Automatic Spatio-temporal Indexing to Integrate and Analyze the Data of an Organization

Craig A. Knoblock
USC Information Sciences Institute
Marina del Rey, CA
knoblock@isi.edu

Aparna R. Joshi
National Institute of Technology
Karnataka
Mangalore, Karnataka
aparna29th@gmail.com

Abhishek Megotia
USC Information Sciences Institute
Marina del Rey, CA
megotia@usc.edu

Minh Pham
USC Information Sciences Institute
Marina del Rey, CA
minhpham@usc.edu

Chelsea Ursaner
Office of Los Angeles Mayor Garcetti
Los Angeles, CA
chelsea.ursaner@lacity.org

ABSTRACT

Organizations are awash in data. In many cases, they do not know what data exists within the organization and much information is not available when needed, or worse, information gets recreated from other sources. In this paper, we present an automatic approach to spatio-temporal indexing of the datasets within an organization. The indexing process automatically identifies the spatial and temporal fields, normalizes and cleans those fields, and then loads them into a big data store where the information can be efficiently searched, queried, and analyzed. We evaluated our approach on 600 datasets published by the City of Los Angeles and show that we can automatically process their data and can efficiently access and analyze the indexed data.

KEYWORDS

spatio-temporal indexing, large-scale integration, efficient querying and analysis, data cleaning, urban data

ACM Reference format:
Craig A. Knoblock, Aparna R. Joshi, Abhishek Megotia, Minh Pham, and Chelsea Ursaner. 2017. Automatic Spatio-temporal Indexing to Integrate and Analyze the Data of an Organization. In *Proceedings of UrbanGIS'17:3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics, Redondo Beach, CA, USA, November 7–10, 2017 (UrbanGIS'17),* 8 pages.
https://doi.org/10.1145/3152178.3152185

1 INTRODUCTION

As we move to a world where everything is computerized, organizations are collecting and maintaining more and more data about everything they do. This is a huge problem for cities since they have to manage and maintain data about a broad range of city services and infrastructure. The challenge is how to help these organizations find and manage the information they already have, and how to provide integrated access to this information to both visualize and analyze the data across a wide range of sources.

We have been collaborating with the City of Los Angeles to provide better access to the data that they already maintain. LA City has published a great deal of their city datasets as open data available in two repositories: https://data.lacity.org and http://geohub.lacity.org. In these two repositories there are about 1,100 datasets that contain detailed information about everything related to the city from business licenses to which trees are planted in each park. The number of datasets is so large that departments within the city will often be unaware of relevant datasets.

To address this problem of locating and querying datasets, we developed a general approach to automatically organizing and indexing the data for a large organization, such as a city. The key idea is to exploit the fact that a large majority of the data for a city has a strong spatial and temporal component. Thus, we developed an approach to determine a spatio-temporal index of each record in each of the city data sources. To perform this task automatically, our algorithms analyze each dataset, identify the fields that contain spatial information for each record, identify the fields that contain temporal information for each record, and then normalize the spatial and temporal information into standard formats across all of the datasets. Once we have determined this spatial and temporal information for all the records, we than store every record from each dataset with the newly derived metadata into Elasticsearch.[1] Elasticsearch is a big data document store that provides efficient access to large numbers of documents. The resulting system, which will typically store millions of records, makes it possible to efficiently access all the records for a specified region within the city over a given time span. Queries such as which datasets contain any records for a particular area can be answered in just a few milliseconds, whereas searching each of the original datasets would take considerably longer.

In the remainder of this paper, we first describe our approach to automatic spatio-temporal indexing of the records in each dataset,

UrbanGIS'17, November 7–10, 2017, Redondo Beach, CA, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5495-0/17/11…$15.00
https://doi.org/10.1145/3152178.3152185

[1]www.elastic.co

including the methods to identify the spatial and temporal attributes, to automatically normalize those attributes, and to organize the resulting data for easy and efficient access. We also present an evaluation of the coverage and efficiency of this approach to indexing the data. Second, we describe how we use Elasticsearch to query and analyze the data, and we present an evaluation of the efficiency of this approach to querying the data. Next, we present related techniques on automatic indexing. Finally, we conclude with a discussion of the generality of the techniques and directions for future research.

2 SPATIO-TEMPORAL INDEXING OF THE DATA

In this section we describe the end-to-end techniques for automatically processing all of the data for a given organization. The system first identifies the spatial and temporal fields of each dataset, geocodes the addresses for any datasets that lack latitude/longitude information, normalizes the formats, and then creates and stores a document into Elasticsearch for each record in each dataset.

2.1 Identifying the Spatial and Temporal Fields

Because datasets usually contain spatial and temporal information in different representations, identifying spatial and temporal fields in multiple datasets is challenging. For example, date values can have many formats such as *dd:mm:yyyy*, *mm/dd/yyyy* or *dd MM, yyyy* and datasets can contain temporal data in any of these formats.

To identify spatial and temporal fields in the data, we use our previously-developed DSL approach to semantic labeling [14]. The purpose of DSL is to label unseen attributes by comparing them with sample data using various similarity measures. In our system, since we need to discriminate spatial and temporal attributes from other attributes in new datasets, we provide DSL samples of both spatial and temporal data. For spatial data, we have the following types of data: longitude, latitude, street address, city, state and zip code. For temporal data, we consider all temporal data as one general date/time type. All sample data is indexed and stored into Elasticsearch for faster queries in the prediction phase.

