query visualization. ............................................................... 22
Figure 9: System Architecture graphic representation. ............................................................... 23
Figure 10: Cloropleth, taken from https://github.com/mbostock/d3/wiki/Gallery. ................. 29
Figure 11: DiverVisual bean .................................................................................................... 32
Figure 12: DiverVisual java classes .......................................................................................... 33
Figure 13: RefineVisual Bean ................................................................................................... 35
Figure 14: RefineVisual java classes ......................................................................................... 36
Figure 15: DiverVisual EER ..................................................................................................... 37
Figure 16: RefineVisual EER ..................................................................................................... 39
Chapter 1

Introduction

The vast amounts of data generated nowadays is recorded in large storage capacity, in high velocity devices and in variety of formats. Many innovative methods have been proposed by researches to process, search and extract knowledge from such massive data stores. For example, data diversification techniques extract meaningful and brief views from the huge result sets gathered by database management systems and query refinement modifies a query according to given user requirements.

However, even with the existence of such supportive techniques, naïve and expert users would prefer to visualize the data in a meaningful and intuitive way, moreover the experts will need to visualize the data in order to show case thesis about new algorithms and methods, therefore, having an application made for this purpose, specifically on the data in which the experiments are taking place, will save valuable time.

Experiments on diversification techniques are being tested with spatial data, latitudes and longitudes of locations in the globe, mainly because the final dispersion of the results is intuitively understandable by looking at the results plotted on a real map. There are many tools designed to visualize information nowadays, but the process of visualizing the data takes many steps from the retrieval of the specific data from many possible different sources, building efficient algorithms, choosing the best indexing method, processing it, choosing and preparing the tool to visualize it, charging the data, making the specific configurations, and sharing it if necessary.

In this context, the present work will implement a mash up of technologies, and the corresponding algorithms to visualize the process and results of diversification and query refinement methods, additionally it will be described the architecture and functioning of the two resulting applications called diverVisual and diverRefinement.
Chapter 2

Literature review / prior art.

In this section are introduced the methods and algorithms as well as concepts used to perform the tasks of diversification and refinement which are techniques used for data exploration applications, along with a brief description on the most relevant software and libraries used to construct the application.

2.1 Data exploration.

Among the practices and concepts commonly exercised nowadays by the scientific community, there is a class of applications referred collectively as Interactive Data Exploration (IDE), in which large, amorphous data sets are explored in a multi-step, nonlinear process with inprecise end goals, this is different from searching in traditional DBMS in which the structure, meaning and contents of the database, as well as the questions to be asked are already well understood [1].

In order to achieve efficient data exploration various techniques have been used, this methods are classified by authors into two broad sub-categories: Query Navigation and Data Summarization. Query refinement, recommendation and formulation are contained in the field of query navigation, which goal is to facilitate the quick finding of regions in large data sets. Accordingly, data diversification is a data summarization technique, the methods in this sub-category aim to provide systems with a concise and meaningful summarized representation of the data, in particular data diversification returns a small set of results that summarizes the universe in a given topic, without requiring the specification of a weighted utility function [2].

The present work will show case, search result diversification and query refinement techniques:
2.1.1 Search result diversification.

Web search and recommender systems are mostly used nowadays to explore the variety of options that can be provided about a certain topic, query or set of words. When performing a search on a particular theme, it would be necessary to avoid homogeneity in the results and try to make available to the user a dissimilar number of outcomes that represents all the categories related to that theme existing in the consulted data source. Consider, for example a tourist that wants to visit some cities of a country during his stay, it would be useful, for instance, to provide them with a set of cities evenly distributed among the territory of the country so the tourist can go over the different landscapes and climate regions of it [3].

Diverse items are defined in terms of content, novelty and coverage, in this context there are various algorithms for result diversification, classified according to the reviewed literature into two key groups: greedy and interchange [3]. In the present document the attention is centered in greedy methods.

“The problem of selecting diverse items can be expressed as follows. Given a set X of n available items and a restriction k on the number of wanted results, the goal is to select a subset $S^*$ of k items out of the n available ones, such that, the diversity among the items of $S^*$ is maximized” [3].

Search result diversification objective is to: given a user query, return a list of diverse relevant results (i.e. records, documents) in order to satisfy a particular user information need [4]. The formal definition of search results diversification is given as follows: Let $X = \{x_1, x_2, \ldots, x_m\}$ be a set of results for some user query. In general, the goal of result diversification is to select a subset $S^*$ of $X$ with $|S^*| = k$, $k < m$, such that the diversity of the results in $S^*$ is maximized [2].

$$S^* = \arg \max_{S \subseteq X} f(S, d)$$

This work will focus in the user content based definition of diversification, which is an instance of the p-dispersion problem, whose objective is to maximize the overall dissimilarity within a set of selected objects. In particular, given a metric $d$ that measures the distance between two results, e.g., the Euclidean distance among two data points, where the diversity of a set $S$ is measured by a diversity function $f(S, d)$ that captures the dissimilarity between the results in $S$ [2].

Various algorithms has been proposed to solve the facility dispersion problem used to the further purpose of diversification, among them, this work is centered in MaxSum and MaxMin algorithms as described below.
2.1.1.1 Max-sum dispersion based diversification

Has the objective of maximizing the sum of all pair-wise distances between points in the set \( S \) given a distance function that calculates the dissimilarity between 2 given records \( d(u, v) \).

Algorithm for MaxSum dispersion problem [5].

**Input:** Universe \( U \), \( k \)

**Output:** Set \( S (|S| = k) \) that maximizes \( f(S) \)

Initialize the set \( S = \emptyset \)

for \( i \leftarrow 1 \) to \( \lfloor k/2 \rfloor \) do

Find \( (u, v) = \arg\max_{x, y \in U} d(x, y) \)

Set \( S = S \cup \{u, v\} \)

Delete all edges from \( E \) that are incident to \( u \) or \( v \)

end

If \( k \) is odd, add an arbitrary record to \( S \)

2.1.1.2 Max-min dispersion based diversification

Aims to maximize the minimum relevance and dissimilarity of the selected set.

Algorithm for MaxMin dispersion problem [5].

**Input:** Universe \( U \), \( k \)

**Output:** Set \( S (|S| = k) \) that maximizes \( f(S) \)

Initialize the set \( S = \emptyset \)

Find \( (u, v) = \arg\max_{x, y \in U} d(x, y) \) and set \( S = \{u, v\} \);

For any \( x \in U \setminus S \), define \( d(x, S) = \min_{u \in S} d(x, u) \);

while \( |S| < k \) do

Find \( x \in U \setminus S \) such that \( x = \arg\max_{x \in U \setminus S} d(x, S) \);

Set \( S = S \cup \{x\} \);

end

2.1.2 Query refinement.

Query refinement is another effective method to guide the user on the data exploration task, in particular it enables database systems to automatically adjust a submitted query so the result satisfies some requirements that have been pre specified. One example of those requirements is to set a constraint on the cardinality of the result set, this provides a solution to the problem of returning too few or too many records as a result of a query [6].
This last technique called cardinality based query refinement has the purpose of quickly navigate a large search space of possible refined queries and return one refined query such that the cardinality of its result is very close to a pre-specified cardinality constraint, returning the query with a minimized deviation between the pre-specified cardinality and the archived one. Some methods have been proposed to solve cardinality based query refinement (e.g., Hill Climbing), however such methods in spite of providing efficient solutions to the problem, does not take in account the dissimilarity between the input query and its corresponding refined version [6].

