

# Final Report

## **Eco-Pro: An Intelligent System for Shipping to Protect the Ecosystem of the Great Lakes**

**Principal Investigator:** Bing Liu

**Graduate Students:** Nitin Jindal and William N. Underwood.

Date: June 15, 2009

### **Main contributing students:**

Hari Prasad Divyakotti  
Nitin Jindal  
Kumaran Babu Kaliamoorthy  
Dinesh Kumar Subramanian  
William N. Underwood  
Yi Zhang

### **Students who helped at various stages:**

Xiaowen Ding  
Murthy Ganapathibhotla  
Lei Zhang

## **Executive Summary**

In this project, we attempted a very challenging task with a high potential impact. It is the vision of the Great Lakes Protection Fund, which we share enthusiastically. For the first time, ship tracking, ship scheduling and disease information are integrated in a single system at the global scale. This is a powerful concept which can benefit environmental and governmental agencies as well as private businesses. With the experience gained and basic research and development done in this proof-of-concept project, we believe that a full system can be built to realize the potential of such a system. The major challenges are in the process of obtaining a reasonably complete set of data from the public domain, cleaning them and making them useful in various forms. However, these challenges can be overcome with sufficient investment and with time. Like many web based projects, one needs to be creative in obtaining data from multiple channels and in an incremental manner. As more and more data are available on the Web, the task of obtaining data will become easier and easier. The project should be a long term one due to its usefulness in the long run and the ever changing world of information and technology. Maintenance and constant improvements will be needed to keep the system and all the information up to date. To make this project truly successful, it perhaps needs to be a private business with significant initial public or private investments. The resulting system will not only serve business interests but also environmental and governmental needs. We believe that the goal is achievable and the system can be constructed. A full system has the potential to create a major impact on society.

## 1. Project Objectives

The objective of this project is to track ships around the world and in particular those ships coming to the Great Lakes, their cargos, and news related to infectious disease and biological threats to the ports on the shipping routes. The task involves extraction of different types of data from Web pages around the world and integration of such data to produce alerts on possible negative effects of incoming ships on the ecosystem of the Great Lakes. This is a large-scale and visionary project and involves a full range of challenging Web and text mining technologies, e.g., web crawling, Web data extraction, text information extraction, information integration, data cleaning, and Web interfaces. It demands the team to have a diverse and rich set of research skills and experience in all these complex fields. We believe that it is well worth the effort. If a full system can be built, it can potentially change many aspects of our society, the shipping industry and the way environmental research and tasks are performed. We share this wonderful vision of the Great Lakes Protection Fund, and took up the challenge although as a computer science team, we know little to nothing about environmental research or the shipping industry.

This project aims to prove the concept and studies the possibility of building such a Web-based information system. The proposed system consists of the following subsystems:

1. A Web crawler for crawling Web pages
2. Ship and Port Data Extractor: This sub-system performs two tasks:
  - (a). Discover public data sources on the Web that contain ship movements information.
  - (b). Extract data from these sources.
3. A News Crawler and an Information Extractor
4. A Data Integrator
5. Cargo Data Extractor
6. A system for public reporting of information related to pollution, diseases and invasive species.
7. A Graphical User Interface System

We will describe these systems and provide a user manual for the system in the subsequent sections. Since many of the techniques are complex, we will not describe them in this report. Instead, we attached relevant research papers, technical reports and theses for readers who are interested in the technical details. One thesis also includes a survey of related existing systems in the world (Subramanian, 2007). Before diving into each sub-system, we first summarize our achievements, the main challenges faced in the project, and the lessons learned.

### 1.1 Accomplishments and Going Forward

We have built a functional proof-of-concept system for tracking ships, extracting ship schedules from the Web, and extracting disease information from news articles. A user interface is also provided for the user to explore the data and information. We believe that it is the first of its kind in the world. It integrates ship tracking, ship schedules, and a map of diseases around the world. Although there are many localized ship tracking systems based on AIS information, there is no system that has all three types of information at this scale. Even the most comprehensive Lloyds system does not have advanced ship scheduling information or disease information.

We believe that this is a very beneficial project. Through this, a number of basic technologies and modules have been developed. We also gained a great deal of experience in building a complex system involving so many cutting-edge Web, data and text mining technologies. With sufficient support, we believe that a full production system can be built, which will not only be useful to environmental and governmental agencies around the world, but also many types of businesses. At the beginning, the system may not have a large amount of data on all the ports and all the ships and their schedules. The data can be accumulated over time and obtained through multiple channels. The bulk of the data can be crawled and extracted from the Web, e.g., shipping company Web sites, port web sites, port operators' web sites, etc. As more and more information is available on the Web, we believe that automated extraction is becoming increasingly important. For those data that cannot be extracted from the Web, obtaining them through other means is possible (e.g., directly from the ports). This is actually the model used in most Web information systems because obtaining the complete data automatically is always very hard if not impossible. However, by offering incentives to companies who own the data it is possible to get their data feeds. In practice, it is unlikely that any system will ever have the complete data. The Lloyds system still lacks data from many ports and many ships, although it has been around for many years as a business.

Building the full system will need a significant amount of investments. Maintaining it is also capital intensive. Thus, the project may need to become a business at some stage in order to sustain itself. Becoming a successful business is a possibility. With the data, many types of data mining and data analysis can be performed. The results can be sold to interested parties. From the environmental point of view, the data can be used to find possible hazards, diseases and invasive species that may be carried by ships to different parts of the world. With the information, environmental researchers can study the likelihood that threats may survive the journey in the sea in order to predict whether they may cause problems for their local ecosystems, e.g., the Great Lakes region. From the business point of view, we believe there is a great potential too. For example, the system can produce and sell indices and data about shipping around the world in the same manner that Dow Jones produces and sells thousands of indices about the financial markets around the world. Shipping and traffic patterns at different parts of the world can also provide insight into economic activities around the world along with socio-economic changes.

To summarize, this is a major project, and significantly more resources are needed to build a complete system. For example, Lloyds had more than 70 engineers working on the project when it started with its system, and they did not work on the disease map of the world or extract ship schedules. Of course, with the smart use of technologies, we believe it can be done with a lot fewer people.

## **1.2 Challenges**

The main challenge that we faced was to obtain the complete set of data about ship tracking and ship schedules. In some parts of the world, the information about ship schedules is not publically available. For example, many US ports including some Great Lakes ports do not have their ship schedules online (partially due to security concerns after 9/11), e.g., the Port of Milwaukee. This causes the problem that the data in the system is incomplete. However, this should not hinder the development of the system because the data can be collected incrementally through different means rather than only through Internet crawling, e.g., installing AIS systems in the Great Lakes

region, and having agreements with ports to obtain their data feeds in turn for some benefits from the system.

The other major challenge is data cleaning. The data from different sources often have discrepancies due to schedule or route changes of vessels. Some sources may not reflect the changes quickly enough or do not make changes at all. This can cause many issues. However, this problem is also solvable through various means (we have implemented many of them). For example, the system can constantly crawl different sites and see which sites make changes to their schedules and also compare shipping company schedules with schedules from ports. Daily arrival and departure information from ports is often the most reliable. Another piece of information that can be exploited to detect changes is ship tracking information from AIS, which are useful for cross-checking or validating the correctness of the data.

Another minor issue is that many shipping companies actually include their cargo delivery schedules in their vessel schedule database. This includes land transportations rather than only ships. However, this problem is solvable during or after Web crawling and data extraction. During crawling, the system can check the modes of transportation (some sites provide such information). After extraction, a geo-database of the world (which we have) can be used to check whether there is a port at a particular destination, or the speed of vessels can be used to determine whether it is possible to travel a particular distance within a particular period of time.

### **1.3 Lessons Learned**

The main lesson learned is that it would be good to focus on specific applications very early in the research and system building so that our team can focus on meeting the user needs as soon as possible. The current system was built as a proof-of-concept general system. The data and various other types of information are not complete. If the user wants some specific information about a particular port (e.g., the Port of Milwaukee), the system may not have the data because the port data is not publically available. If this was planned earlier, it is possible to go through other means to obtain the data, e.g., to collaborate with the port authorities or even shipping companies. Having users involved early is also our experience with a recent Motorola project, which was also a two-year research project. In the first year, Motorola did not assign an expert user to work with us but just gave us some general guidelines and allowed us to explore. Then, when it came to the real users at the end of the first year, everyone thought the system was interesting, but no one wanted to use it because it did not solve any specific problem experienced by the users. In the second year of the project, Motorola assigned an expert user and engineer to work with us. Then, we built a novel data mining system that was exactly what they needed. The system has been in use in Motorola since 2006. Of course, the shipping project is a much larger-scale project. It involves so many aspects and technologies and thus is hardly comparable with the Motorola project. However, if this shipping project is to be continued sometime in the future (which we hope it will), we strongly suggest that real users be involved right from the beginning to guide the project in the direction that will make it useful to them as early as possible. The system can then be incrementally improved, enhanced and expanded as time goes by.

In the subsequent sections, we describe the sub-systems in the system. The writing here is meant to be an introduction or a general discussion about them without much technical depth. For technical details, the readers are referred to the research papers, technical reports and theses, which are attached with this report.