In the prediction phase, we need to recognize temporal and spatial attributes in new datasets. Many of these attributes are stored in unseen formats that our sample data does not cover. To identify temporal and spatial data with unseen formats, DSL extracts a set of similarity features between unseen and sample data for each attribute. Similarity features consist of:

- Attribute name similarity compares names between new attributes with indexed types using string similarity measures, such as Jaccard similarity.
- Value similarity compares the occurrences of overlapping values in attributes using metrics such as Jaccard similarity and TF-IDF cosine similarity.
- Distribution similarity acknowledges differences and similarities on how data are distributed over their range of values using hypothesis tests, such as Kolmogorov-Smirnov test.
- Histogram similarity recognizes similarity even when new attributes have different representations compared with our sample data. In histogram similarity, we convert values into histogram, which removes differences in representations and then use the Mann-Whitney test hypothesis test to compare these histograms.

After extracting similarity features, DSL uses a trained Logistic Regression model to classify whether a new attribute is similar to an indexed type. If the classification probability is higher than a threshold, we say that the attribute is similar with one of the indexed type and we can label the new attribute with that indexed type.

2.2 Geocoding Address Fields

For datasets that do not contain latitude and longitude coordinates, the system uses a geocoder to convert any available street address data into latitude and longitude coordinates. To perform this task, we use the TAMU geocoder. [2] To perform this mapping, the dataset must contain a spatial extent including a street address or zip code. In cases where there is only a street address and no city or zip code, we use a city of "Los Angeles" since all of the datasets are about LA.

Once we query the geocoder for a given location to retrieve its coordinates, we then cache this location-coordinate pair in a Mongo database.[3] We use MongoDb here to efficiency store and retrieve the cached data. This improves the speed of geocoding since we may encounter the same address in multiple datasets. It also reduces the load on the geocoder when we need to re-index the data when the city databases are updated.

Figure 2 provides an example of a street address taken from one of the datasets which is then converted to its corresponding geographical coordinates using the TAMU geocoder.

2.3 Normalizing the Data

In many datasets, the same information may be represented in a variety of formats. For example, we have found at least five different formats for the temporal data in the City of Los Angeles datasets. Since our goal is a fully automatic approach to analyzing the data, we need an approach to automatically clean this data without any user input.

We have developed an unsupervised method for automatic data cleaning. Suppose that we have two attributes a_1, a_2 which have the same semantic types but different formats. We want to convert data values in a_1 so that they will have the same format with data values from a_2.

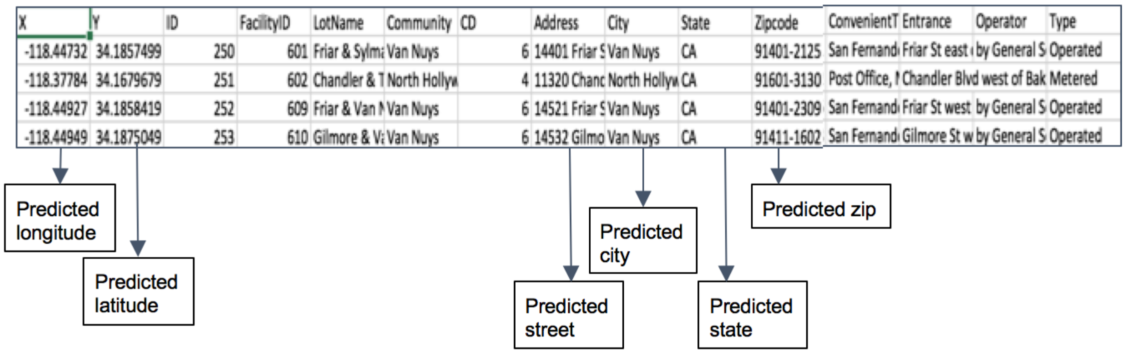
Our method consists of three different phases:

- Inferring data templates from attributes
- Mapping corresponding elements in templates across different attributes
- Replace corresponding elements in the standard format with elements from original format

In the inference phase, we implemented a recursive algorithm to recognize common substrings from sets of values from a format and cluster values into different groups, which requires different mappings to the standard format data. Details of our recursive algorithm are described as follows:

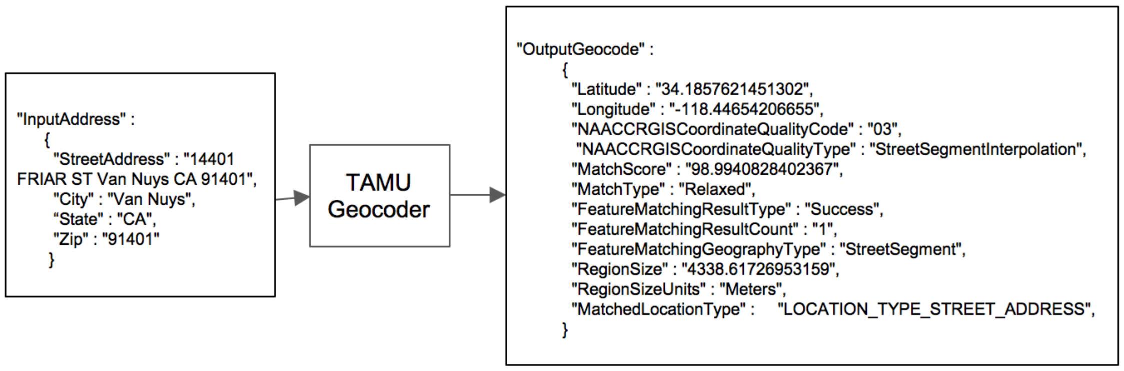
- Find the longest common substring (LCS) for all data values using suffix tress [9].

