12-13-2008

Hierarchical Geographical Identifiers As An Indexing Technique For Geographic Information Retrieval

John Christopher Lakey

HIERARCHICAL GEOGRAPHICAL IDENTIFIERS AS AN INDEXING TECHNIQUE FOR GEOGRAPHIC INFORMATION RETRIEVAL

By

John Christopher Lakey

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2008

HIERARCHICAL GEOGRAPHICAL IDENTIFIERS AS AN INDEXING TECHNIQUE FOR GEOGRAPHIC INFORMATION RETRIEVAL

By

John Christopher Lakey

Approved:

Robert J. Moorhead II
Professor of Electrical and Computer Engineering, Department of Electrical and Computer Engineering
(Major Professor and Director of Thesis)

Tomasz Haupt
Associate Research Professor, Department of Computer Science and Engineering
(Committee Member)

James E Fowler
Professor of Electrical and Computer Engineering, Graduate Coordinator, Department of Electrical and Computer Engineering

Surya S. Durbha
Assistant Research Professor, Department of Electrical and Computer Engineering
(Committee Member)

Sarah Rajala
Dean of the College of Engineering

Name: John C. Lakey

Date of Degree: December 12, 2008

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. Robert Moorhead

Title of Study: HIERARCHICAL GEOGRAPHICAL IDENTIFIERS AS AN
INDEXING TECHNIQUE FOR GEOGRAPHIC INFORMATION
RETRIEVAL

Pages in Study: 39

Candidate for Degree of Master of Science

Location plays an ever increasing role in modern web-based applications. Many of these applications leverage off-the-shelf search engine technology to provide interactive access to large collections of data. Unfortunately, these commodity search engines do not provide special support for location-based indexing and retrieval. Many applications overcome this constraint by applying geographic bounding boxes in conjunction with range queries. We propose an alternative technique based on geographic identifiers and suggest it will yield faster query evaluation and provide higher search precision. Our experiment compared the two approaches by executing thousands of unique queries on a dataset with 1.8 million records. Based on the quantitative results obtained, our technique yielded drastic performance improvements in both query execution time and precision.

ACKNOWLEDGEMENTS

I am deeply indebted to my major professor, Dr. Robert J. Moorhead, for providing me with opportunities, encouragement and guidance.

Many thanks to my patient and loving wife Inessa who has been a great source of strength and encouragement through all this work.

Thanks also to Samuel Espy for his friendship, many hours of discussion, and detailed reviews of drafts of this document.

Finally, I would like to thank my parents, John and Benita Lakey, for being wonderful parents.

TABLE OF CONTENTS

Page

CHAPTER

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Search is a prominent part of the user experience for web and web-based applications [20]. While most search technologies focus on textual content, the web is also rich with location information. With the market for personal navigation devices expected to reach $14 billion by 2010 [6] and with many web sites supporting geo-tagged images and content, the volume of location information on the web will greatly increase. Additionally, text-mining tools capable of extracting location information in the form of place names from existing text content are becoming increasingly common. A recent report commissioned by the National Geospatial Intelligence Agency (NGA) identified the importance of geospatial references in unstructured and semi-structured text documents and noted that text and place name searches are both important areas for research [5]. With the increasing volume and role of location information, techniques must be developed to location enable current search technology.

While much of the current research is focused on developing new location-enabled search engines, our research is motivated by the desire to augment existing search technology with location capabilities. The dominant technique is to index point and bounding box approximations and to support bounding box searches using range queries. This can lead to a significant percentage of false positives and potentially poor search performance.

Herein an alternative approach is introduced using a gazetteer to provide a predefined set of search locations with complex boundaries for indexing and searching. Each document‟s location is approximated by one or more locations in the gazetteer. The document is then indexed using the geographic identifiers of these locations. Likewise, search is performed by supplying the geographic identifier of the desired search location. In a typical usage scenario, an application will provide a user interface for selecting the search location by navigating or searching the gazetteer. The search locations are limited to the contents of the gazetteer, but this is common in many vertical search engines. For instance, the U.S. Geological Survey (USGS) National Biological Information Infrastructure (NBII) program often catalogs biological information using predefined regions and habitats. The U.S. Census Bureau collects and organizes census data using a predefined set of location types including states, counties, divisions, regions and tracts.

We hypothesize that our approach will result in faster search evaluation and yield higher precision when the set of search locations is known in advance. Next, we describe our approach including relevant background material and related work. Experimental results are presented with conclusions and opportunities for future work.

CHAPTER II

BACKGROUND

2.1 Inverted Index

Most text search engines employ the same basic technique, the inverted index, for indexing and searching documents [3].

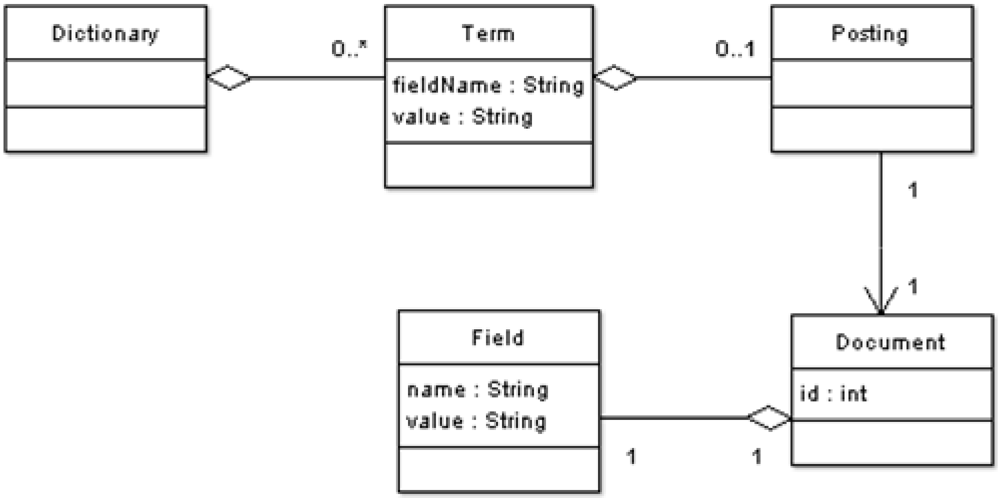

Figure 1 Inverted Index UML Class Diagram

Figure 1 shows the key concepts for this model. Internally the index has a dictionary containing a list of terms, or words, that appear in the indexed documents. Generally the term has an associated field name that makes it possible to search for terms that only appear in the title, the body, or another portion of the document. This list of terms is sorted alphabetically to improve performance when searching for a specific term. For each term, the index maintains the list of postings, where each posting represents a document containing the term. Additionally, the posting contains information, such as the location of the term within the document and the frequency of the term. Since Boolean queries using multiple terms will typically be executed, the postings are sorted by document identifier for quick access. Finally, the document contains a set of fields allowing the association of additional information. The additional information is generally used when presenting search results and includes the document identifier, title and an automatically generated summary.

2.1.1 Searching

Searching the index for a specific term involves finding the term in the dictionary and retrieving the list of matching documents. If the query involves multiple terms, then the search engine must ensure that the particular document is present in the posting list for all specified terms. Most search engines support additional features (i.e. Boolean search operators), with optimizations often separating one search engine from another. However, the basic indexing approach is essentially the same.