In order to propose a solution for the mentioned limitation of the current cardinality-based query refinement techniques it has been proposed a similarity – aware query refinement which meet a specified cardinality constraint and maximize the similarity between the input and output queries [6], provided to this work as java pre-maid libraries designed and adapted to spatial contexts.

The similarity aware cardinality based query refinements can be applied to solve problems in the real world where spatial queries are used, for example, consider the next example where range queries over spatial attributes are performed.

Consider the coordination of police officers that must be allocated in services zones distributed around a city, in which locations of previous incidents are known and take into account that each officer can handle a specific number of incidents \( K \) per day. With such purpose the coordinator allocates a rectangular area for an officer to be in charge of, hence if the number of incidents in selected area or zone has a history of having more incidents than \( K \), then the probability is that some incidents will be left without responsiveness from the officer and on the other hand, if the number of incidents is less than \( K \), then it will lead to waste of police human resources [6].

Also observe that the resulting area (range query) containing the \( K \) needed incidents should be very similar to the initial area containing them, otherwise the solution to propose a total different area would be quite inadequate. In order to give a solution to this problem a technique called SAQR has been proposed, this technique is a refinement method that satisfies the cardinality constraint and also takes into account the similarity that must exist between the initial queries and refined one, this means that the technique will minimize the dis-similarity between the submitted and obtained query [6].
Let’s consider the previous figure. It shows a 2-dimensional space with two attributes $a_1$ and $a_2$ (e.g., Latitude and Longitude coordinates of a point in a world map). The initial range query $I$ is refined to satisfy a cardinality constraint (e.g., $k = 12$). Assume $R_1$ and $R_2$ are two refined queries which both satisfy the same cardinality constraint. Clearly, $R_2$ is essentially different from query $I$, while $R_1$ returns the same cardinality with much more similarity to $I$. That is, the refined query $R_1$ is largely accepted by the user than $R_2$. Emphasizing with the illustration. SAQR aims to meet some pre-specified cardinality constraint $k$ on the refined query $R$ and minimize the distance between the refined query $R$ and the initial query $I$ under some given distance function $D()$, ideally this distance should be 0 which is quite difficult and unrealistic to achieve.

Hence, in this work, we adopt a multi-objective model that particularly captures and quantifies the overall deviation in meeting the user expectations for both similarity and cardinality [7]. Reducing this deviation is essential, and it is one of the objectives of the used SAQR scheme.

### 2.2 Data visualization.

In the context of software applications, it is not uncommon to find data visualizations with too many colors or strident ones, excessive ornamentation or too much text. After the application has been accomplish its objectives in retrieving and processing the data, the presentation and visualization of it is what makes the information understandable for the end user. The design of the end interface could have redundant and sometimes useless elements that will make loose valuable time to the end user, and worse than that make the work underneath seem not as relevant as it is.

Besides, a visualization must be easy to comprehend, accurate, esthetically pleasant and
appropriate to the context, however, the complexity of the underneath processing of the data could lead the developer to choose certain tools that allow him to present the information depending on the feasibility of integration with the most adequate software chosen to develop it, taking in account velocity of processing, scalability, supported data structures, client interface, that is why we will focus our attention in searching the most suitable software packages to accomplish this goal.

The present work is also specific for the visualization of diversification and query refinement techniques in a spatial context, which means that it will be required tools that allow to present information in a customized generated map to be marked with points, lines, areas and to place text and HTML in it, also a second tool that facilitate the manipulation of that information or its representation as visual objects in top of the previously described map, furthermore this tools should be capable to act in a web environment.

It has been set as the started point for the development of the application, that it retrieves data form a MySql managed data base, and then this data will be diversified or refined according to the case using Java data structures and the corresponding implemented algorithms for the case, in this context the investigation of the state of the art in this section will consist in looking for tools that can be integrated in a web environment that presents these initial characteristics.

According to the mentioned above it is needed to choose two specific software tools, the first one should enable the application to display data in a geographical map layer in an interactive way (Web Mapping API), and the second one must allow to manipulate that data represented as visual objects over the map.

2.2.1 Web Mapping APIs

A web mapping API, is an application program interface or a collection of programming utilities designed to generate and visualize geospatial data using the web over lays of drawn world maps that can be utilized for the implementation of a web map without the need to program the elementary functionalities from scratch [8]. This technologies try to provide the used data according to the OpenGis Consortium specification, in order to achieve interoperability among applications using geospatial data. This tools are provided with the underlying maps from maps services or map server back-ends such as WMS, Google Maps, Yahoo! Maps, ESRI ArcGIS, WFS, or OpenStreet Maps among others. The present section pretends to give a brief overview of some of the available web mapping libraries for web.

2.2.1.1 Google Map API

It is a Java Script Library, consisting on multiple map related functions that control the appearance of the map, it makes it possible to incorporate user defined maps on websites, and to overlay information from other sources, the use of the library is free for small applications [9], with the shortcomings that would eventually start inserting ads in users applications and it is not
open source. The current Google Maps v3 was introduced specifically to meet the needs of online mapping through mobile devices. It reduces the amount of data communications overhead, thus increasing the speed of map display [9].

The API also offers drawing functionality which provides a graphical interface for users to draw polygons, rectangles, polylines, circles, and markers on the map with additional functionalities derived from the mentioned before.

![Figure 2: Google Maps API Drawing Library.](image)

![Figure 3: Google Maps Drawing Manager adaptation to obtain rectangular area coordinates when finishing drawing.](image)
The previous figure shows one of the functionalities of drawing manager component in Google Maps drawing library, retrieving the coordinates of a drawn rectangular area when the user finish the drawing, this functionality will be extensively used later in this project.

2.2.1.2 Microsoft (Bing Maps) and Yahoo MAP APIs

In 2005, Microsoft and Yahoo changed their online mapping service to incorporate an AJAX-type interface. By 2006, Yahoo had released its own API. The Yahoo Maps API is much the same as Google’s implementation but does not support polygons and still requires the use of an electronic key. While the key is made freely available, it limits the use of the API to the server that is specified when the key is requested. In mid-2009, Microsoft relabeled its Live Local web mapping service to Bing Maps, a part of the company’s search engine services. In contrast to the Yahoo Maps API, Bing Maps does support polygons. Both applications work with a JavaScript based and a Flash-based API to embed their map contents [9].

2.2.1.3 Open Layers Map API

Open Layers is free, Open Source Java Script client side Library released, under 2-clause BSD License (also known as the FreeBSD) [10]. Free software with open source presents the great advantage of making the code available to developers, which is remarkable in an academic environment. OpenLayers implements industry-standard methods for geographic data access. Openlayers is not tied to any proprietary technology or company [11].

This library has to be set on the client side, one of the tasks the client performs through this tool is to get map images from a map server, every time the user navigate or zoom the map, this happens via asynchronous JavaScript (AJAX) calls to a map server. The map server provides the map itself, there are many server backends as WMS, Google Maps, Yahoo Maps, ESRI ArGIS, WFS and OpenStreet Maps, OpenLayers can be used to perform calls to any of them, but it is not a web server it only consumes data from them [11], OpenLayers allows to have multiple different backend servers for the map can use.