## 2. Web Crawlers

A web crawler is a computer program that browses and downloads web pages from the World Wide Web. It starts with a root URL (or a set of seed URLs), downloads it and extracts all the HTML links from that URL and then crawls them one by one and so on. There are many types of crawlers. The most well known are the universal crawlers, which are commonly used by Web search engines, such as Google, Yahoo and Bing. Such crawlers crawl the whole Web. These require highly sophisticated hardware, software and huge data centers, and of course a huge capital investment to build the infrastructure and to purchase/rent a huge network bandwidth. The whole operation is very expensive. Few companies in the world can operate such crawlers. There are also smaller scale crawlers which only focus on downloading some specific pages in some sites or all the pages in a site. In the process of our project, we built some small scale crawlers which are able to crawl all the pages in Web sites of interest.

The major challenges in crawling are how to make the system robust so that it does not crash when the remote sites and/or the network have problems, and how to constantly crawl a large number of sites at the same time efficiently and yet gracefully (i.e., without overloading the remote server systems). There are different methods of crawling a Web site. Two primary methods are breadth-first and depth-first. In the breadth-first approach, the system starts with the root URL and then crawls all the links in the URL. After that it crawls all the links at level 2 from the root URL and so on. In depth-first crawling, the system starts with a root URL and then go the maximum depth possible and then start again in another direction. For our work, we follow the breadth-first crawling because the depth of Web sites is not much and the schedule pages which we are looking for are usually at the depth of one or two. Breadth-first crawling is also the most commonly used strategy, which maintains a frontier queue to control crawling and a hash mechanism to make sure the same page will not be crawled more than once. This queue can grow very large and in that case some URLs in the queue may need to be put on the disk rather than stored in the memory.

In our project, we crawled three different types of information:

1. English news pages from around the world that report disease related information.
2. Information about ships and ports around the world.
3. Ship schedules from shipping companies and ports.

Since (1) is fairly simple as it usually only involves individual pages from a news site, we will not discuss it further. (2) and (3) are similar, requiring the crawling of sites to find the required information. We use Web sites of ports as a simple example to illustrate crawling.

### 2.1 Crawling Port Web Sites

**Home pages of ports:** For our project, we crawl the entire websites of many sea ports and shipping companies to find their vessel schedule web pages. The home pages of the ports were found from the website World Port Source (<http://www.worldportsource.com>). World Port Source provides interactive satellite images, maps and contact information for 2,438 ports in 185 countries around the world. This is a very good site if the user is just looking for the information about the ports. The most advantageous factor is that the whole port database can be crawled and

downloaded as an XML file. It also presents the information on a map and allows the user to search through the ports specific to a country.

### 2.1.1 Wget

To crawl and download a single web page or all the pages of a website we use the program “wget”. An example of wget is:

```
wget http://www.portofcleveland.com/maritime/schedule.asp
```

This will download the above mentioned URL. To download a whole website (ex: <http://www.portofcleveland.com>) we use the following command:

```
wget -r -nc -l5 -Q500m -o Log.txt --referer=url --no-parent -D http://www.portofcleveland.com -R mpg, mpeg, au, pdf, ps, gif, jpeg, jpg, JPG, JPEG, mov, avi, mp3, zip, gz, xml, doc, xls, mp4, iso, bin, exe, css, jsp, dat, flv, m3u, ppt, rar, swf, tar, txt, wav, wma, wmf http://www.portofcleveland.com
```

The options listed in the command are:

‘-r’: turn on recursive retrieving

‘-nc’: To prevent the file from downloading more than once in a directory

‘-l *depth*’: set the *depth* of crawling from the root page

‘-Q *size*’: limit the maximum *size* of the downloaded files to prevent space explosion

-o *logfile*: Log all messages to *logfile*. The messages are normally reported to standard error.

‘--referer=*url*’: sets http referrer *url*

‘--no-parent’: Do not ever ascend to the parent directory when retrieving recursively.

‘-D <http://www.portofcleveland.com>’: the domain which of the web pages to be crawled

‘-R mpg, mpeg, au, pdf, ps, gif, jpeg, jpg, JPG, JPEG, mov, avi, mp3, zip, gz, xml, doc, xls, mp4, iso, bin, exe, css, jsp, dat, flv, m3u, ppt, rar, swf, tar, txt, wav, wma, wmf’: exclusion list of file extensions not to crawl

### 2.1.2 CronTab

Cron is the name of a program that enables users to execute commands or scripts (groups of commands) automatically at a specified time/date. There are a few different ways to use cron. In the */etc* directory the user will probably find some sub directories called ‘*cron.hourly*’, ‘*cron.daily*’, ‘*cron.weekly*’ and ‘*cron.monthly*’. Placing a script into one of those directories will run it hourly, daily, weekly or monthly, depending on the name of the directory. If more flexibility is needed, one can edit a crontab (the name for cron's config files). The main config file is normally */etc/crontab*. The crontab will look something like this:

**SHELL=/bin/bash**

**PATH=/sbin:/bin:/usr/sbin:/usr/bin**

**MAILTO=root**

**HOME=/**

## # run-parts

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

The first part is almost self explanatory; it sets the variables for cron.

*SHELL* is the 'shell' cron runs under. If unspecified, it will default to the entry in the */etc/passwd* file. Now for the more complicated second part of a crontab file, an entry in cron is made up of a series of fields, much like the */etc/passwd* file is, but in the crontab they are separated by a space. There are normally seven fields in one entry. The fields are:

### minute hour dom month dow user cmd

- minute : This controls what minute of the hour the command will run on, and is between '0' and '59'
- hour : This controls what hour the command will run on, and is specified in the 24 hour clock. Values must be between 0 and 23 (0 is midnight)
- dom : This is the Day of Month that the user wants the command run on, e.g. to run a command on the 19th of each month, the dom would be 19.
- Month : This is the month a specified command will run on. It may be specified numerically (0-12), or as the name of the month (e.g. May)
- Dow : This is the Day of Week that the user wants a command to be run on. It can also be numeric (0-7) or the name of the day (e.g. sun).
- user : This is the user who runs the command.
- Cmd : This is the command that the user wants to run. This field may contain multiple words or spaces.

If the user does not wish to specify a value for a field, he/she can place a \* in the field.

## 2.2 Sources for Ship Schedules

We get the ship schedules from two different types of sources: Shipping companies and port websites.

### 2.2.1 Ship schedules provided by Shipping Container Companies.

We get the ship schedules from various shipping companies. The reasonably big shipping companies often share the voyage information of their vessels on their websites. This information is sometimes up to 4-5 months in the future. We crawl the vessel schedules from 4 such big shipping companies listed below:

- Shipmentlink (<http://www.shipmentlink.com/>)
- CSAV (<http://www.csav.com/>)
- Kline (<http://www.kline.com/>)
- China Shipping (<http://www.chinashipping.com/>)

Apart from these 4 major shipping companies, we also crawl data from the website:

<http://www.schednet.com>

This website gathers the vessel schedules information from about 35 different shipping companies and displays it on their site.

The four shipping companies mentioned above were selected from the list presented on <http://www.schednet.com/weblink/s-centre.asp>

This page has links to vessel schedule pages for 23 major shipping companies around the world.

Crawling of these sites is often very tricky because they do not have explicit Web pages for schedules. Instead, Web query interfaces are used so that the user can issue queries to get required schedules. Our system has to simulate a human user inputs and download the returned Web pages with schedules. For more details about Web query interfaces, please refer to (Liu, 2007).

**Note:** For ships visiting the United States, these shipping companies may also provide the information on how the cargo travels on land from one city to another after a ship has offloaded it at a visited port. For example, a ship may visit the port of Long Beach, California and then the cargo on the ship will go to different cities like Chicago, Cleveland, etc. The shipping companies provide information on the ETA of ships at Long Beach and also the ETA of the cargo at Chicago and Cleveland. Often, this information is mixed with the vessel schedules and difficult to separate. This is true for most of the big shipping companies which we crawl. To deal with this problem, we do not consider the schedules for inland US ports (including Great Lakes ports).

### *2.2.2 Ship schedules on the website of sea ports*

Many ports provide the data for incoming and outgoing ships on their websites. This data is often very accurate, but is limited to only up to one month in advance. We crawl the ship schedules for 114 such ports, which covers the major ports across the world. However, only 19 out of these 114 ports are located in the United States, which is relatively low given the high volume of traffic handled by US ports. We believe this is primarily due to security restrictions imposed after 9/11.

**Find the port schedule page.** Manually browsing through a port's website to find the vessel schedule page is very time consuming. Below, we discuss different methods employed to find such pages automatically:

1. A vessel schedule page will change regularly. Most of the port websites are relatively simple and easy to crawl. So, we crawl the entire port website multiple times over a period of one month and then select the web pages which have changed over that period.
2. A vessel schedule page will contain keywords like "Vessel schedule", "Ship schedule", "ETA", etc. A vessel schedule page will also contain a lot of dates for ETA or ETD of a ship compared to a normal page.
3. Search engines can be used to search for vessel schedules. For example, search queries like "Vessel Schedule" return a large number of web pages from different sources which have ship schedules. The query "Vessel schedule site: [www.portofcleveland.com](http://www.portofcleveland.com)" will return the pages from the site [www.portofcleveland.com](http://www.portofcleveland.com) with keywords "vessel schedule". We used three big search engines, Google, Yahoo and Live search (which is now called Bing), to find ship schedule pages.



The above three methods (which are done automatically) reduce the number of candidate pages which could contain ship schedules. From there, we manually go through these web pages to confirm the pages are indeed ship schedules.