Some search engines support the ability to specify the “fieldname” for a term when issuing a query. For instance, Google supports the "fileType" field that contains the extension of the file being indexed. If one is only interested in finding latex files that contain the word "font", then issuing the query "font fileType:tex" will search the index for the regular text term "font" and for the "fileType" term "tex". The ability to search custom fields is a key extensibility feature provided by many of the open-source and commercial search engine libraries used to power the current generation of interactive web applications.

2.1.2 Index Construction

The process of constructing the index involves the four major steps shown in Figure 2 [18].

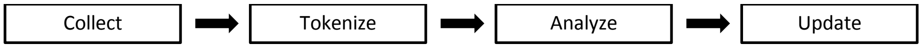

Figure 2 Index Construction Process

1. **Collect** - Collect the documents to be indexed. Most web search engines employ a number of "spiders" that traverse the web and cache local versions of the documents. The spiders are programmed to follow links within HTML documents to create the largest possible collection of documents.
2. **Tokenize** - Documents must be parsed into individual terms before they can be further analyzed. Tokenizers are generally file format specific (e.g., HTML, PDF, etc.) and produce output that is file format independent.
3. **Analyze** - The tokens produced in the previous step are linguistically processed to reduce terms into the root form and to remove simple stop terms such as "a", "and", and "the".
4. **Update** - Finally the resulting terms are used to create postings and update the index. Additional metadata, such as document source, date indexed, and the algorithms applied are typically stored in the index as fields.

2.2 Bounding Boxes

The geographic bounding box, or Minimum Bounding Rectangle (MBR), is “a rectangle, oriented to the x and y axes, which bounds a geographic feature or a geographic dataset. It is specified by two coordinates: xmin, ymin, and xmax, ymax” [19]. While other bounding shapes exist, the bounding box is one of the most frequently used and computationally simple linear bounding shapes [24].

2.2.1 Point and Bounding Box Intersection Search

Indexing point data with an inverted index is realized by separately indexing the latitude and longitude values as fields in the index. Searching the index for points that fall within a bounding box is performed using a query that contains two range clauses. If the point coordinate fields are “lat” and “lon” and the bounding box is expressed as minx, miny, maxx, and maxy, then the general form of the query is as follows:

```
“lat:[minx TO maxx] AND lon:[miny TO maxy]”
```

For example, to find all of the point records that fall within the continuous United States, the query would be:

```
“lat:[-126 TO -65] AND lon:[24 TO 51]."
```

2.2.2 Bounding Box and Bounding Box Intersection Search

When the data being indexed consist of more than simple points, the shape can be approximated using a bounding box. The index will contain the four coordinates of the bounding box: minx, miny, maxx and maxy. The general form of the query required to search against indexed bounding boxes using bounding box requires four range clauses:

```
minx:[minx to *] AND maxx:[* to maxx] AND
miny:[miny to *] AND maxy:[* to maxy]
```

The following query will find all of the bounding boxes within the continuous United States:

```
minx:[-126 to *] AND maxx:[* to -65] AND
miny:[24 to *] AND maxy:[* to 51]
```

Many search engines implement range queries by rewriting the query range clause with primitive queries enumerating terms that appear within the range. For ranges with a large number of values, this yields poor performance and can lead to exceptional conditions caused by limits on the number of clauses allowed in a query. Most search engines now implement techniques for processing range queries that avoid clause limits, but large ranges are still computationally expensive to process.

2.2.3 Bounding Box Issues

While the bounding box is a widely used construct, various issues can cause unexpected problems. Care must be taken when dealing with data that crosses the 180 degree meridian because of the way the globe is artificially split. The bounding box of geometry often changes when a map projection is applied. Furthermore the effectiveness of the approximation can vary greatly, depending on the shape of the original geometry. This is relevant to our study since it can affect search speed and the number of non-relevant results.

The Bounding Box Factor [4] is one measure of the effectiveness of the bounding-box approximation to the original geometry. It is defined as the ratio of the bounding-box area to the area of the original geometry. The minimum value is 1and occurs where the bounding box and original geometry are identical. The maximum value is infinity and occurs when the bounding box is infinitely larger than the original geometry. If the indexed data is evenly distributed geographically, then the ratio of the total number of search results (both relevant and non-relevant) to the number of relevant search results should directly correlate with the Bounding Box Factor.

2.3 Gazetteer

A gazetteer is a geographic dictionary or index [12]. Most printed atlases contain a gazetteer at the back which provides a list of place names with pages and map coordinates where each place can be found. Essentially it is an inverted index for place names in the atlas.

Various online gazetteers are available. The Geographic Names Information System (GNIS), developed by the U.S. Geological Survey for the U.S. Board of Geographic Names, is the Federal standard for geographic nomenclature and the official repository of domestic geographic names data. This repository contains the federally recognized name of each feature and defines the feature location by state, county, USGS topographic map, and geographic coordinates [26]. The Yahoo Internet Location Platform provides a service for "managing all geo-permanent named places on Earth" [28]. The service assigns each geographic entity a unique 32 bit identifier called a

WOEID (Where On Earth Identifier). The system also maintains the parent, children, and neighbors for each geographic entity.

As part of its mission to develop standards for "information concerning objects or phenomena that are directly or indirectly associated with a location relative to the Earth" [15], ISO Technical Committee 211 (TC211) developed the ISO 19112:2003 standard: "Spatial Referencing by Geographic Identifiers" [16]. Figure 3 shows a simplified subset of the ISO 19112 Gazetteer model. The SI_Gazetteer object has a name and is comprised of a set of SI_LocationInstances. Additionally, the SI_Gazetteer references a set of Location types (SI_LocationType) that are supported by the Gazetteer. Each SI_LocationInstance represents a real-world location and is assigned a unique geographic identifier. The location has a representative position specified by a geographic point and a geographic extent. The extent can be a bounding box, geometry, or an identifier. The SI_LocationType object forms a type system for SI_LocationInstance objects and supports nesting through parent and child references.