2.2.2 Information visualization toolkits for web applications.

Multiple tools and a combination of varied technologies are employed simultaneously when designing visualizations in a web application, this is possible by the shared representation of the page (DOM: document object model), unfortunately this interoperability is typically lost with visualization toolkits due to encapsulation of the DOM with more specialized forms, hence, sometimes augmenting costs and difficulty in efficiency, accessibility, expressiveness [12].

It has been developed a variety of toolkits for facilitating visualization design. This section aims to present the generalities of two data visualization toolkits that can be integrated in web
applications, defining a visualization toolkit as one or a series of libraries that allows to represent and convert data in graphics over a web page, for the purposes of this work the main interest is for the toolkit to be able to integrate with the chosen Map API and the underlying technology to achieve the diversification and refinement process over the data.

2.2.2.1 Protovis

Protovis is a graphical toolkit, designed for visualization, is the antecessor of D3, developed by the same creators and no longer under active development as the team is working in development and improvement of D3. The toolkit uses JavaScript and SVG for web native visualizations, is free and open-source, provided under the BSD License. The last release was launched in September 2010 [13].

2.2.2.2 D3 Data Driven Documents.

Data Driven Documents (D3) is an embedded domain-specific language for transforming the document object model based on data, this toolkit is highly compatible with other web tools, support debugging and allows interaction and animation. D3 let designers, to selectively bind input data to arbitrary document elements, applying dynamic transforms to generate and modify content. D3 has been specifically designed to efficiently manipulate documents based on data [12].

The toolkit uses the universal concept of selection that identify a set of elements using simple predicates, then apply a series of available operations that mutate the selected elements, as it is done with CSS and JavaScript, but augmenting mechanisms to add and remove elements to match a data set, bounding nodes to every used data record, producing enter and exit sub selections for the creation and destruction of elements in correspondence with data and allowing transitions over those nodes. Special operators called event handlers respond to user input and enable interactions, with numerous helper modules such as layouts and scales that simplify common visualization tasks, facilitating the app construction job for high level visual abstractions [12].

D3 is suitable for users that have software engineering expertise and also for more general audiences such as web designers, using declarative, domain specific languages (DSLs) for visualization design, which decouples specification from execution details, allowing user to focus on the specific of their application domain, with the main feature of mapping data to visual elements directly in the document object model. The toolkit visual custom abstractions can facilitate portability and performance optimization, it also allows to exploit supporting technologies such as CSS [12].

D3 Adopts the W3C selectors API to identify document elements for selection [12]. The previous makes learning the tool easier, given the familiarity produced due to the extensive use of
the selectors API in commonly used tools as JQuery. Similarly, any number of HTML and CSS
can be applied to selected elements, the toolkit also supports method chaining.

There are numerous example visualizations available in D3 web site, as well as some
illustrations of D3 integration with mapping APIs, which results determining useful for the present
work.

![Figure 4: D3 scatter plot, data obtained from csv file, graphic includes html elements.](image)

The previous figure is a representation of locations (latitude and longitude) in a d3
generated scatter plot, representing cities of Greece, taken from a csv file, and then presented in a
web page.

### 2.2.3 Development tools for integration of software components.

#### 2.2.3.1 Java

Java comprises much more than just a programming language; in fact, java can be
considered as an entire ecosystem comprising the programming language, development, test,
bug, monitoring and management tools, libraries, applications servers and a large variety of
applications. The java language developer usage is growing year after year since the late 1990s
from tens and hundreds to over nine million Java developers nowadays [14].
The java language was originally designed for embedded consumer-electronics applications by James Gosling and was called Oak at this time. Several years later this original development, after receiving contributions from other experts, was renamed to Java and retargeted to the Internet [15].

Since its launching, Java has gained market share over more traditional programming languages, notably over C and C++ as modern general-purpose programming languages due to some aspects as for example: simplicity, easy to learn and use [15 p. 1]. Other important aspect that promotes the Java language is security, what is of fundamental importance in the modern connected word where many aspects of people's life depends on secure and reliable data communication and storage [14 pp. 6-7]. Java was developed from the start as a secure language where the run-time system cannot be subverted by code written in Java [15 p. 319].

In addition to the aforementioned features, java offers: Platform independency: the Java compiled “bytecode” will run on any kind of machine as long as it exists a Java virtual machine developed for this machine. Compiled and interpreted: here is the major difference between Java and C, C++, and other languages, the java code is compiled to bytecodes that are interpreted by the Java virtual machine, and there is no link process. Robustness: the lack of pointers and the automatic garbage collection free the developer of the burden of having to deal with pointer security issues and memory leaks [14 p. 6-8].

The Java language has undergone many revisions and improvements over the last 15 years resulting in a full-fledged platform for creating, deploying and running a wide range of applications including scientific and engineering applications that needs high performance and robustness by the use of multiprocessing and high-performance garbage collector implementations [15 p. 1, 5].

In the early days of Java the interpretation of the bytecode, that makes Java portable, was a real shortcoming for the development of scientific applications due to the big performance reduction, but since of the advent of the development of the recent and more sophisticated Java Virtual Machines (JVMs) with a resource called Just-in-Time compilers, this kind of issue was largely minimized or made almost completely irrelevant [16 p. 10-11].

The so-called Just-in-Time compilers increases the performance of Java applications by compiling the most used sessions of the java bytecode to optimized local machine code what can lead to faster performance than C and C++ programs in some cases [16 p. 11]. This in addition to the fact that Java has a sophisticated and complete support to develop web applications makes Java a very good choice when selecting a language to develop scientific applications with web support.
2.2.3.2 Spring framework

The Spring Framework is one of the nowadays most used Java frameworks and an expressive example of a successful open-source project with a large and active community of contributors and users [17]. The open source project started from the source code of the book Expert One-on-One J2EE Design and Development [18] that differently of other books was design for use with real applications instead of just demonstration purposes.

The basic motivations for the development of the framework were [18 p. 144]:

- “Addressing areas not served by other frameworks at the time and provide an end-to-end solution to integrate specialized frameworks in a coherent way.
- Easy adoption by been clearly divided on layers to allow the use of features without imposing a unique view over the application.
- Easy to use by making simple tasks easy to achieve without trade-offs and at the same time allowing developers to leverage more complex tasks to J2EE services.
- Fostering the use of best practices as for example: programming to interfaces, rather than classes.
- Application objects should have minimal dependencies on the framework (Non-invasiveness).
- A consistent configuration to keep the application configuration process flexible and consistent.
- Easy of testing to permit easy unit testing of whole applications or individual classes.
- Allows easy extensibility to permit easy customizations if necessary.”

One of the biggest services that the Spring Framework provides is the inversion of control mechanism that permits the easy “wiring” of the application components that composes each part of the final solution in a declarative fashion [17 p. 54-61]. Furthermore, Spring also provides may other high-level specialized solutions as for example: aspect oriented programming, relational and non-relational databases support, task scheduling, web application development via a full-fledged MVC framework, high level security resources among others [17 p.1-11].

In summary, Spring Framework simplifies development of applications, reducing the complexity by providing a set of tools that prevents the developer from programming tasks besides the application problem and avoiding the “plumbing work.”