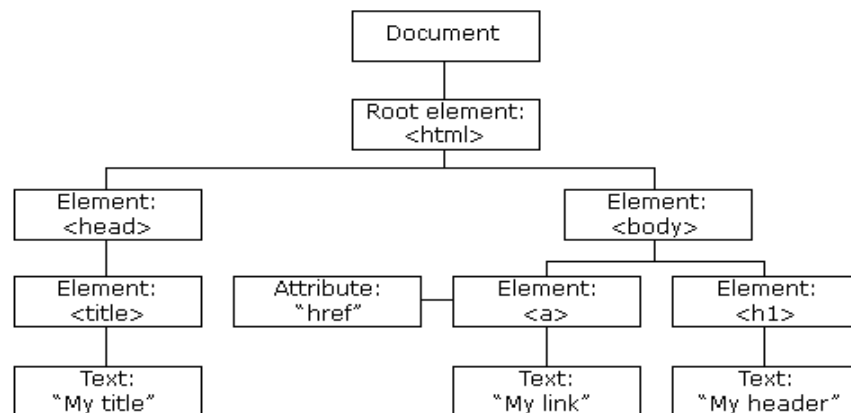
**Port Arrivals Website:** We found a website <http://www.portarrivals.com> which tracks schedules for about 20000 ships. This website is also a major source for our ship schedule data.

### 3. Data Extractor

Once Web pages are crawled and downloaded to the local machine. The system needs to perform data extraction to extract the specific data that is interesting to us, e.g., ship schedules, ship particulars, etc. Since a typical Web page contains a great deal of irrelevant information, identifying and extracting the required information is a major challenge. In Web data mining research, this is called structured data extraction or wrapper generation. Let us use the ship movement data as an example to introduce our data extractor. The ship movement data are usually displayed in tabular form in Web pages. Thus, such data on the Web are called *structured data* because they follow some regular patterns. These data are typically retrieved from underlying databases and displayed on the Web following some predefined templates (but unknown to us). Just like in a database system, each row in the Web table is likely to be a data record. Figure 2 (later) shows the sailing schedule from a Web site. All the information is displayed in a table. Note that there are also other more complex formats. Our task is to automatically extract data items contained in the tables in Web pages and put the data in our local database. Our system starts by building an HTML DOM tree and then finding patterns from the tree in order to extract the required data.

#### 3.1 HTML DOM Node Tree

The HTML DOM represents an HTML document as a tree structure. The tree structure is called a node tree. All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created. The node tree below (Figure 1) shows the set of nodes, and the connections between them. The tree starts at the root node and branches out to the text nodes at the lowest level of the tree:



**Figure 1.** An example of an HTML DOM Tree

### *Node Parents, Children, and Siblings*

The nodes in the node tree have a hierarchical relationship to each other. The terms *parent*, *child*, and *sibling* are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).

- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent

#### *3.1.1 Parsing HTML Documents*

**Converting HTML documents to XHTML:** Extensible HyperText Markup Language (XHTML) is a reformulation of HTML 4.0 to make it XML based. HTML documents are generally poorly written, with no closing tags, ill-formed node hierarchies, etc. So, the conversion of HTML to XHTML is required so that we can parse the HTML document into a DOM Tree for the data extraction step. Conversion of HTML to XHTML follows certain rules as laid out by the W3C. The W3C is the organization that coordinates standardization of Web protocols. It has defined three types of XHTML documents. This is based on the XML Document Type Definition (DTD) that is used by the document. For our needs, we have to deal with HTML pages with frames, so we need a DTD which handles “Frameset”. This is because a lot of vessel schedule web pages have frames, where the frame in the page contains the actual schedule and it sometimes points to some third-party URL.


We experimented with different methods and tools which convert HTML to XHTML and return the DOM tree. A few examples are JTidy, Html Parser, CyberNeko HTML Parser, and Mozilla Gecko. We finally settled on CyberNeko (<http://sourceforge.net/projects/nekohtml>) because it was the most stable and fastest converter for the kind of web pages we deal with. After the HTML document is converted into XHTML the next step is to use an XML parser to build the DOM Tree.

**The difficulty of manual extraction of ship schedules:** Manually extracting the schedule data from a web page which has ship schedules is a very time consuming process. This is because different web pages have different organization and internal structure (or template) so one solution does not fit all. At the beginning, we wrote a manual parser to extract ship schedules for a handful of web pages including pages from shipping companies, the website <http://www.portarrivals.com>, and about 5 port web pages. The format of these web pages has not changed since we started crawling them in November 2007. So, the manual extractor built for these pages is very stable. For the rest of the data, we wrote an automatic extractor which is discussed below.

### **3.2 Automatically Extracting Ship Schedule from a Web Page**

Automatic Web data extraction is a challenging problem with a wide range of applications. This project focuses on the extraction of structured data. As we discussed above, they are typically descriptions of objects retrieved from underlying databases and displayed on Web pages following some fixed templates. Extraction of such structured data (vessel schedules in our case) enables one to integrate data from multiple Web sites into one database.

The figure below (Figure 2) shows an example of a ship schedule taken from the port of Cleveland website. The center of the page is the HTML table. The first row of the table contains the columns headers and the other 3 rows contain the data. This is the most commonly used format to display schedules. So, our goal is to automatically extract this table from such pages.



Arriving	Vessel	Line	Agent	Stevedore
6/17/2009	FEDERAL PRIDE (CYPRUS)	FED NAV	WORLD SHIPPING	FEDERAL MARINE TERMINALS
6/21/2009	ISOLDA (CYPRUS)	POLSTEAM	COLUMBUS SHIPPING	FEDERAL MARINE TERMINALS
6/28/2009	KONINGSBORG (NETHERLANDS)	WAGENBORG	COLUMBUS SHIPPING	FEDERAL MARINE TERMINALS

**Figure 2.** Vessel schedule table from port of Cleveland

### 3.2.1 Automatic Pattern Recognition

As shown in Figure 2, the central table has three rows of vessel schedule. The three rows follow the same pattern. Each row is divided into 5 cells and the corresponding cells in each row have similar content. For example, cell number 1 in each row shows the date of arrival. The contents in column 1 are different from other columns. Below, we give a brief look at the HTML table content of the vessel schedule table. The contents with the same color are the repeating patterns.

```
<TABLE>
  <TH> <TD>Arriving</TD>....
  <TR>
    <TD>6/17/2009</TD>  <TD>FEDERAL PRIDE (CYPRUS) </TD>
  <TR>
    <TD>6/21/2009</TD>  <TD>ISOLDA (CYPRUS)</TD>
  <TR>
    <TD>6/28/2009</TD>  <TD>KONINGSBORG (NETHERLANDS) </TD>
</TABLE>
```

So, the repeating pattern of this vessel schedule is:

```
<TR><TD>Date</TD><TD>Ship name (Country)</TD> ...
```

### 3.2.2 Our Approach

In our method, we follow an unsupervised bottom-up approach to identify repeating patterns (a data mining solution). First, we identify the simplest repeated patterns. For example, the ship names, list of countries, agent names, dates, etc. Then we move up in the DOM Tree to identify the more complex repeating patterns based on any repeating patterns at the child sub-trees.

**Identifying atomic patterns:** For identifying the atomic patterns, which is the similar text content in a table we rely on the following approaches:

1. Identify the atomic known text patterns base. This includes the following:

- a. Ship names. Use a ship information database table to find the column which has the maximum number of entries matching the ship names. This step can generate a lot of false positives because we have 165K ships in our database and a lot of text nodes can find some match even if it's not a ship name. So, we rely on the assumption that there is only one ship name text node in each row of a table. If multiple columns match, we only consider the one with maximum matches.
  - b. ETA and ETD. We first identify the date columns, for which we use 40 different date/time formats (ex date formats: Wed, Jul 4; '01, 4 Jul 2001; 01/05/2008; July 1; etc). To distinguish between ETA and ETD we use three rules in the order of importance.
    - i. The earlier date is labeled as ETA
    - ii. ETA is generally mentioned before the ETD
    - iii. If there is only one date pattern, it is labeled as ETA by default
2. Identify the rest of the atomic patterns. After the known patterns have been identified, the remaining patterns (ex: patterns line, agent, etc) are identified based on the following:
- a. Alignment. The contents in multiple rows will be aligned in a straight line, because that is how structured data is presented.
  - b. Similar neighbors. After we have identified a few patterns, we perform bootstrapping to identify other patterns based on the fact that if two text content nodes are near or are surrounded by the same patterns, then those two nodes will also form a pattern.

**Identify complex patterns:** The next step is to move up the DOM tree to identify the complex patterns. For our work, most of the time the patterns have only the depth of two (row entry followed by the columns). So, we restrict ourselves to the patterns of level 2. Also, we are only interested in patterns which do not have complex components like nested tables, div tags, etc. Two nodes belong to the same pattern if the majority of the corresponding child sub-trees belong to the same pattern. A manual threshold is used to set a cutoff score.

**Identify column headers:** In some web pages, the vessel schedule table also has column headers. So, identifying these column headers helps us in assigning labels to the corresponding patterns. The column header is usually the first non-empty row of the table. It also contains keywords like “name”, “eta”, “agent”, “date”, “arrival”, etc. This process helps in the integration stage if the labels to the patterns are known, which we discuss in the next section.

**Column span:** The major issue while automatically extracting data from a table is dealing with column spans. Sometimes, a cell in the table can span multiple columns. For example, consider the table in the figure below (Figure 3). The top row contains the two headings. But, the other two data rows are split into three because Vessel Info has two parts, ship name and ship agent. So, such a table has to be expanded based on the column span to extract the data. The expansion is done such that the Vessel Info cell is split into two so that the top row also has three cells.