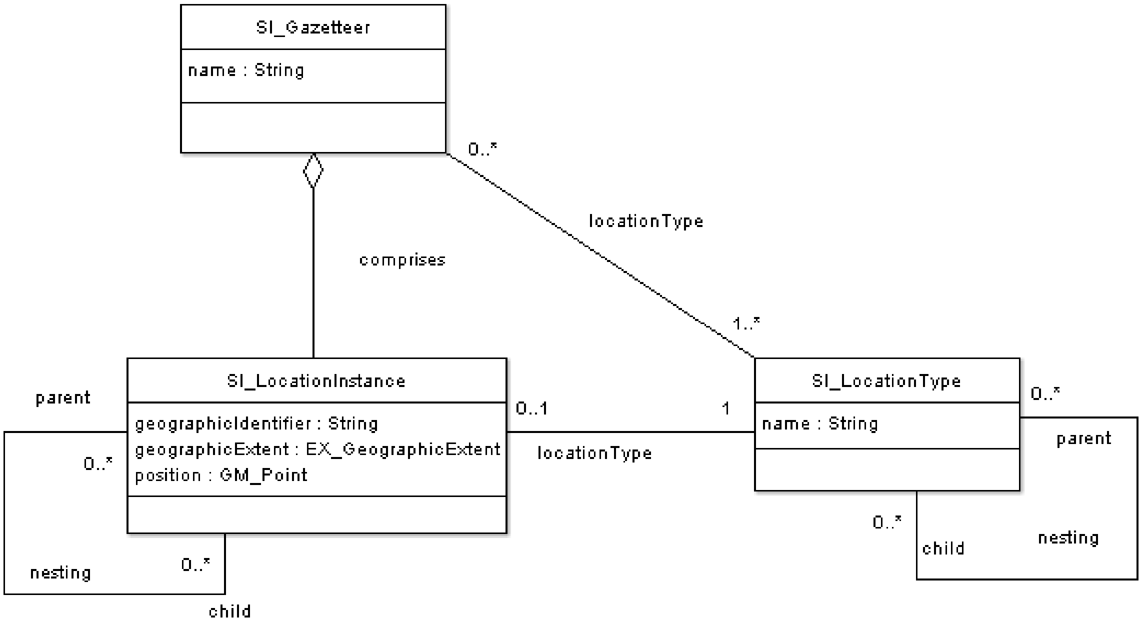

Figure 3 Simplified ISO 19112 UML Model

CHAPTER III

RELATED WORK

Much of the current work in the field of Geographic Information Retrieval (GIR) [17] is related to developing hybrid indexing techniques that combine the inverted index with a spatial index, typically some derivative of the R-tree. The R-tree is a balanced hierarchical structure, similar to a B-tree, except instead of organizing nodes based on a total ordering of the keys, R-tree organizes rectangles according to a containment relationship [22]. A good overview of various hybrid techniques can be found in a technical report [27] describing the hybrid indexing approach for the SPIRIT project. Similarly, work by Zou et al. [30] compares various hybrid indexing techniques based on inverted files and R*-trees. Both approaches employ bounding boxes and therefore will have similar error characteristics to the range-query approach.

Yang et al. [29] describe their use of the Lucene search engine library with range queries to perform spatial and temporal queries against earth science metadata. Since metadata often contain only bounding box extents, precision issues were not a focus of their work. Their test corpus included less than 3000 records and response time was approximately one second. This time likely included the xml parsing and formatting time of their application and did not reflect the true time required to evaluate the query. Their conclusions express concern over capacity limitation of Lucene and indicated that they would pursue an alternate implementation for future work. Lucene version 1.9 introduced the ConstantScoreRangeQuery which supports an unlimited number of terms in the range and resolved the capacity issues associated with range queries.

The PANGEA Framework for Metadata Portals (panFMP) is a metadata search engine built using Lucene. In a recent paper [23], they describe an extension to Lucene that provides a trie (prefix tree) based algorithm for range queries over numeric and data types. They state that search time for range queries is no longer dependent on index size. This is achieved by redundantly storing numerical terms in different precisions. This approach sounds promising, though the additional storage requirements and complexity could lead to scaling issues for indexes with millions or billions of entries.

In addition to the data-driven techniques discussed so far, where the indexing structure is organized by the data being indexed, space-driven techniques have also been advocated [22]. The quadtree [11] is a popular tree-based structure where each node has four children. If each node is assigned a number 1 through 4, then it is possible to compose a string that describes that path from the root to any given node in the tree. This string can be used as an identifier for the spatial extent occupied by the node. The C-Squares specification [21] defines a similar space-driven gridding scheme compatible with World Meteorological Organization (WMO) squares. While this approach has some unique characteristics, implementation for anything but point data is non-trivial since determining the cell covered by a given shape requires rasterizing the shape onto the grid structure.

CHAPTER IV

APPROACH

The approach we take for implementing geographic information retrieval is to assign each document one or more geographic identifiers from a reference gazetteer; then, using standard inverted index techniques, index the identifiers; and search for these identifiers. We implemented a gazetteer based on the ISO 19112 model using Java and open-source technologies. The gazetteer was populated with boundary data from the US Census Bureau. The Apache Solr search server provided the low-level search engine functionality, and the U.S. dataset from Geonames.org served as the document corpus for indexing. Figure 4 provides an overview of the system implemented.

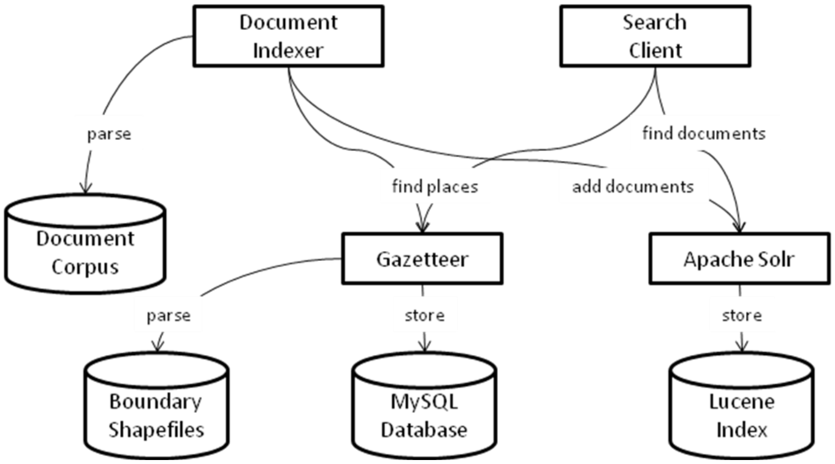

Figure 4 System Overview

4.1 Gazetteer Design

We implemented our gazetteer in Java by mapping the ISO 19112 model to a set of simple Java classes. There is currently not a standard Java API for representing geospatial geometries; however, the open source Java Topology Suite (JTS) is widely adopted and supports all of the concrete geometry types defined by the Open Geospatial Consortium (OGC) Simple Features Specification (SFS). These include: Point, LineString, Line, LinearRing, Polygon, GeometryCollection, MultiLineString, MultiPolygon, and MultiPoint.

The Java objects are persisted to a relational database using the Hibernate implementation of the standard Java Persistence Architecture (JPA) framework. The Hibernate Spatial framework [14] extends Hibernate with mappings from the JTS Geometry object to the native geometry type of the relational database. Table 1 provides a listing of the properties for the Location Instance class with both the Java and SQL type for each property.

Table 1 ISO 19112 Location Instance Object-Relational Mapping

Field	Java Type	SQL Type	Description
geographicalIdentifier	String	varchar(255)	Unique geographic identifier
id	Integer	serial not null	Primary key
parent	LocationInstance	int4 (foreign key)	Reference to parent
title	String	varchar(255)	Human readable display title
geometryExtent	(JTS) Geometry	geometry	Polygon or MultiGeometry
name	String	varchar(255)	Official place name

4.1.1 MySQL

MySQL 5.1 was selected as the relational database implementation. This decision was based on the widespread adoption of MySQL and its native support for a Geometry type with spatial indexing based on the R-Tree structure. Unfortunately, it was discovered during testing that MySQL does not properly check for polygon intersections when performing spatial queries. Specifically, when performing an overlaps test against a polygon stored in the database, it only compares the bounding box of the stored geometry and not the actual polygon. This is likely an optimization to avoid retrieving the actual geometry when evaluating the query. The work around to this problem was to perform a second pass filter on the query results using the actual retrieved geometry. Fortunately the JTS geometry API provides a rich set of spatial operators.

4.1.2 Geographic Identifier

We have chosen to represent the geographic identifier as a string of up to 255 characters. This provides the flexibility to support multiple naming schemes with the same implementation. Since the data in our gazetteer was based on state and county features loaded from census boundary files, we chose to implement an identifier scheme based on the Federal Information Processing Standards (FIPS) codes [9,10]. For state features, the identifier is the FIPS code for that state. For county features, the identifier is the FIPS code for the state and the FIPS code for the county separated by a „,“. The Location instance also contains an integer ID field that is used internally as a system generated primary key for improved performance. Table 2 provides examples of both state and county location instances with their associated identifiers.