2.2.3.3 JavaScript (JS)

JavaScript is a dynamic computer programming language that is most commonly used as a part of web browsers, the implementation of JavaScript allow client side scripts to interact with the web page modifying the content of the document as needed to be displayed. “JavaScript is a
high-level, dynamic, untyped interpreted programming language that is well-suited to object-oriented and functional programming styles. JavaScript derives its syntax from Java, its first-class functions from Scheme, and its prototype-based inheritance from Self” [19].

The overwhelming majority of modern websites use JavaScript, and all modern web browsers consoles include JavaScript interpreters, these browsers are placed on desktops, game consoles, tablets, smart phones among others, this makes JavaScript to be considered the most ubiquitous programming language in history. It is also thought that a web developer must learn 3 main technologies as a strong requisite: HTML to specify the content of web pages, CSS to specify the presentation of web pages, and JavaScript to specify the behavior of web pages [19].

JavaScript was originally developed in 1995 with the purpose of validate users input data in the client side of web applications, prior to JavaScript the user data had to do a server round-trip to be validated by server side code as for example a Perl script. Avoiding unnecessary server round trips was a necessity at the time the due to the internet quite low speed [20 p. 1].

Nowadays, JavaScript was evolved as language at the same time that this use grows until to take part of nearly all aspects of a browser window contents becoming an important feature in every web browser. During the language development, JavaScript suffered with multiple implementations built by concurrent companies what made web development much more complex at the point that at this time many web sites had to have different web pages aimed to different client browsers during the so called “Browsers Wars” [21]. This situation was causing fears in the internet industry that decided that the language must be standardized [20 p. 2].

The first proposal to standardize JavaScript was submitted to the European Computer Manufacturers Association (Ecma) in 1997 by a group of companies interested on the future of the browser scripting and with result the Ecma technical committee comes with ECMA-262 standard a few months later to define a new language called ECMAScript also adopted later by the International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) as a the ISO/IEC-16262 standard and since that the browser developers have tried to use ECMAScript as a basis for their JavaScript implementations [20 p. 2].

2.2.3.4 JQuery

JQuery is a powerful JavaScript library aimed to simplify JavaScript programming by handling complex code with relatively simple parallel instructions while taking care of making the code compatible between different web browsers.

Develop code that runs in various web browsers is normally a tedious and complex task due to differences and inconsistencies between the various JavaScript implementations, this was a common source of bugs and extra resources expenditure on testing and development a few years
ago. The inconsistencies and lack of standardization persists until nowadays and probably will exists forever. jQuery is useful to avoid to have to deal with great portion of these issues while producing a clean and easily managed code [22 p. 3-4].

The basic idea behind jQuery is leveraging the user previous knowledge about how CSS selectors works and transfer it to JavaScript by the use of a cross-browser method of iterating with the DOM [23 p. 15]. The Document Object Model (DOM) is platform and language neutral interface design to allow programs and scripts to access and update content, structure and style of documents also as offering methods to deal with document events. Web browsers uses DOM as an internal structure to represent a given document, a web page in this case, and that access is exposed to the user via JavaScript [22 p. 60].

Probably the most common task web developers perform with JavaScript is selecting DOM elements to operate over and it is one of the jQuery biggest strengths, instead of repeating the same basic code each time jQuery allows to select an element or a set of elements with a simple select call as shown below.

Code using pure JavaScript use to change some elements CSS class [22 p. 64]:

```javascript
var tr = document.getElementsByTagName("tr");
for ( var = i; i < tr.length; i++ ) {
    if ( tr[i].class === 'highlighted' ) {
        tr[i].class = 'normal';
    }
}
```

Same task using a jQuery selector [22 p. 64]

```javascript
$("tr.highlighted").removeClass("highlighted").addClass("normal");
```

In addition to the element selectors jQuery also offers extensive support to DOM manipulation tasks as for example element creation, element attributes manipulation and HTML generation [22 p. 88-90]

2.2.3.5 Ajax

AJAX (Asynchronous JavaScript and XML), or Ajax, is a Web architecture designed to develop interactive Web applications with pages in which the efficiency and the time of loading changes will be significantly improved by exchanging small amount of that with the server, hence
the entire Web Page does not have to be reloaded each time the user request a change, this is usually performed with the aid of special formatted files (i.e. XML or JSON) for the interchange of such portions of information.

Traditional web applications are based on a multi-page interface model, in which interactions are based on a page-sequence model, this is once obtained the HTML it will be parsed sequentially by the browser and the contents will be showed to the user if it is needed to change one part of the web document the web page must be fully reloaded. Ajax changes this by allowing asynchronous requests to be made after a page has been loaded and allowing JavaScript code to update parts of the page in the browser [24], this is making updates of only portions of the web page without reloading the full HTML in the page.

The ideal of exchanging data in a web page at the client side without unloading the web page had been around for some time, using different techniques to circumvent the web browser limitations in speed of processing and displaying contents and internet protocols restrictions since they was originally conceived just to serve content. This technique did not have an agreed name until 2005 when Jesse James Garrett wrote an article titled “Ajax: A New Approach to Web Applications” (http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/) and since that the techniques was referred as Ajax that is the shorter form for Asynchronous JavaScript plus XML.

2.2.3.6 JSON

The technique of issuing asynchronous requests from the browser to the server and receiving back responses has make it possible to improve user experience through generating web sites with quicker feedback, auto-completion with very few full page reloads. The requests are asynchronous because they are done in separate threads that do not block the main script execution, the information is passed from the server to the page and the other way around in form of JSON (JavaScript Object Notation), XML or plain text [25].

JSON is a way to represent data. JSON objects contain key value pairs representing the information to be serialized and transfer over a network connection, as mentioned before and in this case between web server and web application, being serialization a process to transform data structures and objects in a format suitable to be stored in a file or memory buffer or transmitted over a network connection, later this data can be retrieved. JSON representations are easily understood by humans, and is generally used for storing semi structured data [26].

“JSON is a lightweight text-based open standard data-interchange format. JSON is derived from a subset of JavaScript programming language (Standard ECMA-262 3rd Edition—December 1999). It is entirely language independent and can be used with most of the modern programming languages. JSON is a standard and is specified on RFC4627 on IETF (International Engineering Task Force). The specification is made by Doglus Crockford on July 2006. JSON files are saved
with .json extension. Internet media type of JSON is "application/json" [26].”

### 2.2.3.7 Twitter bootstrap

To simplify usage of html and CSS and enabling a richer user experience, twitter bootstrap has been utilized to aid with the appearance of the interface. It is a framework that provides a set of CSS classes and JavaScript functions as templates to facilitate the process of developing the application front end, the design features enable support for both mobile and desktop displays. The developed interfaces are cross browser and device compliant which means that the same site works in both desktop and mobile displays and in different browsers, this is markedly important as mobile usage overtakes conventional web access [27].
Chapter 3

Visualization for data exploration platform.

The Visualization for spatial data exploration platform (VSDEP) is a compendium of application libraries, utilities and software engineering practices that we consider are the most appropriate for solving the challenge of visualizing the previously described diversification and query refinement techniques, two examples has been constructed using this platform, the first one shows the visualization of diversification techniques such as max-min and max-sum dispersion and the second one named Orange (Objective aware range query refinement) show-cases SAQR-S and SAQR-CS query refinement techniques.

3.1 Problem Overview (application scenarios).

Three case scenarios have been considered to demonstrate the use of data exploration techniques in spatial context.