[2]http://geoservices.tamu.edu
[3]http://www.mongodb.com



X	Y	ID	FacilityID	LotName	Community	CD	Address	City	State	Zipcode	ConvenientT	Entrance	Operator	Type
-118.44732	34.1857499	250	601	Friar & Sylm	Van Nuys	6	14401 Friar S	Van Nuys	CA	91401-2125	San Fernand	Friar St east	by General S	Operated
-118.37784	34.1679679	251	602	Chandler & T	North Hollyw	4	11320 Chanc	North Hollyw	CA	91601-3130	Post Office, I	Chandler Blvd west of Bak		Metered
-118.44927	34.1858419	252	609	Friar & Van N	Van Nuys	6	14521 Friar S	Van Nuys	CA	91401-2309	San Fernand	Friar St west	by General S	Operated
-118.44949	34.1875049	253	610	Gilmore & V	Van Nuys	6	14532 Gilmo	Van Nuys	CA	91411-1602	San Fernand	Gilmore St w	by General S	Operated

Figure 1: Example of semantic labeling



```
"InputAddress" :
    {
      "StreetAddress" : "14401
FRIAR ST Van Nuys CA 91401",
      "City" : "Van Nuys",
      "State" : "CA",
      "Zip" : "91401"
    }
```

```
"OutputGeocode" :
        {
          "Latitude" : "34.1857621451302",
          "Longitude" : "-118.44654206655",
          "NAACCRGISCoordinateQualityCode" : "03",
           "NAACCRGISCoordinateQualityType" : "StreetSegmentInterpolation",
          "MatchScore" : "98.9940828402367",
          "MatchType" : "Relaxed",
          "FeatureMatchingResultType" : "Success",
          "FeatureMatchingResultCount" : "1",
          "FeatureMatchingGeographyType" : "StreetSegment",
          "RegionSize" : "4338.61726953159",
          "RegionSizeUnits" : "Meters",
          "MatchedLocationType" :    "LOCATION_TYPE_STREET_ADDRESS",
        }
```

Figure 2: Example of geocoding using the TAMU Geocoder

- If there is no LCS for all data values, find the LCS that appear in the most number of values. The LCS needs to appear in more than 50% of the data values.
- Split values by common substrings and cluster values that have the same number of tokens after splitting.
- Rerun the first three steps for each cluster until there is no LCS found in one cluster.

For example, if we have a temporal attribute in *mm/dd/yyyy* format and we want to convert data in this attribute to standard ISO 8601 format (*yyyy:mm:ddThh:mm:ss.sssZ*). Using our inference algorithm, we can infer that "/" is constant in *mm/dd/yyyy* format. Likewise, constants in ISO format are ":", "T" and ".".

After inferring templates from attributes, we will have a set of clusters for each attribute. Each cluster can be considered as a set of columns since they contain multiple sets of values which are the result of data splitting from the original attribute. With both *mm/dd/yyyy* and ISO formats, because we always have the same numbers of tokens after splitting on constants, we have one cluster for each attribute as shown in Table 1 and 2.

In the mapping phase, we use a similarity measure to map columns c_i from clusters in attribute a_1 to columns c_j in clusters in attribute a_2. Let X be a boolean matrix where $X_{ij} = 1$ iff column c_i is assigned to column c_j and S is a similarity matrix where S_{ij} is similarity score between column c_i and column c_j, the mapping objective function between clusters is as follows:

$$\max \sum_i \sum_j S_{ij} X_{ij}$$

In the replacement phase, data values in columns of clusters from s_2 will be replaced by corresponding columns in s_1. After that,

Table 1: Template of columns for mm/dd/yyyy

dd	/	mm	/	yy
...	/	...	/	...
...	/	...	/	...

Table 2: Template of columns for ISO format

yyyy	:	mm	:	dd	T	hh	:	mm	:	ss	.	sss	Z
...	:	...	:	...	T	...	:	mm	:	ss	.	...	Z
...	:	...	:	...	T	...	:	mm	:	ss	.	...	Z

we will have the final data that contains the same values as s_1 but has the format of s_2. For example, with data in *mm/dd/yyyy*, *dd*, *mm* and *yyyy* columns are mapped with corresponding columns in ISO format. After that, the values inside ISO format columns are replaced by values corresponding columns from *mm/dd/yyyy* format. Finally, we concatenate values in the same rows to obtain data in the ISO format.

In this work, our algorithm can only handle cases where format transformation requires replacement. We are working on improving the algorithm so that it can handle more sophisticated transformations.

2.4 Storing the Data in Elastic Search

ElasticSearch is a distributed full-text search and analytics engine built on Apache Lucene. ElasticSearch allows searching and analyzing of large volumes of data in near real time, thus explicitly addressing issues related to scalability, big data search and performance that traditional relational databases were never designed to support. Many applications can take advantage of advanced Lucene indexing or scoring capabilities of ElasticSearch. ElasticSearch also provides a web interface, Kibana, for querying and visualisation of data.