Vessel Info		ETA
ZIM Xiamen	Evergreen	2009/01/03
YM Seattle	Yang Ming	2009/01/04

Figure 3: Column Span example

More details about data extraction can be found in (Jindal and Liu, 2009; Liu, 2007), which has highly sophisticated algorithms for automated data extraction.

Apart from extracting ship schedules, we have crawled and extracted information about ships and ports from various Web sites and stored it in our local database (the schemas are given in the appendix).

Number of ports with information in database:	2037
Number of unique ships with details in database:	365913
Number of unique ships with call sign in database:	131280
Number of ports that have websites:	1458

## 4. Data Integration

After data are extracted from multiple Web sites, they need to be integrated so that they can be put in the same database table. The goal of data integration is twofold:

1. Fill the extracted data into the database table based on a pre-defined schema
2. Resolve conflicts between schedule data from multiple sources

The first step means to match the data columns and values from one site with those of the other sites. For example, many sites use tables to list ship movement information. The question is how to know which column in each site is the name of the ship, or the departure date, etc, and how to match such information from multiple sites. The problem is hard because different Web sites use different words to express the same information. There are many approaches in the research of data or information integration (Liu, 2007). We took the approach of starting with a global schema and then matching the labels from each web site to that global schema. We believe that this approach is appropriate for our application because for the shipping domain, we know what pieces of information are important for vessel movements and schedules. All the extracted schedule information is then mapped to this global schema. Regarding the second step, we use many approaches and indicators to solve the problem. Below, we describe the two sub-systems.

### 4.1 Schema-Based Data Integration

We get ship schedules from different sources as outlined above. We then need to match them. In our case, we already know the schema of the final table. That is, we look for the following 9 items when extracting the ship schedule data which is the global schema used in our database:

Ship name, Ship call sign, Ship IMO No., Port, Expected time of arrival (ETA), Expected time of departure (ETD), Port of origin, Port of destination, Agent/Owner/Manager name

Not all sources have all the fields listed above.

In the shipping domain, each ship has a unique callsign. Using callsign as the starting point to integrate data is natural. Thus, for each extracted data table, the system starts by finding the corresponding callsign of the ship name and finding the corresponding port entry for the port to get the unique ID of the port.

**Finding ship callsign from ship name :** In some Web sites, the web pages do not list the ship callsign, just the ship name. So, we match the ship name with our database to get the callsign. If multiple rows in the database match the ship name, then that ship schedule is not considered. We

found that for roughly 10% of the ships in the ship schedules we could not find a unique call sign, hence they were not considered further.

Once the ship names and callsign are resolved, we match the other columns. Traditional techniques in data integration are applied to match columns and values, e.g., using linguistic similarity (name similarity) of labels (or schemas in the web tables) and value similarity of items from different sites. For example, ETA is an abbreviation for the expected time of arrival. The value match refers to values of the same type to be identified. For example, the date can be expressed in many forms. We have compiled these forms and use them to identify the date columns from different sites. In many cases, a combination of both label and value similarities is used to identify the right column matches. Due to different data formats, we also need to convert them into one standard format to be stored in our database. For details of these techniques, refer to (Liu, 2007).

Below, we also discuss some other related issues.

**Multiple ports with the same name:** A small number of ports in different countries share the same name. For example, there is a port Vancouver both in Canada and the United States. This does not pose a big problem, because most of the time the ship schedule data also have the port country field which helps resolve the conflict. However, for cases where the ambiguity cannot be resolved, we do not consider those schedules.

**Different types of sources:** For filling the database table with the schedules extracted, the schedules from different sources are treated separately. Any conflict resolution due to the same ship schedule showing in different sources is described below. The sources are:

schednet, shipmentlink, csav, kline, chinashipping, others

For a given ship, when we get the new ship schedule from the most recently crawled data, we have to update the schedule for that ship in the database. Sometimes, the new ship schedule conflicts with the data already in the database. This can happen because the ship may have changed its route due to bad weather, etc.

**Updating data from a shipping company:** The shipping company usually provides the entire schedule information for a ship given some date range (for example, within 2 months of the date of crawl). So for a given ship, we delete the old schedule in the database which falls within that range and then insert the new ship schedule.

**Updating data from a port website:** The ship schedule on port websites is very accurate but sometimes that can change too but only by a few days. For example, if a ship was slotted to arrive at a port on May 5, but a few days later the port website updates the schedule and now the ship is slotted to arrive by May 7. So, for a given ship and port, we get the latest ETA and then delete the old entries from the database within two days of that ETA before inserting the new entry.

## 4.2 Resolve Schedule Conflicts: data cleaning

### 4.2.1 Conflict between Two Data Points on Map

The two schedule points (ship, port1, ETA1, source1) and (ship, port2, ETA2, source2) for a ship conflict if the actual time taken to travel from port1 to port2 (based on longitude/latitude distance and average ship speed of 20 nautical miles per hour) is not the same as the difference between

ETA1 and ETA2. This system is not fully accurate, because the ships rarely follow a straight path from one point to another. But, only 15% of the ships have any conflict at all and the low average ship speed used by us limits the damage to only a small number of cases.

**Haversine Formula:** The distance between two latitude/longitude points on the map is computed based on the Haversine formula. The Haversine formula ‘remains particularly well-conditioned for numerical computation even at small distances’ – unlike calculations based on the spherical law of cosines. The formula below calculates the distance “d” between two points on (lat1, long1) and (lat2, long2).

$$\begin{aligned} R &= \text{earth's radius (mean radius} = 6,371\text{km)} \\ \Delta \text{lat} &= \text{lat2} - \text{lat1} \\ \Delta \text{long} &= \text{long2} - \text{long1} \\ a &= \sin^2(\Delta \text{lat}/2) + \cos(\text{lat1}) \cdot \cos(\text{lat2}) \cdot \sin^2(\Delta \text{long}/2) \\ c &= 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \\ d &= R \cdot c \end{aligned}$$

#### 4.2.2 *Smoothing with AIS Data*

Some of the ship schedules are not accurate, primarily due to error in reporting, outdated schedules, etc. We also get the ship AIS data which gives the longitude (Long) and latitude (Lat) information of a ship. So, we remove schedule entries which are inconsistent with the AIS data. Consider a schedule entry of a ship visiting port A on date D1. Let the geographical location of port A be (Lat1, Long1). Now, we find the entry in the AIS data for the ship when it was last seen near date D1. Consider the point (Lat2, Long2) on date D2. We compute the time it would take the ship to travel from the nearest AIS point to the port. This is calculated based on the difference between points (Lat1, Long1) and (Lat2, Long2) and assuming the average ship speed is 20 nautical miles per hour. If the time is not approximately the same as the difference between the dates D1 and D2, the entry is deleted.

#### 4.2.3 *Conflict within Schedule Data*

**Reliability of sources:** When two schedule data points conflict, we can either delete both the points or delete only one of them. The preference is decided based on the reliability of the source from which the schedule data point was generated. In our data set, we found the schedule data points from the port websites and other non-shipping company related web pages (referred to as “others”) to be most reliable followed by schedule from the shipping companies followed by the schedule from the website schednet. So, the order of preference among the sources is:

$$\text{others} > \{\text{csav, china shipping, kline, shipmentlink}\} > \text{schednet}$$

This ordering is used to decide which schedule data point to delete when two points conflict.

**Resolving conflict between the schedule data:** For a given ship, we first sort the schedule by ETA. Then we look at the consecutive schedule points to check if they conflict or not. If two consecutive schedule data points conflict, we delete the data point which is from the source with the lowest reliability based on the order described above. We repeat this process until no conflicting schedule data point from the lowest reliability source is left. After that, we move on

to the next least reliable source and so on until no conflicts are remaining. So, the conflict is resolved in the following three steps:

1. Delete all schedule data points from source 'schednet' which conflict with some other data point
2. Delete all schedule data points from sources {csav, china shipping, kline, shipmentlink} which conflict with some other data point
3. Delete all schedule data points from other sources if they conflict with some other data point

## **5. Ship Tracking**

The system combines both observed and reported ship movement data. Observed data is drawn from publicly-shared Automatic Identification System readings, while reported movements come from the World Meteorological Organization's Voluntary Observing Ship Project. These sources are described in detail below.

### **5.1 AIS Data**

Automatic Identification System (AIS) is used by ships worldwide for tracking and identification purposes. Ships employing the system continuously broadcast their vessel details along with their current navigation system readings over VHF. All passenger ships and all ships with gross tonnage of at least 300 tons must transmit AIS data. The range of such a system depends on the size and strength of the transmitter, but is typically around 20 nautical miles. Many hobbyists and other interested parties receive AIS transmissions and share them publicly through a central server managed by Centro de Observação Astronómica no Algarve (COAA). COAA sells software called ShipPlotter for monitoring publicly shared transmissions (Figure 4).

As ShipPlotter does not provide any automatic mechanism for data export, a custom VBScript application was developed to translate individually-mapped data points into an XML format for standardizing and ease of processing. The application is scheduled to run continuously, extracting new data points as they are reported by the server. An additional .NET application was developed to monitor changes to the XML data, translating any new records to the global schema, allowing it to be merged with data from various sources. The ShipPlotter program, the software for extracting ShipPlotter data, the data transformation applications, and the eventual database all reside on a computer running Microsoft Windows.