Scenario 1. First, let’s consider a tourist that is visiting a given country for a certain period of time, the person has limited resources and his aim is to visit as many different regions, climates, and cultures in diverse cities during his stay, if it is true that if we had the information available we could consider dissimilarity between cities based in the recently mentioned parameters is also true that it is possible to sample cities that are evenly distributed along the territory which will represent a set of diverse cities to be visited. So, the tourist could use our system to choose a country and a number of cities k to be visited, in this case the user should input the country and the number k of cities to be visited and the program will proportion the localization of k cities evenly dispersed in the chosen territory. This can be accomplished when the user chooses the max-min method for diversification.

Scenario 2. Secondly, consider a defense system for a country, the army could want to know which are the most vulnerable cities in the county, after many considerations they have decided that the bordering cities are the ones that need more protection and resources in case of conflict, similarly to the first scenario they have limited resources to shelter these cities so they should limit them to
a number k of cities. In this case the user should input the country and the number k of cities to be taken into account and choose max-sum method for diversification, the program will show then the localization of the k cities in the borders of the country.

The program should also be capable to accomplish the two before described functionalities, but instead of picking a determined country the user should be able to draw and pass as an input a rectangular area in which the process of diversification according to max-min or max-sum methods should be performed, the user should enter the required cardinality k and the program should provide the user with the location of the k cities in the chosen area, if max-min is chosen then the cities will be evenly dispersed along the rectangular area and if max-sum is chosen then the visualized cities will be in the borders of the drawn rectangular area.

Scenario 3. Finally, let us consider the coordination of police officers that must service allocated zones distributed around a city, in which locations of previous incidents are known, it has to be taken into account that each officer can handle a specific number of incidents k per day. With such purpose the coordinator allocates a rectangular area for an officer to be in charge of, hence if the selected area or zone has a history of having more incidents than k then the probability is that some incidents will be left without responsiveness from the officer and on the other hand, if the number of incidents is less than k then it will lead to waste of police resources. In this case the user should input the required resulting cardinality and similarity between the initial and resulting rectangular area in which the coordinator plans to allocate the human resources, and the parameter alpha which represents the tradeoff between the required similarity among resulting queries and the accomplishment of the required cardinality.

As a result the program will provide the user with a new rectangular area that will contain a number as close as possible to k required number of historic incidents and that will be as similar as stated by the parameter alpha to the initial rectangular area.

3.2 Application functionality.

In the context of the previously described scenarios, diverVisual application has been constructed to simulate and solve the two first stated scenarios and diverRefinement application has been constructed to do this with the last proposed scenario, the applications have the following described functionality.

3.2.1 DiverVisual functionality.

The diverVisual application objective is to perform the diversification process on a set of records representing cities that are entered by the user, there are two ways to do this: by selecting a country from a dropdown list box containing all the world countries, in this case the cities belonging to the chosen country are used, or by drawing a rectangular area in a provided map, in this case the cities within the rectangular are used for the purpose of diversification.
The user, then must select one of the algorithms to perform the diversification process; max-sum algorithm is likely to show the points in the boundaries of the selected area, and max-min shows evenly distributed points on the chosen area, hence for the first described scenario in the previous section max-min algorithm should be chosen, and for the second mentioned scenario max-sum algorithm should be chosen. The user must also determine the required cardinality of the resulting diversified set, this parameter is given as a percentage of the total number of cities chosen as in the initial set. An additional parameter to enter is the range of population in the cities to be considered when selecting the initial set, this is because the population data of some cities in the downloaded database does not contain complete information about its population, in these cases, a population value of zero has been given to them, so if the user wants to retrieve all cities in a particular area the range should be determined as more than minus one.

Finally, a map with the final results of the process will be displayed, in this map a set of blue circles represents the initial set of cities to be diversified, and on top of it a set of red points represents the set of diversified cities with cardinality equals to a percentage of the first set cardinality.
3.2.2 RefineVisual functionality.

The application objective is to modify an initial query entered by the user and present as a result a refined query that satisfies some entered cardinality and similarity constraints, this is performed using SAQR (Similarity Aware Query Refinement). The process will be performed to solve the third proposed scenario in the application scenarios section.

The demo process will be performed over a set of incidents registered in San Diego City, the cases will be plotted as points and overlaid onto the city map, then it will be displayed a graphical tool that allows the user to draw the initial query as a rectangular area, this rectangular...
area represents the initial query to be refined, after this the user must enter the required cardinality and required similarity.

Finally the system will present as a result the refined query as another rectangular area containing the required set of records.

3.3 System Architecture.

There are various standards and specified paradigms to describe the architecture of a web application, they propose a way to explain the organization of the components that are going to be used. Most of the approaches coincide in dividing the used dynamic components, static components and processes of a web application in presentation layer, business layer, data access
layer and data stores showing the interactions among them.

Below is the high-level architecture diagram of the platform, based in the software components used to develop it. The main purpose of the diagram is to show a concise graphical description of the software technologies used in the construction of the software platform.

![Diagram of the platform's architecture](image)

**Figure 9: System Architecture graphic representation.**

As presented, the platform displays a mash-up of technologies in a web application that combines functionalities from multiple sources, creating a new hybrid web application with functionality unavailable in the original individual application libraries and APIs. Mash-ups construction is an emerging branch in web developing that uses public API’s for accessing data that has been used traditionally only inside the original applications.

Two sample applications has been constructed using this platform, the first: diverVisual allows the visualization of diversification techniques such as max-min and max-sum dispersion, the second one named as Orange (Objective aware range query refinement) or refineVisual allows the visualization of SAQR-S query refinement technique.

The **presentation layer** composed by the *user interface components* and the *user interface process components* is structured as follows. The user interface components used to construct the HTML code to be parsed by the browser are a set of JavaScript libraries: ajax, jquery, json, and d3, a front end framework containing html and css based templates called bootstrap, and the google maps application program interface. Here is important to highlight the relevance of the d3.js library integration which is used to plot data structures on top of the maps, d3 is a remarkable powerful tool when it comes to data visualization over web applications, another important feature is the
employment of google maps drawing manager contained in the google maps API, specifically in selecting range queries as rectangular shapes on a real map provided by an external service in the cloud: the google mapping service.

The user interface process components are separated in diverVisual and refineVisual (Orange) jsp pages and resources. As mentioned earlier, two separately applications have been built over the same software platform.

The previous layer sends http requests and receive the required data as json files and map overlays from the bussines (process) layer which is composed by the process components that are developed java libraries containing the processing logic of diversification and refinement techniques; the workflow components are aided with spring library annotations, and java implemented services to call the calculation and data components; additionally, the service interfaces contain a tomcat server to resolve the obtained jsp pages, and an external service that makes the application dependent on internet: the Google Mapping Service, hence this application cannot run without connection to the internet being dependent on google maps service.

Finally, the process layer uses diverVisual, diverRefinement and Java MySQL jdbc libraries organized into the data access layer to obtain data from the data sources that similarly to the java libraries are divided in two databases one for each diverVisual and diverRefinement storage.

Regarding the scope of this work and the architecture of the system, it has been considered important to deepen in the specific implementation details of the below listed topics:

- User interface.
- Google Maps Service and API and D3 library.
- Data diversification and Query refinement libraries.
- Database.

3.3.1 User Interface.

The user interface facilitates the interactive data exploration process using visual representations of both data and queries, it also coordinates the interactions among the different components of the application. Both interfaces diverVisual and diverRefinement are constructed for web presentation so they can be parsed by a browser. In both cases jsp pages has been created to interact with the user collecting the necessary inputs and returning the required outputs.