Table 2 Example Location Instances

Location	ID	geographicIdentifier
State of Mississippi	48	28
Oktibbeha County	2631	28.105

4.1.3 Census Boundary Data

The U.S. Census Bureau boundary files for states and counties were used as data for the gazetteer. This data was downloaded from Census Bureau‟s Cartographic Boundary Files website [25] in ESRI Shapefile format [8]. This data is loaded automatically by the gazetteer when it is first initialized.

4.2 Index Design

The inverted index implementation is based on Apache Solr [2]. Apache Solr is a popular open-source search server with an HTTP API. Solr internally uses the high-performance Apache Lucene text search engine library and provides a number of additional features including support for a data schema, including numeric types, dynamic fields, and unique keys. The support for numeric types is employed to store coordinate data and perform bounding box queries against those coordinates. Documents are added to the index via HTTP by submitting an XML record containing the fields of the document to be indexed. Search is performed by submitting queries to the Solr server and the matching documents are returned as an XML result set. For our implementation, we have configured an instance of Solr with a document schema that includes fields to support searching by both bounding box and geographic identifier.

While text-based search engines work with lexically ordered character strings, techniques have been developed to support simple data types through special formatting. For instance, by zero padding the string representation of an integer, numerical range queries can be performed using text-based range queries supported by many search engines. These techniques require that values in the data being indexed and the values in the query string are formatted in a special way. Using similar techniques, Solr provides a real data schema with support for numerical types, dates, and unique identifiers. In order to exploit these capabilities, Solr must be configured with a schema that defines how specific document fields are treated. Table 3 provides an example schema; Table 4 describes the key attributes of the field definitions within the schema.

The location field contains the set of geographic identifiers associated with the document. This field has type string, which, unlike the text type, is not further analyzed by the search engine before indexing and storing. The location field is also multi-valued, making it possible to have multiple locations associated with a single document. The latitude and longitude fields are type float, which means they are analyzed as floating point numbers and are mapped into a lexical form that supports range queries. Both latitude and longitude are single values, meaning that a document only has one point location.

Table 3 Sample Schema with Spatial Indexing Support

Name	Type	Indexed	Stored	multiValued	Description
id	string	true	true	false	The id of the document being indexed. The id field is also specified to be the unique key for the document.
title	string	true	true	false	The name of the document being indexed. Useful for display
latitude	float	true	true	false	Representative latitude of the document. Useful for map display
longitude	float	true	true	false	Representative longitude of document. Useful for map display
location	string	true	true	true	The list of locations (geographic identifiers) associated with the document.
text	text	true	false	true	Catchall field containing text to be searched.

Table 4 Key Attributes for Field Definitions

Attribute	Description
name	The name of the field
type	The name of a previously defined type from the <types> section
indexed	True if this field should be indexed (searchable or sortable)
stored	True if this field should be retrievable
multiValued	True if this field may contain multiple values per document

4.3 Constructing the Index

The data being indexed in this experiment is the U.S. places dataset from GeoNames [13]. This data consist of 1,886,123 places of interest within the United States. While this data is often used to populate a gazetteer, here the data is being used differently. Each record in the dataset represents a document to be indexed, and the location of the document is the location described by the record. The data is distributed as a single text file with one record per line. The general process for indexing the dataset is presented as the following pseudo code.

```
foreach (Record record: records) {
    Document doc = computeDocument(record);
    submitDocument(doc);
}
```

Adding a new document to Solr or updating an existing document requires posting an XML document summary to the server containing the documents fields. Solr analyzes each field based on the schema and updates the index with the new values. The example in Figure 5 adds a document summary for "Mississippi Research and Technology Park".

```
<?xml version="1.0" encoding="UTF-8"?>
<add>
    <doc>
        <str name="name">Mississippi Research and
Technology Park</str>
        <str name="id">4436279</str>
        <arr name="location">
            <str>US.28</str>
            <str>US.28.105</str>
        </arr>
        <float name="latitude">33.469566</float>
        <float name="longitude">-88.790886</float>
        <float name="elevation">100.0</float>
    </doc>
</add>
```

Figure 5 Sample Solr Request

With the exception of the "location" field in the previous example, all field values were taken directly from the GeoNames dataset. Determining the location values requires querying the gazetteer to find all Location instances that overlap the geometry of the document being indexed. The GeoNames dataset only contains point geometries, although our approach supports point, line, bounding box, polygon and multiple geometry representations. For each intersecting Location instance, the string identifier of the location is added to the location field of the document.

4.4 Searching the Index

In our approach, searching for documents associated with a geographic identifier is directly supported by the Solr search engine using the standard query language. For instance, to find all of the documents associated with US.28.105 (Oktibbeha County, Mississippi) that contain the word "research" in their name, the query string “name: research AND location:US.28.105” is submitted to the Solr search server.

Searching for documents within a bounding box requires constructing a slightly more complex query. The Solr query syntax in based on the syntax implemented by the Lucene Query Parser and supports Boolean operators (AND, OR, NOT), the ability to query specific fields, as well as wildcards, prefix queries and range queries. Bounding box queries are constructed as a conjunction (AND) of two range queries along with any other query terms [1]. Consider the following example that uses range queries to find all documents with a latitude and longitude that fall within the bounding box for Oktibbeha County and contain the word "research" in their name: “name: research AND longitude:[-89.0087 TO -88.6691] AND latitude:[33.2859 TO 33.5656]”.

The response to a Solr query is returned as an XML document containing metadata about the request and the fields of the matching documents. Figure 6 contains a response with a single result.