<http://www.coaa.co.uk/shipplotter.htm>

### **5.2 VOS Data**

The World Meteorological Organization (WMO) runs a program called the Voluntary Observing Ship Project (VOS). The goal is to collect real-time meteorological observations from points across the globe, aiding in climate study and weather forecasting. A byproduct of these reports is the ability to track participating ships based on the locations associated with each transmission.

The VOS data is shared through the Local Data Manager (LDM) system, which consists of a network of primarily academic institutions providing data streams for consumption by research organizations. Data from the VOS streams is continuously intercepted via TCP/IP and stored in



local files. The data received by LDM consists of hundreds of files with data ranging from seismic information to the desired ship movement data. The first processing step is to discover which files contain relevant ship information. To do so, the contents of all files are crawled and a probabilistic approach is used to determine which files warrant further inspection. Ship location reports within these files are often buried in irrelevant data and formatted inconsistently. As such, an intelligent parser was designed to locate quality individual reports within each file and translate structured, semi-structured, and unstructured reports into a format compatible with the global schema. All components of the system run on Ubuntu Linux.

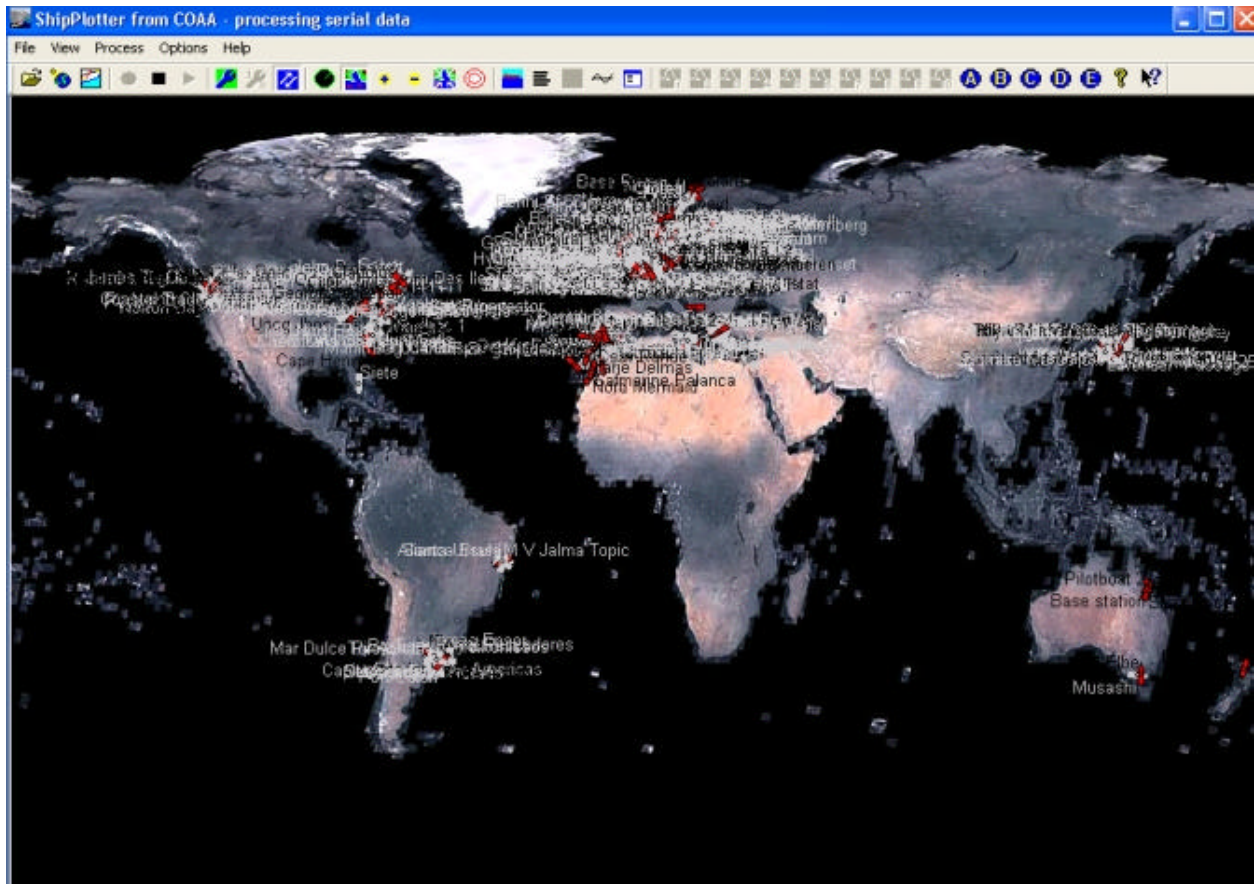


Figure 4. ShipPlotter interface.

In recent years, concerns over ship and port security have prompted U.S. government agencies and the International Maritime Organization to advise against including call signs in VOS transmissions. As a result, many ships now mask their actual call signs, reducing the amount of usable data that is publicly available from voluntary reports.

<http://www.unidata.ucar.edu/software/ldm/>

### 5.3 Ship Movement Data

The information collected for each ship location data point includes the call sign, report/observation date, latitude, longitude, destination, ETA, speed, and course. All transmissions include the call sign, date, latitude, and longitude, while the other information is provided inconsistently depending on the privacy preferences and/or hardware capabilities of the ship.

As of early June 2009, the database contains over 7.5 million ship movement data points from nearly 30,000 different ships. Wikipedia reports that approximately 40,000 ships carry AIS class A equipment. The website [boatnerd.com](http://boatnerd.com) lists 203 ships that traveled Great Lakes waters in 2008. Our database contains movement data for 138 of those ships in 2008.

## 6. Information Extractor

This sub-system crawls and downloads articles from news Web sites around the world. Our current effort is focused on English language news sites. We use Google News to help with the work as Google has almost all English news sources in the world. The system is now able to do the following: automatically monitor Google health news in real time and extract emerging disease reports (EDR) in news articles, filter those irrelevant articles, and extract specific disease outbreak information from new articles (i.e., the disease name, time and location of the outbreak). In detail, the system performs the following tasks:

1. Connect to the news articles from Google News. We wrote a program that can connect to Google News and download all the Google health news constantly.
2. News content extraction. This is the same problem as the schedule extraction, but more complex (see (Jindal and Liu, 2009; Liu, 2007) for techniques). It is called structured data extraction. The purpose is to extract the title, the news agency, the time and the news content from each news Web page. This is a difficult problem as there are thousands of news sites in the world and each site has a different format. Obviously, we cannot do it manually or even write a program for each site to extract its information. We have designed and implemented an automatic extraction algorithm to extract those pieces of information by discovering patterns from pages.
3. Sentence classification to identify EDR sentences. For each news article, the system builds a machine learning model based on support vector machines (Liu, 2007) to classify whether the sentence contains emerging disease reports or not. We call this step semantic classification as opposed to the traditional text classification.

The setting of semantic text classification is the same as traditional topic-based text classification. We have a set of documents  $D$  and each document  $d_i \in D$  is labeled with a class  $c_j \in C$ , where  $C$  is a set of known classes. A supervised learning algorithm is applied to build a classification model. However, semantic text classification usually has more refined categories or classes. Different classes are hard to be separated based on bag-of-words or n-grams alone. For example, the sentence, “*the district hospital reported that 10 people were diagnosed with cholera early today*”, reports a possible cholera outbreak. It is easy to observe that the words “reported” and “diagnosed” are indicative of an outbreak. The times, “today” and “this morning”, indicate a new event. Using the words alone, however, is insufficient as

the following sentence illustrates: “10 people from the district hospital submitted a report early today on cholera diagnosis.” This sentence uses very similar words, but has a completely different semantic meaning, i.e., it does not report a disease outbreak at all.

In this work, we define semantic information as any information extracted from the text that is not based on keywords or n-grams. Clearly, there are multiple levels of semantic information. At the highest level, it is the full understanding of the text, which is still not possible with the current technology. At lower levels, we have features with different amounts of semantic contents, which can be extracted from sentences based on the current NLP techniques. The exact features used in this work are discussed in (Zhang and Liu, 2007).

An interesting question is whether the traditional bag-of-words representation is still useful in semantic text classification. We believe that the answer is yes for the following reasons:

1. To express some special semantic meaning, certain specific keywords are still more likely to be used, although the same words can be used to express other information but with less frequency. See the example above.
2. Semantic feature extraction is still not perfect. There may be many errors. Keywords can help offset some of these errors.

Figure 5 illustrates the difference between traditional text classification and semantic text classification as described in this work. Note that we do not make a difference of the types of classes or texts used in a classification task because classification of any type of categories may be assisted by some level of semantic information. For the complete detail, please refer to the attached research paper and thesis (Zhang, 2008).

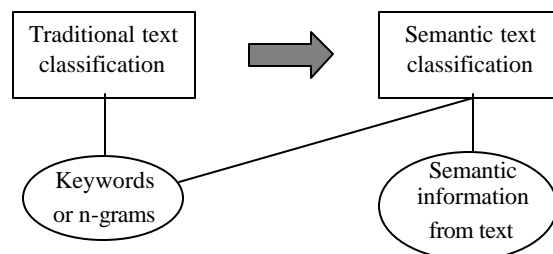


Figure 5: Traditional text classification and semantic text classification

4. Disease extraction: After separating EDR sentences from non-EDR Sentences, the next step is to extract the specific information about an outbreak from the identified EDR sentences. There are three kinds of information we are interested in: outbreak location, outbreak time and the name of the disease. For outbreak time, since we are dealing with real-time news, any EDR news and thus EDR sentences must report an outbreak in a recent time period, so the report date of the EDR news can be regarded as the outbreak time approximately. For name of the disease, it is possible to obtain a list of known disease names using a database. Thus, our task is confined to extract only the location of the outbreak. More specifically, we study the problem of extracting EDR location named entities from EDR sentences.