For better comprehension of the structure, we will divide the composition of the mentioned jsp pages in:
3.3.1.1 **HTML and CSS structure.**

Web pages are based in HTML and CSS that along with JavaScript is all that a browser can recognize to be parsed, while constructing the web page one option is to write the necessary HTML and CSS to compose the required front end of the application from scratch, the other option is using certain libraries and components that will speed up this duty and make the resulting code much more effective specially for the system to run in different browsers and devices.

When building a web project from scratch it is not always the case that there is enough resources to employ the professional service of a designer, therefore, the developer will need to create the visual interface of the page, which is not only a time consuming process but also a task of professionals in the designing field, consequently the resulting web interface could be visually unsatisfactory and more importantly it could malfunction in different screens, browsers and devices. In this context is sensible to consider that most websites are made of the same basic parts including navigation and layout, these basic parts have been constructed for other applications by some experimented designers including community contributions, this code is available under free licenses to be used in new applications, here is where a framework like bootstrap will help to structure the page to work well in different devices and have an esthetically acceptable web page look and feel.

Bootstrap simplifies the usage of HTML and CSS, by providing the application with standardized concepts, patterns and templates to be used in the interface structure such as navigation controls, buttons, fonts, forms, etc, distributed in a grid system structured in columns and rows. The bootstrap library is included in the jsp front end pages and has been used to format the application presentation, the library has been downloaded to work locally in the server and it is stored in the resources folder of the web application (webapp/resources) and called to the web page by:

```html
<link href="<c:url value="/resources/bootstrap/css/bootstrap.min.css"/>" rel="stylesheet" type="text/css" />
```

From here the bootstrap library, has been used to divide the graphic interface in rows and columns in which the required forms, maps and figures are allocated, it provides homogeneous fonts, buttons and form designs, it also allows the application to work in different screen sizes, this is very important considering the nowadays tendency to use mobile devices.
3.3.1.2 Form interaction for displaying.

The process of diversification and refinement takes place in the server side of the application where a set of java developed process components performs the required calculations, in order to execute this operations the user should input a set of parameters from the user interface, then these parameters are sent to the server to be used as java variables.

In order to perform this task jsp and spring tag libraries are included in conjunction inside the jsp interface page file, this code then passes through the translation process in tomcat server converting it into pure java files, this is accomplished by the use of certain tag libraries that facilitates the coding, in this specific project the next tag libraries have been used:

```html
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://www.springframework.org/tags" prefix="s"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
```

These tag libraries will help associating and passing each value in the interface form to a java class or most commonly called Bean, in the server side a java class representing the form bean is instantiated, this java class includes all the variables required to make the expected calculations, finally, the results of the calculations will be passed back to the form and other interface components as JSON objects.

In summary, the process will be performed in three phases, the first phase comprises the HTTP request to get the initial jsp page and the subsequent display of the form page. In the second phase when the user inputs some values into the fields and submit the form there will be performed the data binding or conversion in which the strings written as jsp and spring tags will be converted and allocated into the different data types of the java class, in this phase validation is also performed, finally the resulting values are sent back to the interface in JSON format.

The interface also contains the google maps API library that allows the displaying of maps and overlays in which the data corresponding to the diversified or refined methods are plotted with the aid of d3.js library

```html
<!— Google Maps API library -->
<script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?libraries=places,drawing&amp;sensor=true&amp;key=*"></script>

<!— d3.js library -->
<script type="text/javascript" src="c:url value="/resources/d3/d3.min.js" />"></script>
```

One of the ways of entering data through the interface is the use of the google maps drawing library which allows constructing various shapes over a map, providing the necessary
methods to obtain data regarding those shapes, for instance when drawing the rectangular areas representing queries, google maps drawing library is used, from the drawn rectangle the corresponding corners latitudes and longitudes are obtained as data to be entered into fields of the form and subsequently passed into the application java components for processing.

3.3.1.3 Scripting.

JavaScript is used with the purpose of coordinating the different interactions between the user and the interface, in this section it will be briefly mention some of the technologies used to make these interactions happen.

For instance, when a user draws a rectangular area, the action of entering the resulting rectangle latitude and longitude corner values in the corresponding html input fields are handled with the aid of JQuery and Google Drawing Manager events.

```html
<!-- JQuery library -->
<script type="text/javascript" src="c:url value="/resources/jquery/1.10.2/jquery.js" /"></script>
```

The data records to be plotted as data points in the map are passed to the interface as JSON objects these are transformed into SVG figures that are handled by d3 library and then plotted onto google map overlays, the data will be requested to the server performing AJAX calls which allows to update only the map division, this last step objective is for avoiding the recharge of the entire page unnecessarily and only re-plot the obtained points in the map.

Finally, the user and interface events, such as pressing buttons, clicking on the map or finishing the drawing of a picture is handled by triggered JavaScripts functions aided by JQuery commands.

3.3.2 Google Maps Service and D3.js library.

In this section it is more extensively explained how google maps service and d3.js library are used together to provide the map display component of the user interface providing natural and convenient representation of data, following is a section dedicated to these two libraries, differentiating them from the others, and giving them major relevance, because the resulting application centers its attention in spatial context as maps and data driven visualized documents, given this, the key libraries and services to perform the task of visualization for spatial data exploration are the considered in this section.

3.3.2.1 Google Maps Service.

In the construction of the diverVisual and refineVisual applications the google maps service and APIs are used, and within it the drawing manager module. Google maps service
consists in a series of APIs and technologies provided by Google, offering maps that can be displayed in 3 modes satellite imagery, street maps and street view perspective, in the present application only map and satellite views will be used for the data displaying.

Via Google Maps API it is possible to embed maps in an application, this will require to send the request through the API to the google server, this petition should contain the required map center specified in geographical latitudes and longitudes and the required zoom for it to be displayed, then google maps service will return the wanted map to the interface, in this case to an HTML div in which the first overlay containing the map will be displayed, this makes diverVisual and refineVisual applications dependent on connection to the internet, this is because the database of imagery and map data is in the servers of Google and cannot be downloaded completely freely to use, this database is periodically updated for users convenience.

When a user drags the map the triggered events will automatically retrieve from the service the new maps to be presented simulating the movement of the user around the world map, as well as the zoom functionality, so this functionality came within the application. After the map is drawn in the signaled spot of the interface, the drawing manager functionality will be loaded to be used, this module allows the user to draw shapes in another overlay placed into the map. For the current application, only one shape will be showed in the tool bar: the rectangular shape, the user can click on this shape to start drawing the required rectangle representing a user query to be diversified or refined according to the application module in use. Once the shape is drawn the ‘rectanglecomplete’ event will be triggered, when the events occur a JavaScript function is executed to obtain the rectangles north east and south west corners latitudes and longitudes and set them into the specified input fields to be passed to the form bean to construct the query necessary to perform diversification or refinement as required.

The previously described process is executed in the client side, performing requests to the google maps service, it does not required any libraries to be stored in the application folders and moreover those cannot be downloaded to be used locally in the client, it is necessary for the application to be connected to the google maps service on internet.