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <lst name="responseHeader">
        <int name="status">0</int>
        <int name="QTime">1</int>
        <lst name="params">
            <str name="indent">on</str>
            <str name="start">0</str>
            <str name="q">name:research AND
                          location:US.28.105</str>
            <str name="rows">10</str>
            <str name="version">2.2</str>
        </lst>
    </lst>
    <result name="response" numFound="1" start="0">
        <doc>
            <str name="name">Mississippi Research and
                             Technology Park</str>
            <str name="id">4436279</str>
            <arr name="location">
                <str>28</str>
                <str>28.105</str>
            </arr>
            <float name="latitude">33.469566</float>
            <float name="longitude">-88.790886</float>
            <float name="elevation">100.0</float>
         </doc>
    </result>
</response>
```

Figure 6   Sample Solr Response

4.5 Experiment Design

The experiment was designed to support the comparison of two techniques for integrating spatial constraints into a standard search engine. The first approach uses a gazetteer to assign a geographic location to each document being indexed by the search engine. Each document in the search corpus has geographical coordinates (latitude, longitude) and is assigned a set of location identifiers using the gazetteer. The second approach searches directly against the latitude and longitude coordinates. The key fields from the document, including the latitude; longitude; and newly assigned location identifiers, are added to the search engine.

Once all documents have been indexed, a series of searches is performed and the response times are recorded. Two searches are performed for each county in the continuous United States. First, the index is queried by the geographic identifier of the county and then by the bounding box of the county. The response times and the size of the result set are recorded. This process is performed for all counties. To negate the effects of network delay in the measurements, the response time recorded is the query processing time reported in the Solr response. Additionally, zero records are returned, thus limiting the processing time to determining which records should be returned but not actually retrieving any data. In an attempt to minimize the timing effects of "lucky" cache hits and random pauses caused by garbage collection, this process is repeated ten times with the highest and lowest timing values for each county discarded and the resulting eight values averaged to arrive at final timing values. Finally, the results are written to a spreadsheet for further analysis. The schema for the spreadsheet is described in Table 5.

The software for executing queries and gathering results was implemented as a standalone Java program. All components of the experiment were installed on the same computer to avoid the effects of network delay. Table 6 provides a detailed listing of the execution environment for the experiment.

Table 5 Schema for Results Data Spreadsheet

Column	Description
geographicIdentifier	Geographic identifier of the search location
locationCount	Number of search results returned when searching by identifier
locationTime	Execution time for the search by location
bboxCount	Number of search results returned when searching by the bounding box of the location
bboxTime	Execution time for the search by bounding box
locationArea	Area of the location computed using the polygonal boundary representation
bboxArea	Area of the location computed using the bounding box representing

Table 6 Experiment Execution Environment

Component	**Description**
Hardware	HP Pavilion a6110n PC (Processor: AMD Athlon 64 X2 4400+ (2.3 GHz); RAM installed: 2 GB DDR II SDRAM; Hard drive: 320 GB Standard
Operating System	Microsoft Windows Vista 32-bit
Java	Java 6 SE Update 5
Database	MySQL 5.1.5
Search Engine	Apache Solr 1.2.0
Third Party Java Dependencies	Hibernate 3.3.0, Hibernate Spatial 1.0.M1, Java Topology Suite (JTS) 1.9

4.6 Data Sets