ElasticSearch has aided numerous organizations in overcoming the shortcomings of their previous approaches to meet the requirements of real time data processing, storage and retrieval [8]. We selected this open source technology as our integrated data store since it supports efficient spatial search, enhanced search capabilities, and low query response times. ElasticSearch provides an extremely rich query and indexing API.

We store all of the data across all of the original sources in ElasticSearch. In this process, we use a common format for representing all the data. The meta fields in ElasticSearch consist of "_index" to represent the index to which the document belongs, "_type" to denote the document's mapping type, "_id" for the document's ID and "_source" for the original JSON representing the body of the document. We introduce six fields at the top level of the _source meta field. These fields are described as:

(1) **table_name** for the name of the SOURCE table
(2) **row_in_dataset** for the row of the record (instance of the dataset) in the original source
(3) **url_link** for the address of the dataset
(4) **date** for the temporal extent of the record
(5) **location** for the spatial extent of the record
(6) **data** for the raw data for the record

An example of a record in JSON in the specified format is given in Figure 3.

Before we load the data in Elasticsearch in this specified common format, we define mappings for the Date and Location fields. We establish a geo_point mapping for the Location field. The geo_point datatype is an object with "lat" (latitude) and "lon" (longitude) keys, which Elasticsearch uses to support efficient spatial searches. Similarly, we define a date mapping with a "datetime" format for the Date field, which Elasticsearch uses to support efficient filters by the date range.

2.5 Evaluation of the Spatio-temporal Indexing

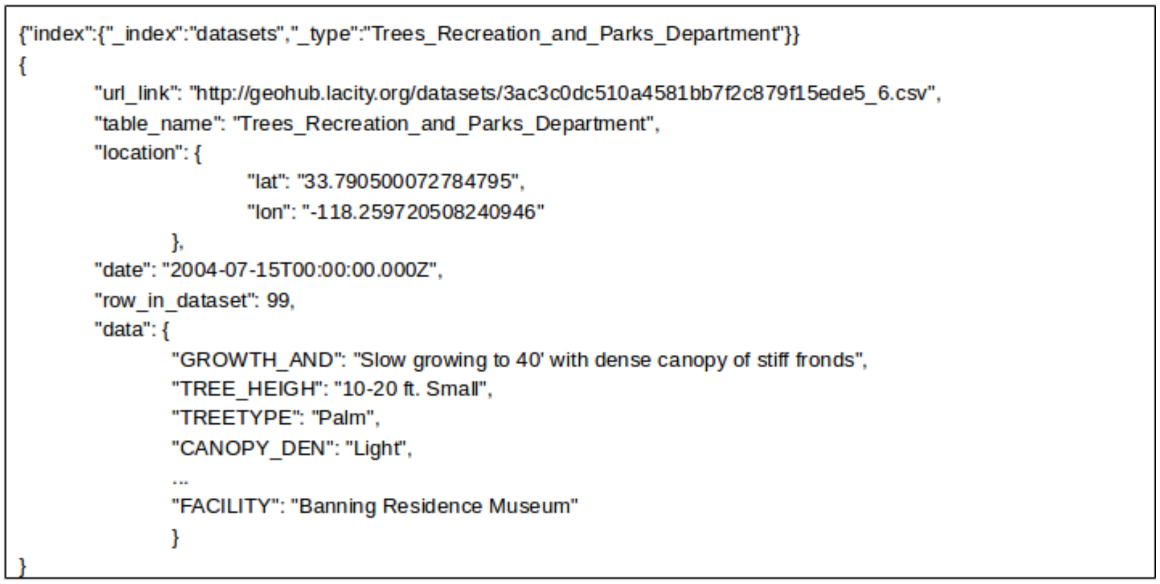
In our experiments we processed the first 600 datasets from the City of Los Angeles. Of these datasets, 300 were taken from the http://data.lacity.org portal while the remaining 300 datasets were taken from the http://geohub.lacity.org portal. Running on a Macbook Pro with a 2.3 GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 memory, it took approximately 60 hours to insert the records that could be processed from the 600 datasets, which included almost 4 million records. The large time duration was due to the large number of API calls made to the mongoDB cache and, if needed, the geocoder for the cases where there were no latitude and longitude coordinates present in the dataset.

Of the 300 datasets from the geohub.lacity.org portal, 177 of them had a spatial extent where each record could be mapped to a latitude/longitude coordinate. The remaining datasets were not mapped since they had a spatial extent consisting of some more complex shape involving multiple coordinates. Of the 177 datasets, 20 datasets did not contain any latitude and longitude coordinates, but they did have an address and we geocoded that address using the TAMU geocoder.

Table 3 shows the analysis of the mapping of the spatial data. Of the 177 datasets from Geohub, 23 datasets were incorrectly mapped to the wrong spatial extent. This was usually because the semantic labeling module incorrectly predicted a column as the spatial extent which was then passed to the geocoder resulting in incorrect coordinates. There were 122 datasets that were determined to have no spatial extent out of which 16 datasets were incorrectly predicted as they did have some spatial extent. This was because the semantic labeling prediction score of the coordinate column was below the threshold value of 0.25.