The google maps API could be used to place markers to represent the records retrieved after diversification and refinement process but the mentioned markers does not allow the versatility of information displaying than the d3.js library allows, that is why in the next step this library is used in conjunction with another google maps overlay to plot the required data points over the recently drawn map.

“Copyright: The Google Maps terms and conditions state that usage of material from Google Maps is regulated by Google Terms of Service and some additional restrictions. Google has either purchased local map data from established companies, or has entered into lease agreements to use copyrighted map data’’ [28].
3.3.2.2 D3.js Library

The d3.js library is based on java Script, and used in the client to selectively bind input data to arbitrary document elements, applying dynamic transformations to generate and modify the content in a web page, conveniently d3.js library can be used in conjunction with google maps API.

The points which are the data records that contain the information of its associated longitude and latitude, representing a city in case of diverVisual and an incident in the case of refineVisual has been plotted using d3.js library, additionally in refineVisual (Orange) application a bar chart with the relative efficiency of the refinement techniques is plotted with the aid of d3js tool as well. Now, the data points could be plotted over the maps only using google maps markers, this would not result in the desired visualization given that google markers are based on icons from files that cannot easily be dynamically modified as needed, on the other hand SVG forms used by d3.js library can be modified and parameterized with numerical information obtained from other dimensions of the record, moreover d3.js library will allow to represent more complex sets of data in many different useful statistic graphics built within the library.

D3.js is a powerful tool that allows to draw beautiful and representative visualizations of data, these visualizations can be easily overlaid into a ‘real map’ from google maps API, d3.js allows various visualizations for most complex sets of data that could be used in later works over the present platform.

![Figure 10: Cloropleth, taken from https://github.com/mbostock/d3/wiki/Gallery.](image-url)
For example, the above figure has been drawn using D3.js from a tsv (tab separated values) file containing the code of the states in the country and a numeric value for the unemployment feature represented by the next cut sample of the tsv file.

unemployment.tsv sample:

```
id      rate
1001    .097
1003    .091
1005    .134
1007    .121
1009    .099
1011    .164
1013    .167
1015    .108
1017    .186
```

In the unemployment.tsv the column id is related to the state unique key and the rate value is the rate of unemployment in the given state.

The code of the geographical polygon area is bound with its respective state code, and the rate of unemployment is bound with a quantize scale in which different rgb tones of blue are assigned to ranges of values, this is accomplished in no more than 60 lines of code, from here it can be inferred the utility of using D3.js library when visualizing data that comes from relational data bases.

Going back to diverVisual and diverRefinement applications, SVG circles are constructed and assigned its color property depending of the set from where they proceed, in the case of diverVisual application if they proceed from the set of all the chosen cities to be diversified, blue is assigned as the color of the shapes, and if they proceed from the set of diversified cities red is assigned, these two sets are calculated with java methods that take the data from the database, making the required calculations and passes json objects containing the required data to the interface, this json objects contain the latitudes and longitudes of the points to be plotted as SVG circles. The complexity of the used data structure is minimum, but the powerful capabilities of plotting data with d3.js library could be shown later as the data complexity increases.

The entire d3.js library is stored in the local server, it doesn’t function connected to the d3.js site on the internet, it is located on the folder resources under webapp. To plot the data points an ajax call to the server is made with the required input data to perform the tasks of diversification or refinement, when done, a json object is passed to the client which is parsed with d3.js library, for each data point a svg circle will be created, then these ones are coupled to the data and then they are transformed and translated into their respective positions in the map.
3.3.3 Data diversification and query refinement libraries.

These libraries form the core component of each application, in which processing of the diversification specified methods and query refinement SAQR-S method are implemented. They interact with the user interface, through which they receive their required inputs from the user and returns the corresponding results of diversification and refinement respectively.

The following described libraries consist of a series of classes developed in java, which are activated using the mvc (model view controller) paradigm and spring framework to facilitate the interactions between these components through spring annotations.

The structure and interaction of classes are explained in this chapter with the visual aid of summarized class models of the software systems, in these specific representations the classes are directly related and generated from the source code.

3.3.3.1 Data diversification libraries.

The objective of search result diversification from a user query is to return a list of diverse relevant results. Let \( X = \{x_1, x_2, ..., x_m\} \) be a set of results for some user query. In general, the goal of result diversification is to select a subset \( S^* \) of \( X \) with \(|S^*| = k, k < m\), such that the diversity of the results in \( S^* \) is maximized [2].

According to the diversification definition above, the given user query will determine the territory that contains the cities to be diversified, the user query will be constrained by two parameters, the area in which the cities are located and the range of cities population, from this, the retrieved set will be called \( X \): the set of cities to be diversified and the set of resulting diversified cities will be called \( S \), the number of cities to be obtained when the diversification process take place will be called \( k \). Two methods can be used to perform the process of diversification, Max Sum and Max Min.

In this section we assume that the user has input the constraints to construct the necessary query, this will be handled in conjunction by the user interface spring and jsp tag libs and the corresponding bean (diverVisual Bean) which is the binding element between the interface and the process layer.
The diverVisual bean contains all the necessary variables to pass data from the user interface to the process layer. From here, the java virtual machine performs the procedure of diversification by the use of a dedicated java class for this, it acts in conjunction with controller and database classes to perform it.

Below are some of the classes used to develop diverVisual application, an extract of the most important classes that represent the core of diversification process has been chosen, there are classes that has been left apart, such as the ones that performs validations on the inputs or the ones that shows the process progress on the screen, among others.

![DiverVisual Bean](image)

**Figure 11**: DiverVisual bean.
Figure 12: DiverVisual java classes.

As showed in the previous figure, once obtained the parameters to construct the query, the controller class communicates with the data class which retrieves the corresponding records from the database and transforms it into an array of cities, city is a predefined java class containing the
necessary attributes to represent it. Once the array containing the set $X$ of cities has been constructed, the controller passes it to the corresponding diverAlgorithm (max-min or max-sum) according to the user choice, in the diverAlgorithm classes the process of diversification is performed for both algorithms max-min and max-sum the distance between all possible combinations of pairs of cities is calculated and stored into a two dimensional array representing the Similarity Matrix, also the two most distant cities are calculated and placed into the array representing set $S$ as the first two diversified cities.

In order to calculate the third and subsequent points in Max-Sum algorithm, the distance between all points in the already selected diverse subset to the remaining points in set $X$ is calculated, then for each point in the set $X$ the distances to the points in the diverse subset are summed, subsequently, the point with maximum sum of distances is added to the diverse subset.

Respectively, in order to calculate the third and subsequent points in Max-Min algorithm, the distance between all points in the already selected diverse subset to the remaining points in set $X$ is calculated, for each point in the set $X$ the distances to the points in the diverse subset are summed, then the point with minimum sum of distances will be added to the diverse subset. The class then returns back the array containing the set $S$ of $k$ diversified cities, which is then sent to the interface as a JSON file to be displayed as SVG figures into the map.

The algorithms performed here are naïve greedy methods of diversification, the procedures to make them more efficient in terms of computational time have not been applied yet.

3.3.3.2 Query refinement libraries.

As stated before query refinement methods adjust a submitted query, so the result satisfies some requirements that have been pre specified, in this case a constraint on the cardinality $k$ of the result set is specified. After the process, a refined query is obtained this query result set’s cardinality is equal or very similar to $k$, and the resulting query is as similar as possible to the initial query as the achievement of the requirement on the cardinality on $k$ permits it.