For this experiment, the gazetteer was configured with 3,271 state and county locations and the document corpus consisted of 1,886,123 places of interest. The specific details of each data set are provided in Table 7. Table 8 provides summary statistics for the GeoNames dataset and Figure 7 provides a density map of the locations in the GeoNames dataset. While the maximum density is over 250k places/square degree, the color scale is capped at 10k places to provide detail in the lower range of the scale. The New York, District of Columbia, and San Francisco areas comprise the majority of the 15 counties exceeding 50k places/square degree.

Table 7 Data Sets Used in the Experiment

Dataset	**Source**	Version / Date	**Description**
Census 2000 County and County Equivalent Areas Cartographic Boundary Files	U.S. Census Bureau	May 08, 2001	ESRI Shapefile containing U.S. counties and equivalent areas from the 2000 census
Census 2000 State and State Equivalent Areas Cartographic Boundary Files	U.S. Census Bureau	May 08, 2001	ESRI Shapefile containing U.S. states and equivalent areas from the 2000 census
GeoNames U.S. Database	Geonames.org	January 05, 2008	A daily export of the GeoNames database

Table 8 GeoNames Dataset Statistics

Statistic	**Value**
Total Places	1,886,123
Minimum Density	120 places / deg^2
Maximum Density	266,887 places / deg^2
Mean Density	3,885 places / deg^2
Median Density	2,371 places / deg^2
Standard Deviation	9,556

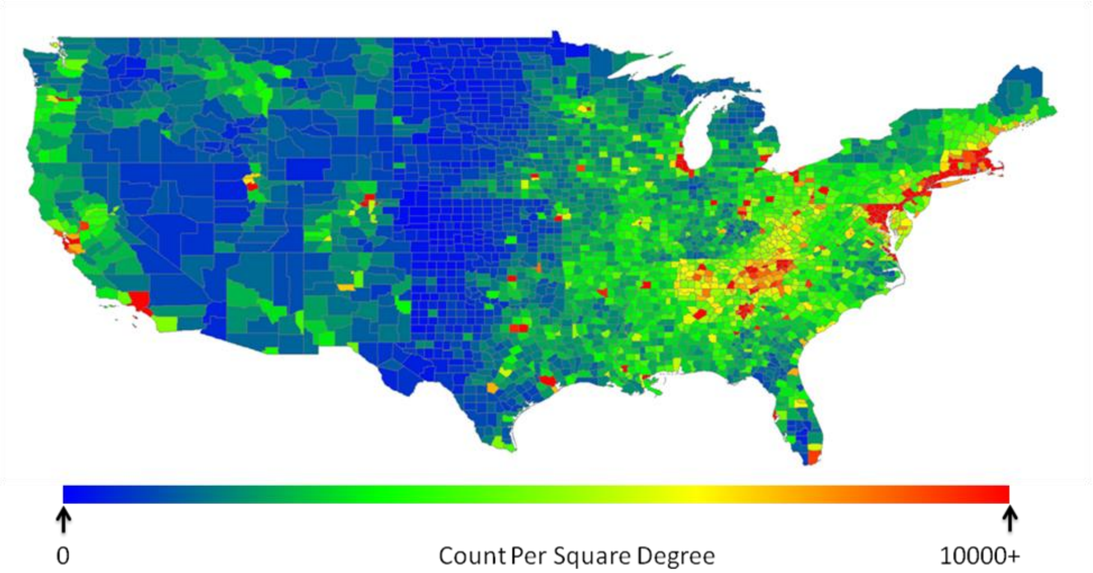

Figure 7 GeoNames Dataset Density Map

CHAPTER V

EXPERIMENTAL RESULTS

The results of the experiment described in chapter IV are presented below.

5.1 Bounding Box Factor

The Bounding Box Factor (BBF) is the ratio of the area of the bounding box approximation for a feature to the actual area of the feature. This was computed for each county in the continuous United States. Summary statistics are presented in Table 9 and a map-based visualization of the BBF for each county is displayed in Figure 8. These numbers are somewhat lower than values provided by [4]. The differences are likely explained by our use of unprojected data versus Caldwell"s use of the Albers Equal Area projection. Additionally, our experiment was limited to the continuous United States and did not include Alaska, Hawaii and various political entities external to the continuous United States.

Table 9 Bounding Box Factor Statistics

Statistic	**Our Results**	**Caldwell's Results**
Minimum	1.00151	1.003001
Maximum	8.39002	42.077043
Mean	1.49599	1.609442
Standard Deviation	0.48215	0.897699

5.2 Search Precision

The Non-Relevance Factor (NRF), an inverse measure of precision, is the ratio of the number of search results returned with the bounding box query to the number of results contained within the actual boundary of the county. NRF values were computed using each county‟s bounding box as the search area. Figure 9 provides a map-based visualization of the NRF for each county. Table 10 contains NRF summary statistics with our BBF data repeated for comparison. The values for the two factors are very similar, indicating the BBF is a strong determining factor in the NRF. The chart in Figure 10 plots the NRF (Y axis) against the BBF (X axis) and the linear trend line further indicates the correlation between the NRF and the BBF.

Table 10 Non-Relevance Factor Summary Statistics

Statistic	**NRF**	**BBF**
Minimum	1.0	1.00151
Maximum	6.76126	8.39002
Mean	1.47543	1.49599
Median	1.35538	1.4093
Standard	0.4918	0.48215

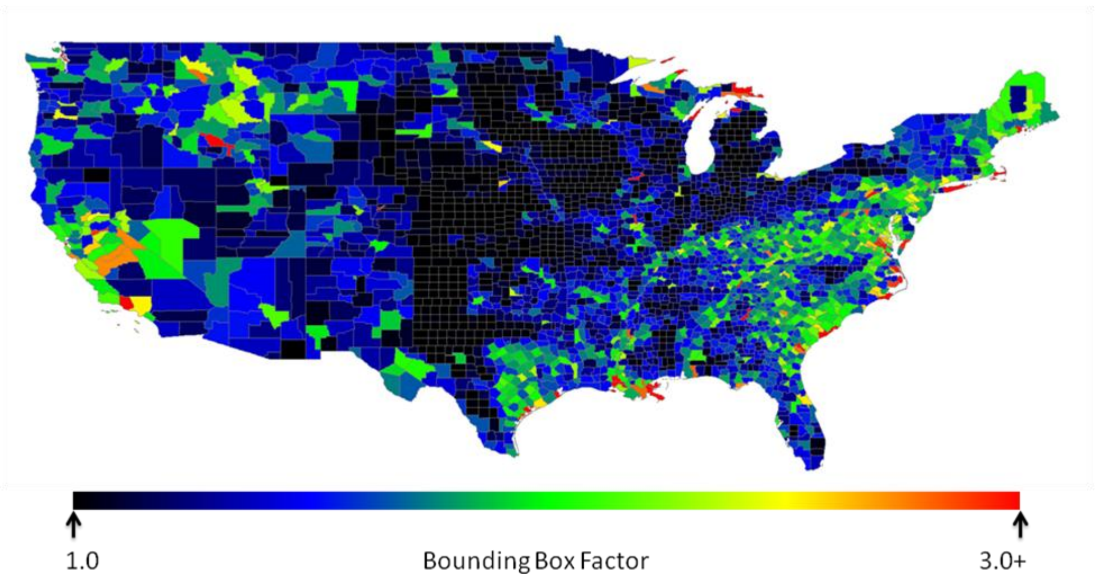

Figure 8 Bounding Box Factor Visualization

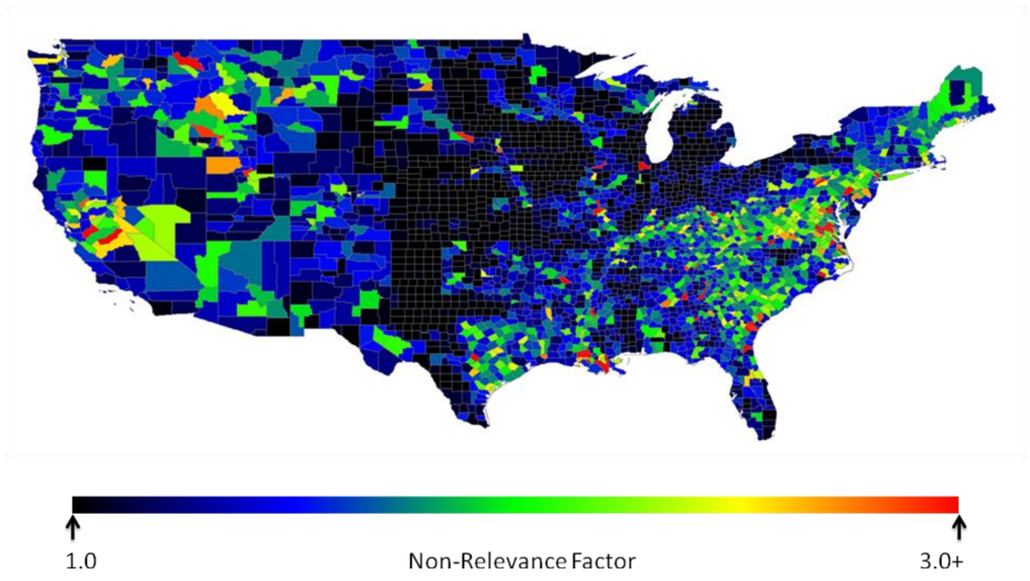

Figure 9 Non-Relevance Factor Visualization

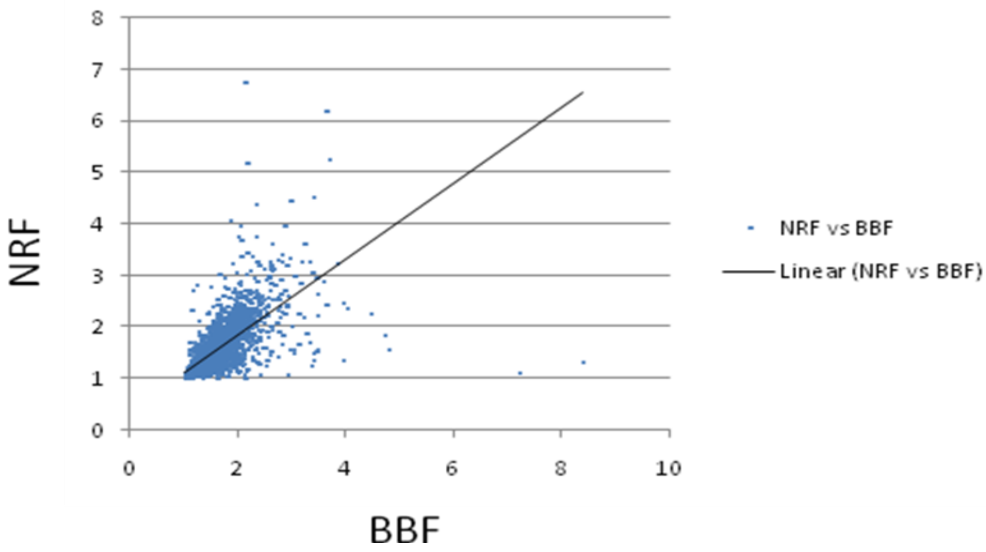

Figure 10 Non-Relevance Factor versus Bounding Box Factor

5.3 Search Response Time

Queries were performed for each county using both the bounding box and geographic identifier approaches as described in Chapter IV. Table 11 provides summary statistics for both approaches. The times are rounded to the nearest millisecond with the mean and standard deviation rounded to the nearest hundredth of a millisecond. Response times for the bounding box approach varied from 24 ms to 2688 ms with a standard deviation over 200 ms. Figure 11 displays a county-level visualization of the response times for bounding box searches and indicates that response time correlates to the size of the search area. The chart in Figure 12 plots the response time for each bounding box search (y axis) against bounding box area (x axis). The linear trend line also indicates a strong correlation between response time and the bounding box area. This is expected given that the number of terms the search engine must traverse increases with the search area since the terms are based on the latitude and longitude coordinates of the document locations.