**Problem definition:** Given an EDR Sentence, extract the EDR location named entities of the outbreak.

Note that it is assumed that the given sentences are EDR sentences as the problem of classification of EDR sentences has been done in the last step.

Clearly, the problem is an information extraction problem. The general information extraction problem has been studied by numerous researchers and many existing techniques have also been reported in the literature. Conditional random field (Lafferty et al., 2001) is perhaps the most effective general approach to solving the problem. However, we showed that it does not perform well based on our data (Zhang, 2008). We proposed a novel technique based on sequential pattern mining to generate extraction rules. These rules are then used to match and to extract disease locations.

Specifically, we use label sequential rules and class sequential rules for the purpose. These rules are described in (Zhang and Liu, 2008) together with their mining techniques. Traditional sequential pattern mining in data mining does not generate any rules, but only produces frequent sequences that meet a user-specified minimum support. Thus, each type of rule mining consists of two steps, frequent sequential pattern mining (unsupervised) and rule generation (supervised).

Another important novelty of the proposed technique is that the data used for mining or learning are sequences obtained in dependency trees from natural language parsing. That is, only some important paths in dependency trees are used rather than the original sentences. This is because the structure in a dependency tree embeds the essential relationships of concepts in a sentence, while the information contained in a raw sentence can be quite diverse, making it difficult to mine key patterns to be used for extraction. The details are quite involved, please refer to (Zhang, 2008) for details.

The whole process of the proposed technique consists of the following steps:

- a. Obtain the first sentence that contains a disease and a candidate location from each news article. This sentence is usually the first or the second sentence in the news report, which is not surprising.
- b. Build a dependency tree of each sentence using a parser. In our work, we use Minipar (Lin and Pantel, 2001) for the purpose. The list of diseases is also input into Minipar so that it can also recognize disease names that we are interested in. Our disease names are obtained from Centers for Disease Control and Prevention (Centers for Disease Control and Prevention, 2008).
- c. Extract relevant paths of each dependency tree to form the sequence data for mining and learning. The detail will be discussed in (Zhang, 2008).
- d. Mine sequential patterns from the path sequence data, and generate label and class sequential rules based on the manually tagged data of disease locations. Label sequential rules and class sequential rules are combined to form a classifier to be used to recognize whether a candidate location is a disease outbreak location.
- e. Test the classifier using unseen test data using cross-validation to assess its precision recall and F value.

To the best of our knowledge, the proposed technique has not been used in existing approaches. To evaluate the proposed technique, we use a large number of health news articles crawled from Google News (Google News, 2008) in 17 days and historic reports

from ProMED-mail (ProMED-mail, 2007). The extraction result demonstrated the effectiveness of the technique. The proposed technique is also compared with the current state-of-the-art extraction algorithm conditional random field. Our method outperforms conditional random fields significantly. See (Zhang, 2008) for details.

To further improve the technique, we found and crawled a geo-name database, which contains most location names of every country in the world. Many country and city names also have abbreviations (e.g., US, LA). We have added those discovered abbreviations into the database. Our extraction system first extracts the disease, location and time from a news article, and then confirms the location name with the geo-database. One main issue is that many local newspapers/news agencies do not state the location of the disease outbreak in detail. For example, a local US newspaper may only state the name of a local town without giving the name of the state. This is fine for local readers because the name has no ambiguity, e.g., Oak Park in a Chicago newspaper. However, for a person who is not local, this presents a difficulty. The same is true for a system because there are many US cities/towns having the same name in different states. To solve the problem, we have searched the Web and found some Web sites that list news agencies and their physical addresses. The lists help us to determine the disease location referred to in the newspaper. A crawler has been written to crawl the lists and store them in our database. However, these lists are not complete. We plan to find additional sources in the future which contain the missing news sites and their physical addresses (this is again an incremental process). For those news sites for which we cannot find addresses, we try to determine their locations from their Web sites. This is a challenging problem because it is not easy to figure out where to find the address from a site.

For full details about information extraction related to diseases, please refer to the thesis (Zhang, 2008).

## **7. A Cargo Data Extractor**

This was determined to be infeasible because most cargo data is not publicly available for free. For the private data from Piers, we subscribed for a year. The database contains all the cargo of ships coming to the US. It also contains the information about senders and recipients of cargos. The extraction from the subscription is fairly simple as the data can be downloaded as Excel files. There is one major limitation to this data, namely, it is only fully available one month after the ship has arrived at a US port. This will not be useful for checking for environmentally unfriendly cargos. However, it is useful to find cargo patterns for business and other uses. This data is huge. The subscription service is not sufficient to retrieve the data because there is a limit on the number of cargo rows that can be downloaded. If this project is to be continued, it will be useful to work in collaboration with Piers or the Coast Guard, who own the data. The project can provide data mining algorithms to mine useful patterns from the data. In exchange, the project can get to use the data for environmental and other purposes.

Of course, for cargo information that is available on the Web, extracting them using our Web extraction technology will not be a problem.

## 8. Producing “Warning Reports”

Our current prototype system has the ability to provide ship schedules of several ports in the Great Lakes, and a few hundred major ports around the world. The system also extracts the disease-related news information. However, we are unable to produce the warning reports as the scope of the project does not allow a fully implemented risk ranking system. This is proposed as a main task in a subsequent project together with researchers from the University of Notre Dame.

## 9. Graphical User Interface

We have built a graphical user interface to the system for the user to interact with the system. The user can issue various commands or provide system parameters. The visual display system is based on the Google Maps UI, which can have different modes of display: an illustrated map or satellite imagery. The system has many functions to show ports, ships, ship schedules and disease information. The details of the user interface are given in the user manual in the appendix.

## 10. Project Summary

This project attempted a challenging and integrated task of ship tracking, ship schedule crawling and extraction, and disease crawling and extraction, which involves a whole range of state-of-the-art Web, data and text mining techniques. For the first time, ship tracking, ship schedule and disease information are combined in a single system. This visionary project was championed by the Great Lakes Protection Fund. A full system once built can have a major impact on environmental and governmental agencies as well as private businesses. With the experience gained and basic research and development work done in building the proof-of-concept system, we believe that a full system can be constructed to realize the potential of such a system. As time goes by, many challenging issues will be resolved. We also believe that it is useful and important to collaborate with other organizations (e.g., user organizations and organizations that own data, e.g., the Coast Guard or Piers) to build the system and thus to reduce the effort and the time in the process. Due to the potential high impact and the ever-changing world of information and technology, and to make this project truly successful, it perhaps should be transformed into a private business with an initial funding from the public or private sector. We are very optimistic that the goal is achievable. The resulting system at the global scale will positively influence our living environments, businesses and society as a whole.

## Publications

Nitin Jindal and Bing Liu. Matching Trees with Nested Lists for Web Data Extraction. Technical Report (to be submitted as a research paper), Department of Computer Science, University of Illinois at Chicago, 2009.

William N. Underwood. Information Synthesis from Web Pages. MS Thesis. Department of Computer Science, University of Illinois at Chicago, 2009.

Yi Zhang and Bing Liu. Semantic Text Classification of Disease Reporting. To appear in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research &*

*Development on Information Retrieval (SIGIR-07), 2007.*

Yi Zhang and Bing Liu. Semantic Text Classification of Emergent Disease Reports. Submitted to *the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.

Yi Zhang and Bing Liu. Extracting Locations of Disease Outbreaks in News Articles. Technical Report, 2008.

Yi Zhang, Extracting Disease Outbreaks in News Articles. PhD thesis, July, 2008 (completed). Department of Computer Science, University of Illinois at Chicago.

Three MS project reports have also been completed at Department of Computer Science, University of Illinois at Chicago.

## References

- [1] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *The 18th International Conference on Machine Learning*, 2001.
- [2] D. Lin and P. Pantel. *Discovery of Inference Rules for Question Answering*. Nat. Lang. Eng., vol.7-4, 2001.
- [3] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*. Springer, 2007.
- [4] N. Jindal and B. Liu. Matching Trees with Nested Lists for Web Data Extraction. Technical Report (to be submitted as a research paper), 2009.
- [5] ProMED-mail. <<http://www.promedmail.org>>, 2007.
- [6] D. K. Subramanian. Ship Data Extraction and Search. MS project report, Department of Computer Science, University of Illinois at Chicago, 2007.
- [7] Y. Zhang and B. Liu. Semantic Text Classification of Disease Reporting. To appear in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval (SIGIR-07)*, 2007.
- [8] Y. Zhang and B. Liu. Semantic Text Classification of Emergent Disease Reports. Submitted to *the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.
- [9] Y. Zhang and B. Liu. Extracting Locations of Disease Outbreaks in News Articles. Technical Report, 2008.
- [10] Y. Zhang, Extracting Disease Outbreaks in News Articles. PhD thesis, July, 2008 (completed).

## Appendix

### The System User Manual

The user interface facilitates mapping of four major types of entities: disease outbreaks, ports, ships, and aggregate ship traffic.