Table 4 shows the analysis of the mapping of the temporal data. Of the 177 datasets that were indexed into elasticsearch from the Geohub portal, 154 datasets have temporal extent and were correctly mapped by the model. There were 6 datasets which did not have any temporal data but the model incorrectly predicted them as having some temporal data. Two datasets did not have any temporal data and were correctly predicted by the model. Four of the datasets had temporal data but was not predicted by the semantic labeling module. The remaining five datasets predicted the wrong column for the temporal data.

As shown in Table 3, for the data.lacity.org portal there were 35 datasets which were identified as having some spatial extent out of which five were incorrectly mapped. And 256 datasets were predicted to not contain any spatial extent out of which 37 datasets



```
{"index":{"_index":"datasets","_type":"Trees_Recreation_and_Parks_Department"}}
{
        "url_link": "http://geohub.lacity.org/datasets/3ac3c0dc510a4581bb7f2c879f15ede5_6.csv",
        "table_name": "Trees_Recreation_and_Parks_Department",
        "location": {
                        "lat": "33.790500072784795",
                        "lon": "-118.259720508240946"
                },
        "date": "2004-07-15T00:00:00.000Z",
        "row_in_dataset": 99,
        "data": {
                "GROWTH_AND": "Slow growing to 40' with dense canopy of stiff fronds",
                "TREE_HEIGH": "10-20 ft. Small",
                "TREETYPE": "Palm",
                "CANOPY_DEN": "Light",
                ...
                "FACILITY": "Banning Residence Museum"
                }
}
```

Figure 3: A record of Trees_Recreation_and_Parks_Department dataset in JSON

did in fact contain some spatial data. This was because the model did not account for addresses containing street address and geo-points in the same column and also did not account for addresses stored in multiple lines instead of one single line. This can be solved by retraining the model accordingly. Of the 35 datasets, 17 of them had to be geocoded as they did not have any latitude or longitude coordinates present.

As shown in Table 4, Of the 35 datasets, 19 datasets were mapped correctly with their temporal extent and eight datasets were predicted correctly as not having any temporal extent. Five datasets were predicted to have some temporal extent when they did not have any temporal data in the dataset. Of these, two datasets had temporal data but the model mapped these datasets without any temporal extent. The remaining dataset was mapped to the wrong column.

In the 600 datasets, there were a few datasets that were faulty or unaccessible either due to maintenance of the datasets or dataset corruption.

3 QUERYING AND ANALYZING THE DATA

Once the data from all of the original data sources has been loaded into Elasticsearch, we can now efficiently query and visualize the available data.

Visualization of queries with geographic and temporal bounds on a map with street level precision enables easy analysis of huge amount of integrated data. One of the advantages offered by this practice is that users can interact more naturally with the data, without having to master a query language to understand the underlying structure of the data sets. The right presentation makes it easier to organize and understand the information in addition to gathering insights from analyzing the data.

To provide the visualization of the data, we use Kibana, which is an open source data analytics platform that is designed to work with ElasticSearch to provide an interface through which data in the ElasticSearch cluster is analyzed and correlated. Kibana leverages the API calls to access the ElasticSearch storage and sift through parsed data with user generated queries. In addition to providing interactive insights, Kibana makes monitoring the data simple with quick visualizations for the filtered data that results from queries.

Once all of the data is inserted into ElasticSearch, queries to filter the data based on spatial and temporal information can be performed efficiently and the results can be visually presented on maps. Different types of spatial queries can be performed such as retrieving all data present within some radius of a certain location or given the topleft and bottomright coordinates, getting back the data present in that rectangular box which is supported by the geo_distance and geo_bounding_box in ElasticSearch. Similarly, data can be filtered temporally with the range query, such as a query retrieving documents that have temporal information after and/or before a date value.

For instance, the query in Figure 4 retrieves and displays the documents that have locations that lie within 2km of the location specified by coordinates (latitude, longitude) as (34.052749942056835,-118.29490008763969) having dates that fall after the date specified, "19900101T17:56:13.833Z".

We can also efficiently perform queries to identify all of the datasets which have records that satisfy given geographical and temporal bounds. An example is shown in Figure 5. This query returns the record count for each table/dataset satisfying the specified filter conditions, which can also include keywords.

3.1 Evaluation of the Querying and Analysis

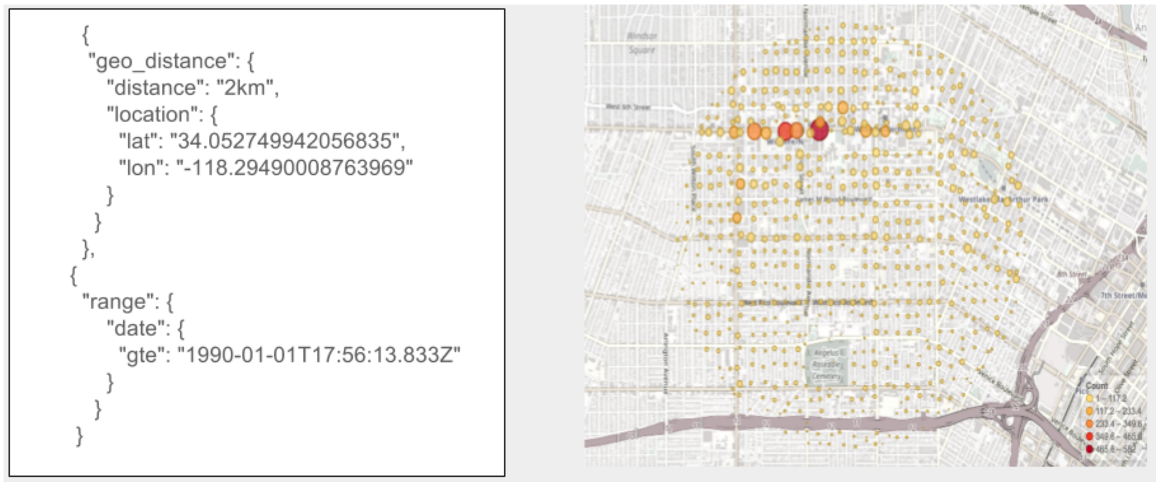
In this experiment, we evaluate the scalability of the model and the time taken to retreive the results on different types of queries.