Unlike the bounding box approach, response times for the geographic identifier approach are relatively constant with a mean value of 1 millisecond and a standard deviation of just 0.14 milliseconds. Additionally, the geographic identifier approach is on average two orders of magnitude faster, and the maximum response time is only 6 milliseconds compared to 2699 milliseconds. Figure 13 provides a visualization of geographic identifier search response times. The scale is adjusted to range from 0 to 6 milliseconds in order to provide some variation in the map. The map provides no indication of correlation between response time and area, a fact which is confirmed by the chart in Figure 14. The maximum response time of 6 ms for Benton County in Washington State, visible in Figure 13 and Figure 14, is an outlier and was not reproducible by additional tests. This outlier is likely the result of garbage collection in the Java virtual machine or background activities initiated by the operating system.

Table 11 Query Response Time Statistics

Statistic	**Bounding Box**	**Geographic Identifier**
Minimum	24 ms	0 ms
Maximum	2688 ms	6 ms
Mean	469.90 ms	1.00 ms
Median	369 ms	1 ms
Standard Deviation	209.95 ms	0.14 ms

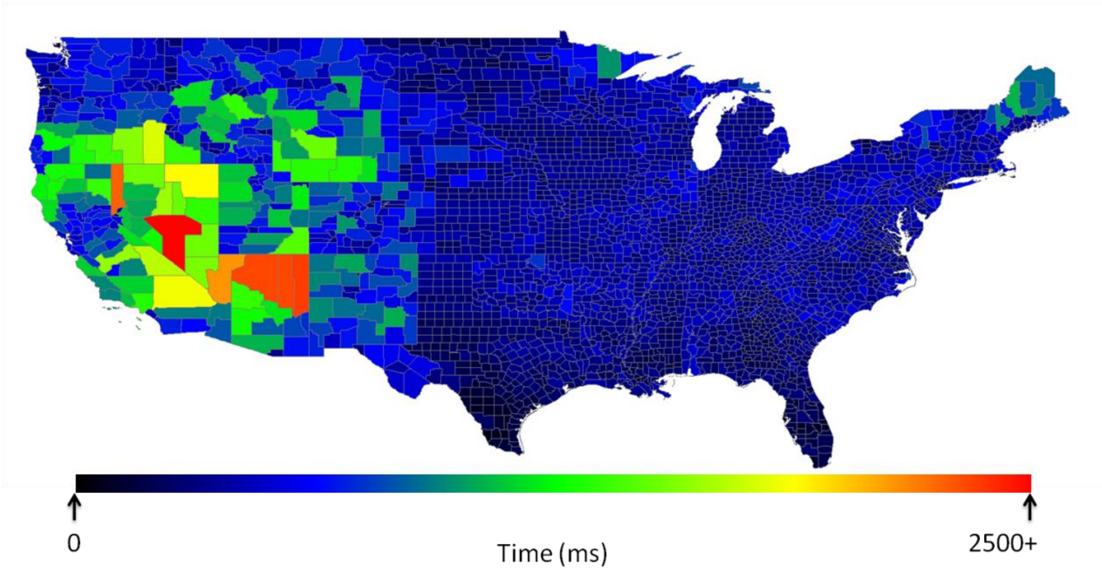

Figure 11 Bounding Box Search Response Time Visualization

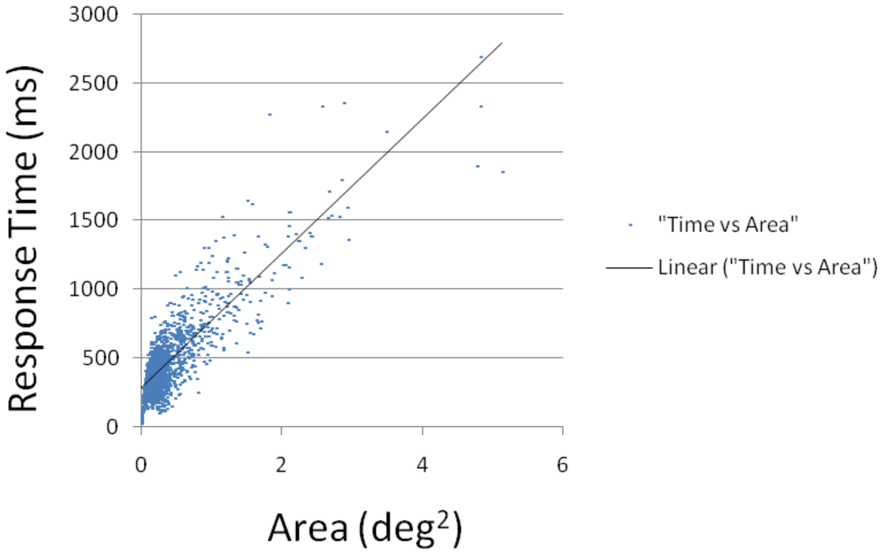

Figure 12 Bounding Box Search Response Time versus Bounding Box Area

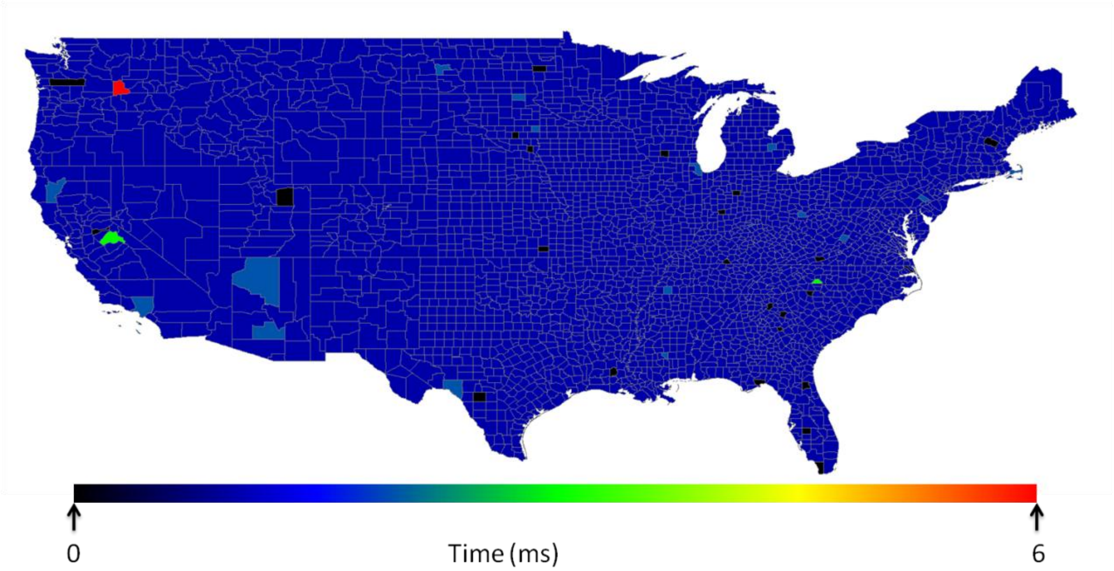

Figure 13 Geographic Identifier Search Response Time Visualization

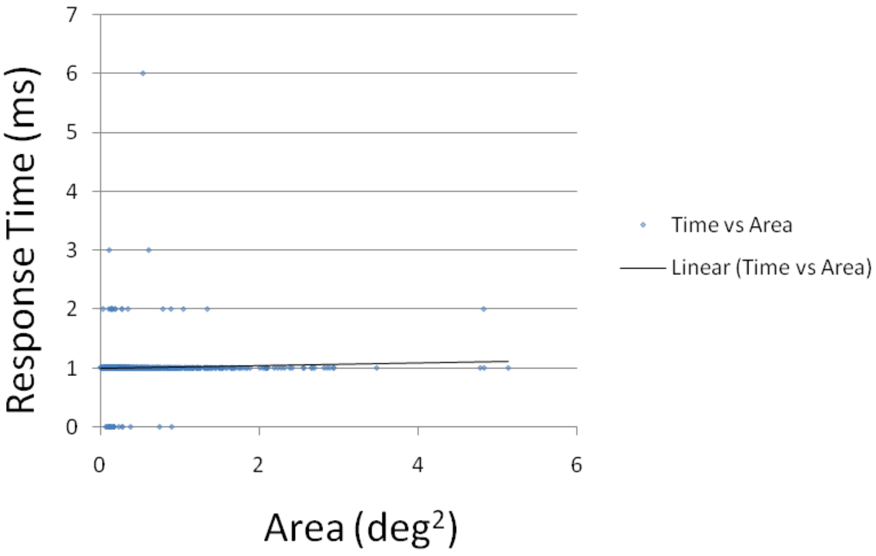

Figure 14 Geographic Identifier Search Response Time versus Area

CHAPTER VI

CONCLUSIONS

While location content has grown to play an increasingly important role in the web and web-based applications, the search technology that powers many web applications does not provide specific support for location-based search. Supporting location-based search using existing technology is significant because it provides a way for the many businesses and web-sites that depend on the current generation of search technology to integrate spatial search into their current capabilities. We implemented two techniques for performing location-based search using an unmodified open-source search engine and tested the hypothesis that searching based on geographic identifiers assigned using a gazetteer provides increased precision and faster response times than techniques using range queries with bounding boxes. By demonstrating that search based on geographic identifiers provides over two orders of magnitude performance improvement, as well as 100% relevant responses compared with over 30% non-relevant responses of the bounding box approach, we feel that we have sufficiently demonstrated the validity of our thesis statement.

There are numerous possibilities for future work related to this project. First, while our gazetteer implementation supports hierarchical relationships, this was not exploited by our searching techniques. Future work should explore the use of hierarchical relationships for improving search efficiency. In addition to hierarchical relationships, other semantic relationships, such as synonyms, should be explored. Second, our experiment only measured the performance improvement comparing indexing document locations based on point data. We believe that our approach will provide even greater improvement when indexing documents with bounding box and polygonal extends. Likewise, our experiment did not address projected data, which often has larger and less accurate bounding boxes. Finally, we hope to expand our technique to support more spatial operators by exploiting pre-computed binary topological relationships [7].

REFERENCES

[1] The Apache Software Foundation. (2000). *Apache Lucene.* [Online]. Available: http://lucene.apache.org/java/docs/queryparsersyntax.html.

[2] The Apache Software Foundation. (2007). *Lucene Solr.* [Online]. Available: http://lucene.apache.org/solr/.

[3] T. Bray, "On Search: Basic Basics," blog, 18 June 2003, http://www.tbray.org/ongoing/When/200x/2003/06/18/HowSearchWorks.

[4] D. Caldwell, "Unlocking the Mysteries of the Bounding Box," *Coordinates, Series A,* 29 Aug. 2005; http://purl.oclc.org/coordinates/a2.htm.