In other words, given an initial query and a constraint on the cardinality of the result set, SAQR (Similarity Aware Query Refinement) returns a refined query that tries to minimize the dissimilarity between the initial and refined query, and at the same time tries to return a result set with the required cardinality or to minimize the difference between the required cardinality and the resulting one.

Here we assume that all the values required as inputs has been already entered by the user and bound through the corresponding bean to the interface. The controller class asks the data class to retrieve all the incidents to be considered to construct the refined query, then the data class retrieves the necessary records from the database and places the required information into an array of incidents which is passed to the controller.
The controller then sends the previous array of incidents along with the necessary values to perform the refinement to the model package, which contains classes (developed by the Data and Knowledge Engineering (DKE) research group of the Queensland University and used in this project) that will return coordinates to be passed to the interface and finally drawn as a rectangle representing the refined query.
Figure 14: RefineVisual java classes.
Many other classes intervene in the processes of visualizing query refinement, for instance, the class that calculates and passes the similarity matrix percentage of computation to the interface, but since these classes in spite of being part of the core of the program are more assistance classes, has not been considered either in the diagrams or for explanation in this chapter.

3.3.4 Database

Two databases have been constructed for the applications in this project, both of them are managed by MySQL DBMS.

3.3.4.1 Search result diversification data.

The data used in the first visualization application is the set of countries of the world, with the associated set of cities inside each country, referenced by its country code including longitudes, latitudes, and population for each one.

The data corresponding to the countries was loaded using the ISO 3166 standard list for country codes, which is an internationally established standard for the representation of the code numbers that belong world countries\(^1\).

The data corresponding to the cities of the world with its population, was downloaded from MaxMind which is an industry provider of IP intelligence and online fraud detection tools, in this company web site’s section of open source databases it can be found a service of free downloadable databases under license Creative Commons Attribution-ShareAlike 3.0 Unported License which requires to include the paragraph below when some of this data is used:

“This product includes GeoLite2 data created by MaxMind, available from http://www.maxmind.com”.

The diverVisual database contains two tables, the table country contains 256 records corresponding to the world countries codes and names, and the table cities contains 3,173,958 corresponding to cities names and other information, the data structure is given by the next EER diagram:

---


<table>
<thead>
<tr>
<th>Relation: country</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>code</td>
<td>ISO country codes</td>
</tr>
<tr>
<td>name</td>
<td>Country name in English</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation: city</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>country_code</td>
<td>Reference to the country the city belongs to.</td>
</tr>
<tr>
<td>city_id</td>
<td>Numeric unique value for each city.</td>
</tr>
<tr>
<td>city_ascii</td>
<td>City name in ASCII format.</td>
</tr>
<tr>
<td>city_name</td>
<td>Name of the city.</td>
</tr>
<tr>
<td>region</td>
<td>Geographical region for the city, not referenced.</td>
</tr>
<tr>
<td>population</td>
<td>Number of inhabitants in the city</td>
</tr>
<tr>
<td>latitude</td>
<td>Numerical latitude provided by the geocoder.</td>
</tr>
<tr>
<td>longitude</td>
<td>Numerical longitude provided by the geocoder.</td>
</tr>
</tbody>
</table>

Not all the features are used in the process, the diversification is done based only in the distance between cities which only considers latitudes and longitudes to perform the calculus.

3.3.4.2 Query refinement data.

The data used for the second visualization is a historical dataset of crime incidents of the city of San Diego, CA in USA.

The data has been extracted from clarinova.com-crime-incidents-casnd-7ba4. San Diego Regional, Data Library. 2013-08-07 http://sandiegodata.org. This data is represented in the EER diagram below:
Figure 16: RefineVisual EER.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>incident_id</td>
<td>Unique identifier for the incident.</td>
</tr>
<tr>
<td>incident_date</td>
<td>Date in which the incident occurs.</td>
</tr>
<tr>
<td>latitude</td>
<td>Numerical latitude provided by the geocoder.</td>
</tr>
<tr>
<td>longitude</td>
<td>Numerical longitude provided by the geocoder.</td>
</tr>
</tbody>
</table>

Not all the features are used in the process, the refinement is done based only in a spatial distance measure that determines if the crime is or not into the required rectangular area corresponding to the required query, which only considers latitudes and longitudes to perform the calculus.
Chapter 4

Conclusions

4.1 Summary and conclusion

The present work centers its attention on the visualization of data exploration techniques, specifically in data diversification max-sum and max-min diversification algorithms, and similarity aware query refinement (SAQR).

Three case scenarios has been proposed to showcase the processes of data diversification and query refinement and two sample applications has been built to visually demonstrate the results of this processes: the first application called diverVisual shows a visual solution for the first two scenarios and similarly the second application called refineVisual or Orange presents the visualization of the third scenario processing and solution.

The mentioned applications have been developed over a web platform that combines applications, services and libraries into a mash-up of technologies that allows the interactive visualization of the previously described data exploration techniques, d3.js library, google maps API, Bootstrap framework, jsp pages, JavaScript, JQuery, Ajax, Spring framework, java virtual machine, MySQL DBMS, Tomcat application server, along with other software, make it possible the binding of data records with html elements and its overlaid and display over real maps.

D3.js library and Google Maps APIs and services are more extensively explained, differentiating them from the others and giving them a major relevance given that one of the core components of this project is the visualization of data records in spatial context such as maps and data driven visualized documents, hence the key libraries and services to perform the task of visualization for spatial data exploration are d3.js library specialized in creating useful visualizations of relational data and google maps APIs and service which allows the images overlaid and displaying in an spatial graphic interface that uses real maps.

The platform for visualization of spatial data exploration platform architecture is described along with both diverVisual and diverRefinement application implementation details, additionally
the composition of java classes that perform the processes of diversification and query refinement are expounded along with the description of its methods, finally the database structure and the data loading sources are described and cited.

In conclusion, the proposed scenarios to perform the diversification and query refined tasks are demonstrated in an interactive web platform containing a mash-up of applications, libraries and services.

Firstly, the diverVisual application is developed and implemented, this application diversifies by location a group of selected cities from a world map then returns a set of diversified cities according to some user constraints using max-sum or max-min diversification algorithms, here the diversification technique core libraries has been coded specifically for the present work.

Secondly, the Orange (refineVisual) application is developed and implemented, this application refines an initial query represented by a rectangular area that contains crime information of San Diego city to obtain a refined query according to cardinality and similarity user imposed constraints, to obtain the required resulting query containing the required number of crimes in a very similar rectangular area, here the query refinement methods core libraries has been coupled to the system as black box functions developed by the Data and Knowledge Engineering (DKE) research group of the Queensland University.

Both applications provide users with real maps and allows them to select zones representing range queries and enter additional user constraints and other parameters interactively. Then, by using diversification techniques and a novel SAQR (Similarity Aware Query Refinement) scheme, the application produces clear visualizations of the obtained results. Both applications have been built over the same software platform which proves its effective usage for multiple scenarios.
Appendix A

Companion disk.

The companion disc contains.

- DiverVisual libraries and code. (diverVisual folder)
- RefineVisual libraries and code. (refineVisual folder)
- Sql file to populate and construct diverVisual database. (divervis.sql)
- RefineVisual sql to populate and construct refineVisual database. (incidents.sql)
Bibliography


New York: Apress ;


New York, NY: Apress ;