Table 3: Analysis of indexing and mapping of the datasets with respect to spatial extent

Portal	Total Number of Datasets	Datasets predicted and mapped with spatial extent	Datasets which predicted incorrect Spatial Extent	Datasets predicted having NO spatial extent	Datasets incorrectly predicted as having NO spatial extent	Datasets which were geocoded	Faulty Datasets
http://geohub.lacity.org/	300	177	23	122	16	3	1
https://data.lacity.org/	300	35	5	256	37	17	9

Table 4: Analysis of indexing and mapping of the datasets with respect to temporal extent

Portal	Total number of datasets predicted and mapped with Spatial Extent	Datasets having temportal extent and correctly predicted	Datasets predicted with wrong column as temporal extent when no temportal extent exists	Datasets not having temportal extent and correctly predicted	Datasets having temportal extent but not predicted	Datasets having temportal extent but wrong column predicted
http://geohub.lacity.org/	177	154	6	2	4	5
https://data.lacity.org/	35	19	5	8	2	1



```
{
 "geo_distance": {
   "distance": "2km",
   "location": {
     "lat": "34.052749942056835",
     "lon": "-118.29490008763969"
   }
  }
 },
{
  "range": {
    "date": {
      "gte": "1990-01-01T17:56:13.833Z"
    }
  }
}
```

Figure 4: Query with a specified radius

At increments of each 500,000 records, we make different queries on the indexed ElasticSearch and record the time taken for each of these queries to run.

The different types of queries we ran our experiments are:

(1) Fetch all the records in a 2 km radius from a given coordinate x,y where the date is greater than some date d.
(2) Fetch all the data that is indexed in the data-store.
(3) Given a rectangular box with its top left coordinates x_1,y_1 and bottom right coordinates x_2,y_2, fetch all the records that are located inside this box.
(4) Fetch the table name and url for all the records in a 2 km radius from a given coordinate x,y where the date is greater than some date d.

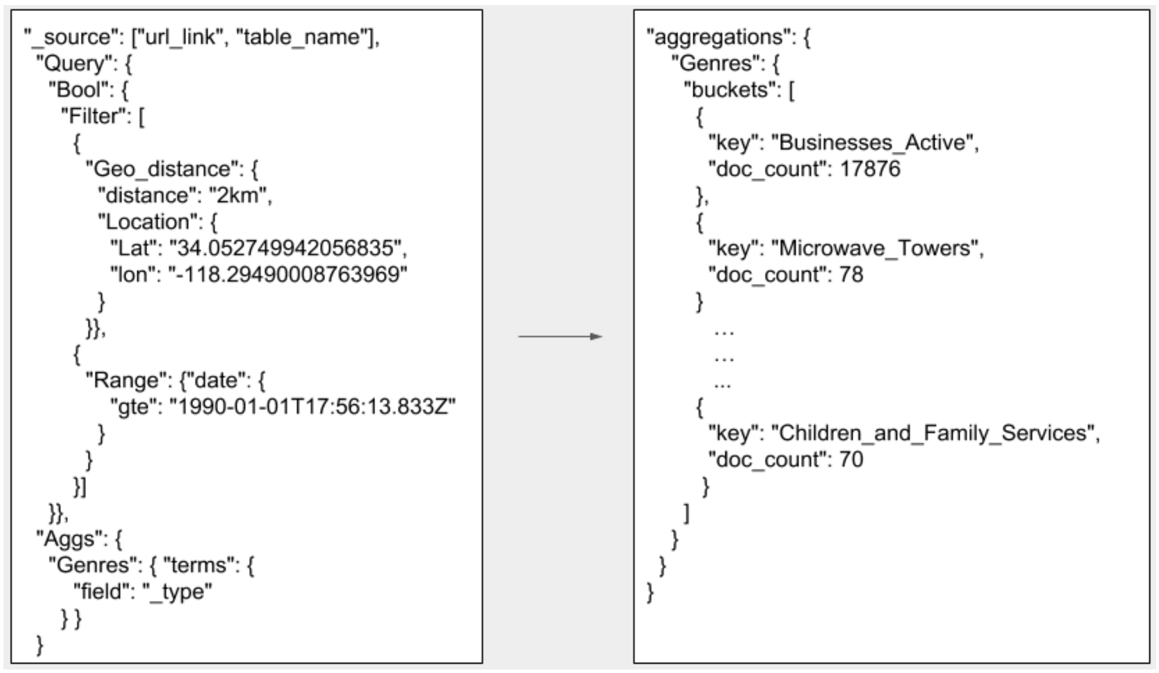
As is evident from Figure 6, the model is scalable since even on querying on larger number of records at each increment, the time taken to execute these queries is less than 100ms.