[5] Committee on Basic and Applied Research Priorities in Geospatial Science for the National Geospatial-Intelligence Agency, Mapping Science Committee, National Research Council, *Priorities for GEOINT Research at the National Geospatial-Intelligence Agency*, The National Academies Press, 2006.

[6] M. Copeland, "Location, Location, Location," *Fortune ,* vol.156, no.10, pp.147-150, 2007.

[7] M. Egenhofer, "A Formal Definition of Binary Topological Relationships," *3rd International Conference on Foundations of Data Organization and Algorithms (FODO),* LNCS 367, W. Litwin and H. J. Schek (ed.), Springer-Verlag, pp.457-472, 1989.

[8] "ESRI Shapefile Technical Description," white paper, ESRI. July 1998. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

[9] Federal Information Processing Standards Publication 5-2 (FIPSPUB5-2): *Codes for the Identification of the States, the District of Columbia and the Outlying Areas of the United States, and Associated Areas*, 1987 May 28.

[10] Federal Information Processing Standards Publication 6-4 (FIPSPUB6-4): *Counties and Equivalent Entities of the United States, its Possessions, and Associated Areas*, 1990 August 31.

[11] R. F. Finkel and J. L. Bentley, "Quad Trees: A Data Structure for Retrieval on Composite," *Acta Informatica,* vol. 4, pp. 1-9, 1974.

[12] Gazetteer, *The Oxford English Dictionary*, New York; The Oxford University Press, 2006.

[13] The GeoNames Geographical Database. (undated). *GeoNames Database Export* [Online]. Viewed 2007 November 20. Available: http://download.geonames.org/export/dump/US.zip.

[14] Hibernate Spatial. (2008, January). [Online]. Available: http://www.hibernatespatial.org .

[15] *ISO/TC 211/SC Business Plan 1.3*, June 2005; http://isotc.iso.org/livelink/livelink/fetch/2000/2122/687806/ISO_TC_211 __Geographic_information_Geomatics_.pdf?nodeid=1002196&vernum=0>. ISO_TC_211__Geographic_information_Geomatics_.pdf.

[16] *ISO 19112:2003, Geographic information – Spatial referencing by geographic identifiers*, 2003; http://www.isotc211.org.

[17] R.R Larson, "Geographic information retrieval and spatial browsing," *Geographic information systems and libraries: patrons, maps, and spatial information*, L.C.Smith and M.Gluck, eds. , The 1995 Clinic on Library Applications of Data Processing, April 10-12, 1995, 1996, pp. 81-124.

[18] C. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[19] MiMi.hu. (undated). "Minimum Bounding Rectangle." [Online]. Viewed 2007 November 20. Available: http://en.mimi.hu/gis/minimum_bounding_rectangle.html.

[20] J. Nielsen, "Mental Models for Search are getting firmer," blog, 9 May 2005, http://www.useit.com/alertbox/20050509.html.

[21] T. Reese. (2005, December). *C-squares Specification-Version 1.1*. [Online]. Available: http://www.cmar.csiro.au/csquares/spec1-1.htm.

[22] P. Rigaux, M. Scholl and A. Voisard, *Spatial Databases - with application to GIS*, Morgan Kaufmann, San Francisco, 2002, 410pp.

[23] U. Schindler and M. Diepenbroek, "Generic XML-based Framework for Metadata Portals," *Computers & Geosciences*, 2008, in press, doi:10.1016/j.cageo.2008.02.023

[24] D. Sunday. (2001). *Bounding Containers for Polygons, Polyhedra, and Point Sets (2D & 3D).* [Online]. Available: http://geometryalgorithms.com/Archive/algorithm_0107/algorithm_0107.htm.

[25] U.S. Census Bureau. (2007, February). *Cartographic Boundary Files. County and County Equivalent Areas 2000*. [Online]. Available: http://www.census.gov/geo/www/cob/bdy_files.html.

[26] U.S. Geological Survey. (2008). *U.S. Board on Geographic Names*. [Online]. Available: http://geonames.usgs.gov.

[27] S. Vaid, C.B. Jones, H. Joho and M. Sanderson, " Spatio-textual indexing for geographical search on the web," *Proceedings 9th International Symposium* (SSTS 2005), Angra dos Reis, Brasil, LNCS 3633, Springer, Berlin/Heidelberg, 2005, pp.218-235.

[28] Yahoo! Inc. (2007). *Yahoo! GeoPlanet.* [Online]. Available: http://developer.yahoo.com/geo/.

[29] R. Yang, X. Deng, M. Kafatos, C. Wang and X. S. Wang, "An XMLBased Distributed Metadata Server (DIMES) Supporting Earth Science Metadata*,"* *Proceedings of the 13th International Conference on Scientific and Statistical Database Management*, IEEE CS Press, 2001, pp. 251-256.

[30] Y. Zou, X. Xie, Ch. Wang, Yu. Gong and W-Y. Ma, "Hybrid Index Structures for Location-based Web Search," *Proceedings of the 14th International Conference on Information and Knowledge Management*, ACM New York, 2005, pp. 155-162.