#### Disease Outbreaks

Clicking the “Get Outbreaks” button will map all detected disease outbreaks within the date window specified.

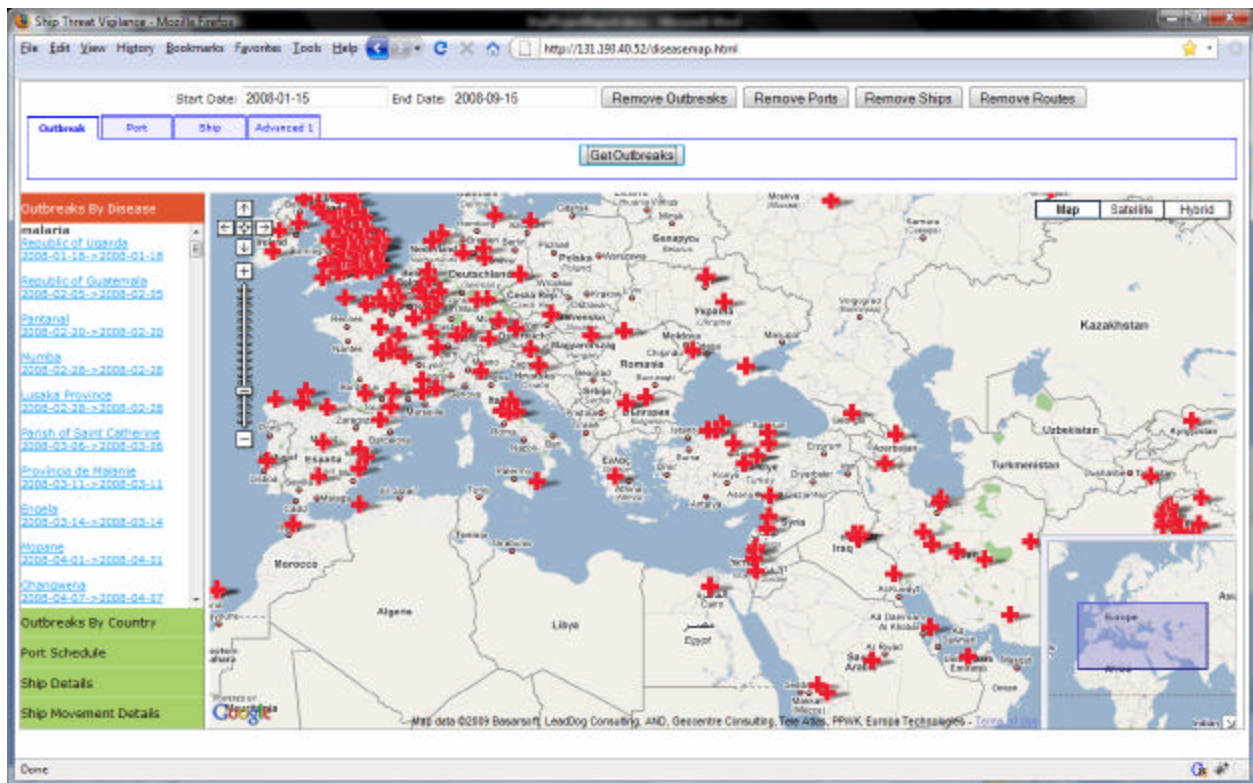


Figure 1. Mapped disease outbreaks.

Users can click each plotted point to view details about the outbreak including disease type, date, location, and a link to the news report. Alternate views of the outbreaks are provided in the left navigation, where users may view outbreaks grouped by disease type or by country.

Mapped outbreaks can be removed by clicking the “Remove Outbreaks” button at the top of the page.

#### Ports

Using the two buttons displayed under the Port tab, users can plot all ports in the system or all ports with associated schedule data.



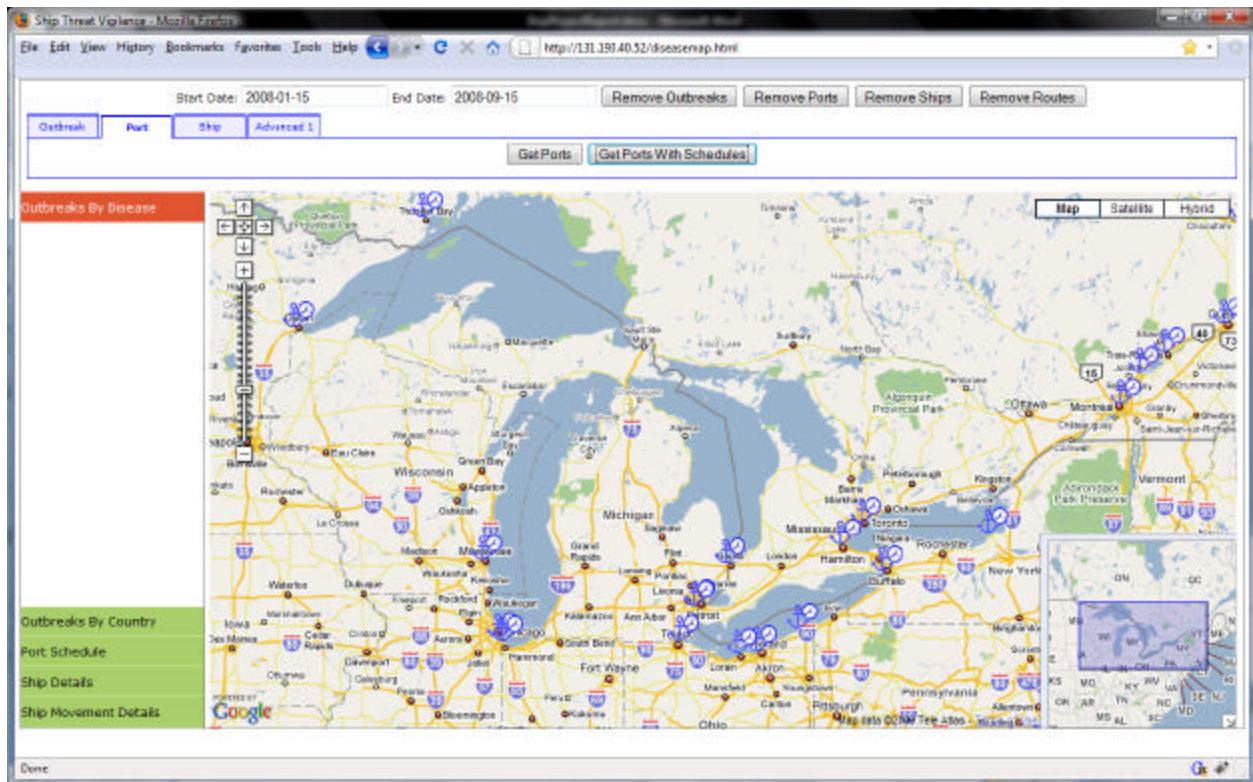


Figure 2. Plotted ports with schedules.

Clicking a port icon will display the name, ID, and location of the selected port. If the port has a webpage, a link will also be shown. The displayed balloon will also include a “Get Schedule” link which will populate the “Port Schedule” portion of the left sidebar with all ships scheduled to dock at the selected port at some point within the defined date window. Users are given the option to plot the route of each ship scheduled to dock at the port, which will include all past and future scheduled docking points for the ship along with detected ship movements.

Ports can be removed from the map by clicking the “Remove Ports” button at the top of the screen.

### **Ships**

The “Ship” tab offers several ways to search and display ships on the map. Individual ships can be located either by call sign or ship name. Typing the first few characters of a call sign or ship name will produce an auto-suggest list of possible ship matches, aiding in search ease and accuracy. After selecting a ship and clicking the “Search” button, all observed ship movements within the given date range will be plotted on the map. The date of each point can be viewed by clicking a location marker. In addition, the list of all locations in the date range can be viewed under the “Ship Movement Details” section of the left sidebar. Clicking “Follow Ship” in this panel will automatically trace through each data point at an interval of one second per point.