4 RELATED WORK

With a rapid rise in the spatio-temporal applications, the need for new query processing methods for data dealing with both the domains: spatial and temporal, is increasing. Some examples of these applications include location-aware services [12] and traffic monitoring [13].

Further, models have been proposed for automating extraction, integration and analysis of data. R. Ahsan, et al [1] describes a Data Integration through Object Modeling (DIOM) framework employing a spatial-temporal model for generalizing information extraction. The entity classifier proposed in the model extends the Stanford Name Entity Recognizer for identifying objects in the spatial-temporal model. While this work automatically classifies entities based on inference from implicit knowledge such as titles or footnotes (unstructured data) in spreadsheets, we focus on deducing the semantic types of data with machine learning techniques by training the model on relevant types.

Rattenbury [17] described an approach for extracting place and event semantics to assign to photos by burst detection assuming that significant patterns for event and place tags are manifested as bursts over small parts of time or space. This work relies on the usage distribution for extraction of spatial and temporal semantics



```
"_source": ["url_link", "table_name"],
 "Query": {
  "Bool": {
   "Filter": [
    {
     "Geo_distance": {
      "distance": "2km",
      "Location": {
       "Lat": "34.052749942056835",
       "lon": "-118.29490008763969"
      }
     }},
    {
     "Range": {"date": {
       "gte": "1990-01-01T17:56:13.833Z"
      }
     }
    }]
  }},
 "Aggs": {
  "Genres": { "terms": {
    "field": "_type"
   } }
 }
```

```
"aggregations": {
  "Genres": {
   "buckets": [
    {
     "key": "Businesses_Active",
     "doc_count": 17876
    },
    {
     "key": "Microwave_Towers",
     "doc_count": 78
    }
     ...
     ...
     ...
    {
     "key": "Children_and_Family_Services",
     "doc_count": 70
     }
   ]
  }
 }
}
```

Figure 5: Query for retrieving sources having a satisfying record

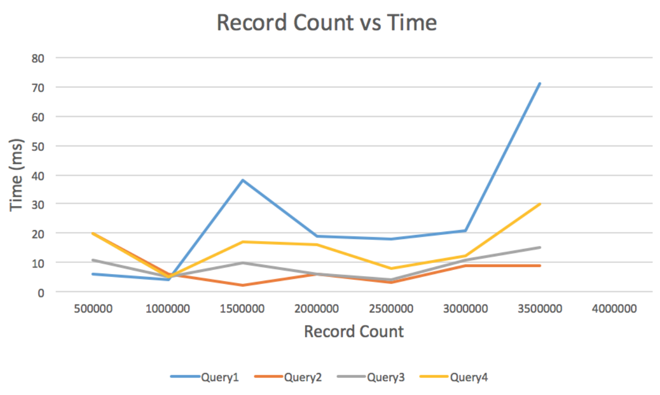


Figure 6: Scalability of the model using a time vs record count graph

whereas our approach is independent of the distribution of spatio-temporal information for identification of semantic types.

Before the use of temporal information in other applications is possible, the first task to solve is to extract and normalize temporal expressions. There are two commonly used methods to approach this. First is rule-based and second uses machine learning techniques. HeidelTime [18] is a proposed system for extraction and normalization of temporal expressions that falls in the first category as opposed to ours. HeidelTime is a rule-based system primarily using regular expression patterns for the extraction of temporal information and knowledge resources along with linguistic clues for their normalization. Numerous frameworks for mining spatial data have been proposed. Chawla et al [5], presents a framework for prediction of locations using map similarity.

Mei and Shen[11] have studied probabilistic approaches to automate labeling multinomial topic models where they model the problem as an optimization problem of minimizing KL divergence between distributions and maximizing mutual information between a label and a topic model. Attribute labeling has also been studied in Zhu's work [20], with efficient integration of all useful features by learning their importance. They propose a probabilistic model to integrate data record detection and attribute labeling with availability of semantics.

In the field of data cleaning, Raman et al [16] proposed a system called Potter's Wheel that automatically infer structure from data and allows user to interact with system to provide transformation rules. However, their inference method runs in exponential time since they enumerate all possible valid structures instead of finding constants and slots as our approach. Moreover, our approach also infers the transformations automatically based on clustering and similarity matching. Gulwani [6] has used programming-by-example (PBE) as an interactive way for a user to provide a small number of examples so that the system can transform data based on these examples. However, PBE systems require input-output pairs to be aligned to learn the transformation rules and thus cannot be used in data integration since we do not have the alignments between data across sources.

More semantically-rich location analysis problems have been studied in the domain of web-based information retrieval. Arampatzis et al's work [2] aims to extract geographic information for a web page, based on the page links and network properties, as well as geographic terms that appear on the page. The second related research effort [15] in GeoIR focused on extracting the scope of geographic terms or entities based on co-occurring text and derived latitude-longitude information. In [10], automated methods for extracting geo-temporal semantics from text, using simple text mining methods that leverage on a gazetteer service are described.

In recent years, with the explosion of published data from cities in the United States and around the world, multiple research projects have been conducted in integrating and using this large amount of data. Barbosa et al. [3] have done a detailed analysis of open urban data in 20 cities in the United States and Canada. They have run multiple experiments on understanding the data in different aspects such as volumes, schema diversity, data sparseness and informativeness, etc. However, all of their experiments are run separately and mostly based on heuristic rules, which makes these modules difficult to reuse in new applications. On the other hand, Ribeiro et al [4] implemented a system called UrbanProfiler that processes data from New York City Open Data and provides complex search queries, including spatial visualization of the data. While it is an automatic system, some of their components such as automatic type detection are built based on heuristic rules that are specifically for the data and may requires manual labor to extend for data in other domains. In contrast, our system provide a full pipeline for automatic labeling, normalizing, indexing and querying the data. All of our modules, such as semantic labeling, data normalizing and analyzing are built in an unsupervised manner. Therefore, the system can be adapted to run on other datasets from different domains with little effort.

5 DISCUSSION

The availability of big data infrastructure today makes it possible today to approach data challenges in novel ways. For example, we can integrate the City of LA's datasets by performing spatio-temporal indexing on every record of each dataset and constructing a document from those records and then loading them into a single big data document store. The resulting system then provides millisecond-based access to over four million records that were originally stored in separate databases. The resulting organization now makes it possible to find and access data in ways that would have been inconceivable even a few years earlier.

While this approach to indexing and integrating the data of a city provides a huge advance over what was previously available, it is just the first step in providing integrated access to the data of a city. Time and space provide a great way to organize much of the data, but there are many types of analysis that require a better model of the rest of the fields of the available data. In previous work, we developed semi-automatic methods to create a detailed model of a dataset in a system called Karma [7, 19]. However, given the human interaction required, that work does not scale well to the large number of datasets available here. So an important direction of future work is developing automatic mapping techniques that will support more detailed types of analysis over the data of a city.

ACKNOWLEDGEMENTS

We thank Dan Goldberg for generously allowing us to use the TAMU Geocoder (http://geoservices.tamu.edu/Services/Geocode/) to geocode many thousands of addresses.

REFERENCES

[1] R Ahsan, R Neamtu, and E Rundensteiner. 2016. Using entity identification and classification for automated integration of spatial-temporal data. *International Journal of Design & Nature and Ecodynamics* 11, 3 (2016), 186–197.

[2] Avi Arampatzis, Marc Van Kreveld, Iris Reinbacher, Christopher B Jones, Subodh Vaid, Paul Clough, Hideo Joho, and Mark Sanderson. 2006. Web-based delineation of imprecise regions. *Computers, Environment and Urban Systems* 30, 4 (2006), 436–459.

[3] Luciano Barbosa, Kien Pham, Claudio Silva, Marcos R Vieira, and Juliana Freire. 2014. Structured open urban data: understanding the landscape. *Big data* 2, 3 (2014), 144–154.

[4] Daniel Castellani Ribeiro, Huy T Vo, Juliana Freire, and Cláudio T Silva. 2015. An urban data profiler. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 1389–1394.

[5] Sanjay Chawla, Shashi Shekhar, Weili Wu, and Uygar Ozesmi. 2001. Modeling spatial dependencies for mining geospatial data. In *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, 1–17.

[6] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices*, Vol. 46. ACM, 317–330.

[7] Craig A. Knoblock and Pedro Szekely. 2015. Exploiting Semantics for Big Data Integration. *AI Magazine* (2015).

[8] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W Godfrey. 2014. Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 328–331.

[9] Laurent Marsan and Marie-France Sagot. 2000. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *Journal of computational biology* 7, 3-4 (2000), 345–362.

[10] Bruno Martins, Hugo Manguinhas, and José Borbinha. 2008. Extracting and exploring the geo-temporal semantics of textual resources. In *Semantic Computing, 2008 IEEE International Conference on*. IEEE, 1–9.

[11] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 490–499.

[12] Mohamed F Mokbel, Walid G Aref, Susanne E Hambrusch, and Sunil Prabhakar. 2003. Towards scalable location-aware services: requirements and research issues. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*. ACM, 110–117.

[13] Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao, and Liviu Iftode. 2004. Trafficview: A scalable traffic monitoring system. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*. IEEE, 13–26.

[14] Minh Pham, Suresh Alse, Craig Knoblock, and Pedro Szekely. 2016. Semantic labeling: A domain-independent approach. In *ISWC 2016 - 15th International Semantic Web Conference*.

[15] Ross Purves, Paul Clough, and Hideo Joho. 2005. Identifying imprecise regions for geographic information retrieval using the web. In *Proceedings of the 13th Annual GIS Research UK Conference*. 313–18.

[16] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 381–390. http://dl.acm.org/citation.cfm?id=645927.672045

[17] Tye Rattenbury, Nathaniel Good, and Mor Naaman. 2007. Towards automatic extraction of event and place semantics from flickr tags. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 103–110.

[18] Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 321–324.

[19] Mohsen Taheriyan, Craig A. Knoblock, Pedro Szekely, and Jose Luis Ambite. 2016. Learning the semantics of structured data sources. *Journal of Web Semantics* 37, C (2016).

[20] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 494–503.