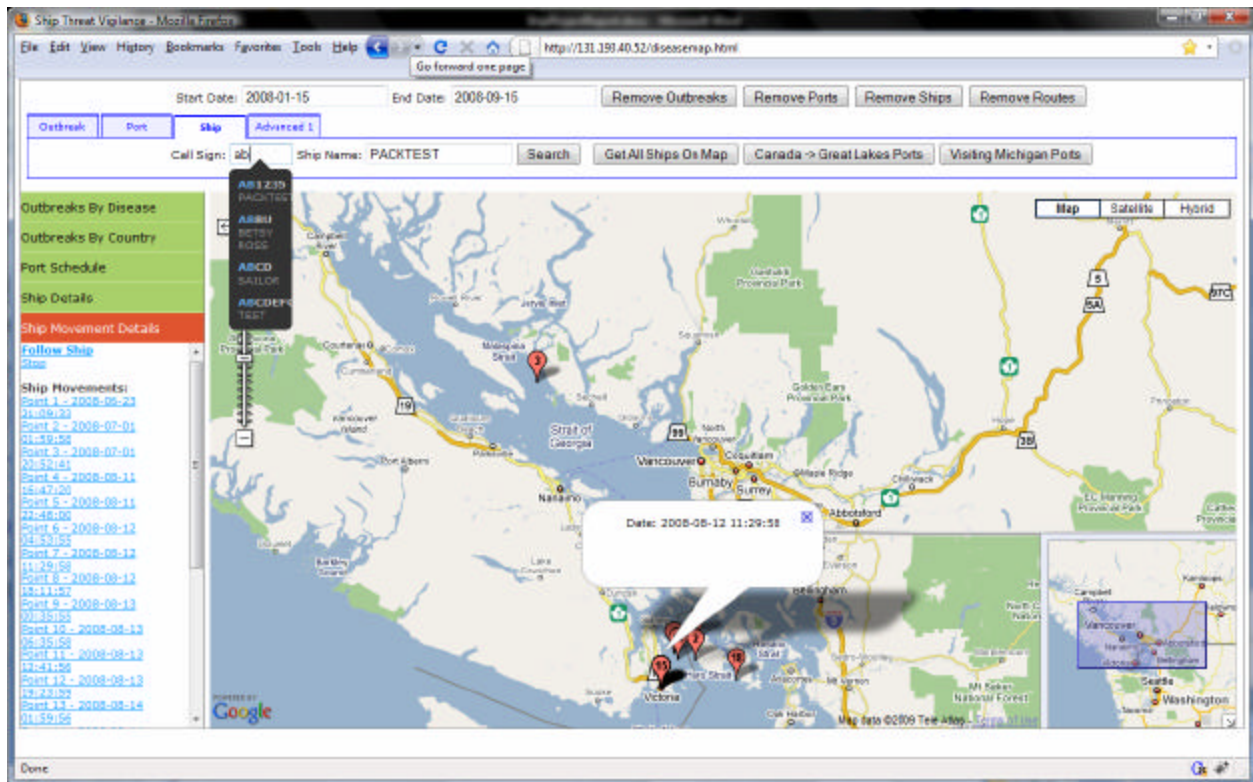


Figure 3. Ship search and route mapping.

“Get All Ships On Map” will plot all ships traveling within the bounds of the displayed map window for the specified date range. In cases where the ship appeared multiple times in that location and time window, the most recent point will be plotted. Clicking a ship marker displays the ship call sign, name, and a link to plot the overall route for the ship within the designated date range.

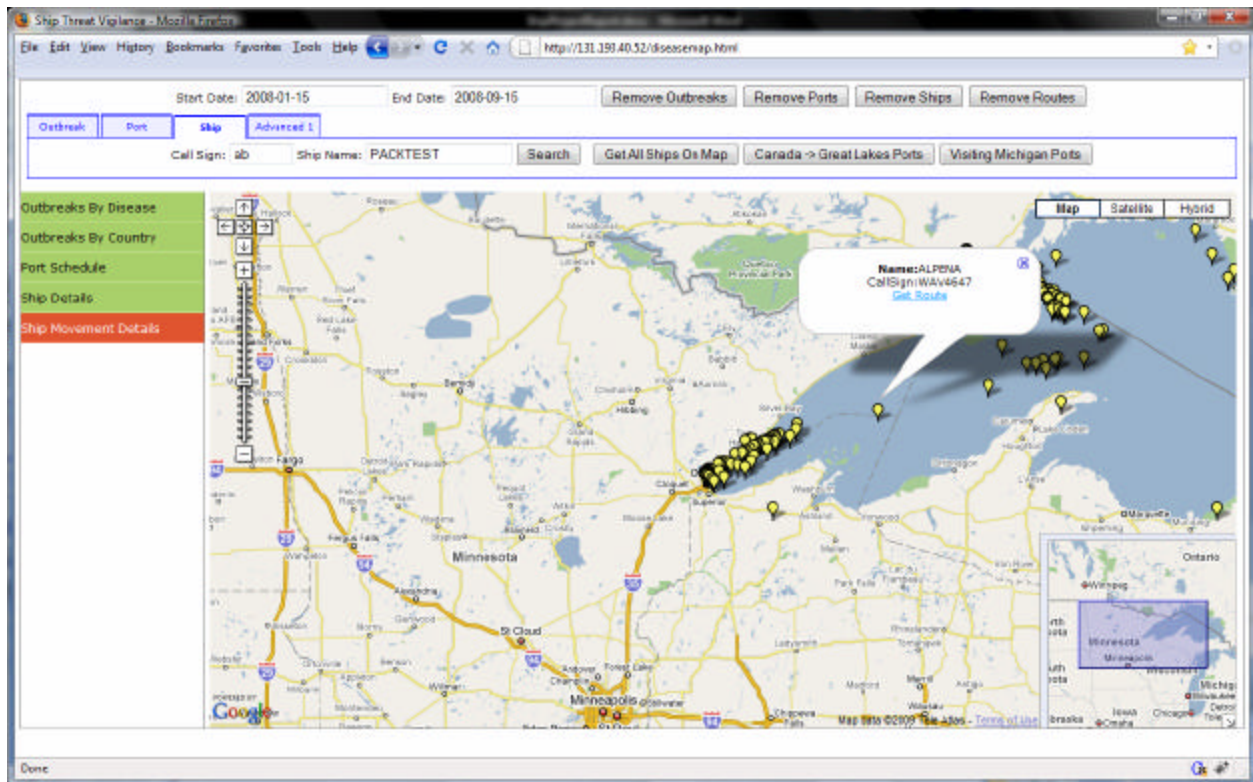


Figure 4. Plotting multiple ships.

Two other buttons are provided for displaying specific sets of ships. The “Canada → Great Lakes Ports” plots all ships that were scheduled to dock in a Canadian port prior to a Great Lakes port. The “Visiting Michigan Ports” displays all ships scheduled to dock in a Michigan port. Ships are marked at their most recently observed point.

### Advanced Search

The “Advanced 1” tab provides a mechanism for viewing all ships visiting specific Great Lakes ports along with the traffic of these ships within a specified timeframe prior to visiting the selected port. “Visiting” a port can be defined either by proximity to the port (1, 3, or 5 miles) or by a scheduled docking at the port. Selecting a port from the “select a port” drop-down list will plot the most recent proximate (or docked) points for all ships visiting the selected port. It will also populate a second drop-down list with the names of all retrieved ships. Selecting one of these ships from the drop-down list will plot the observed route of the ship.

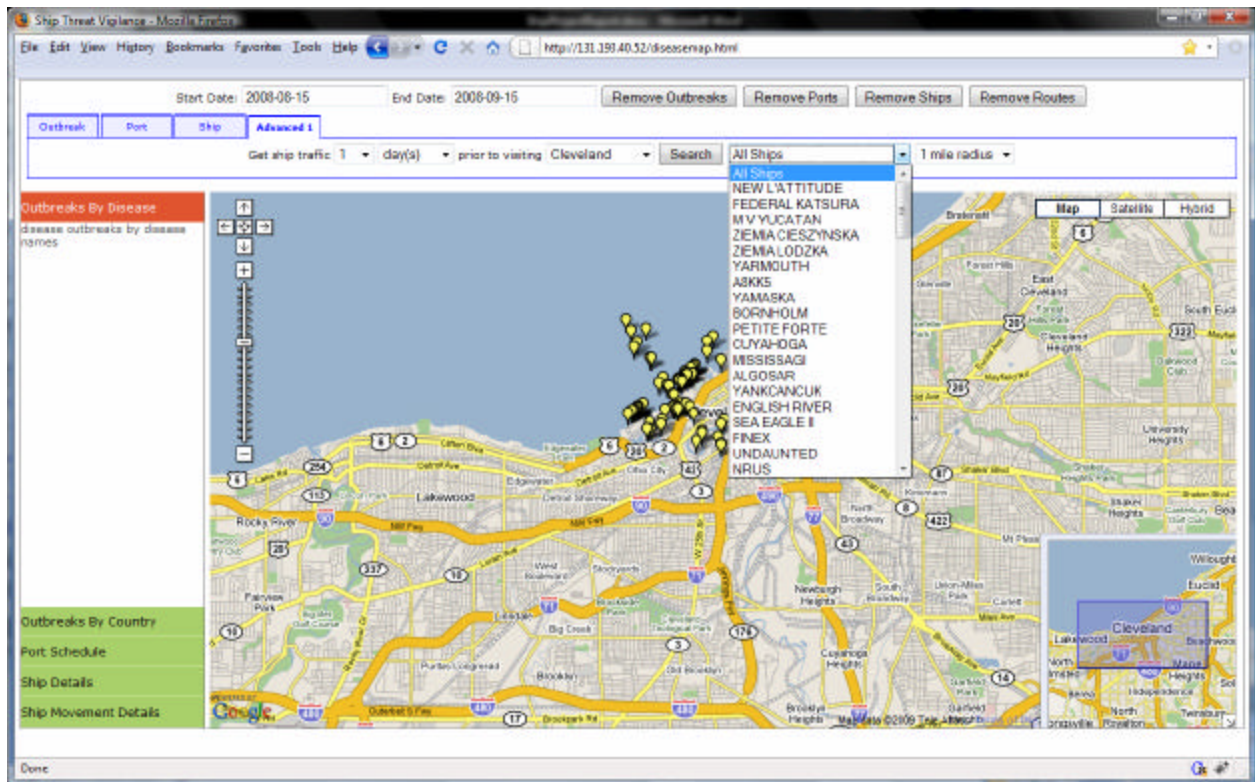


Figure 5. Ships traveling within 1 mile of the port of Cleveland.

Once a port has been selected, prior traffic can be plotted by selecting a timeframe (1-12 days, weeks, or months) and clicking the “Search” button. Observed points are also restricted by the selected start and end dates at the top of the screen.

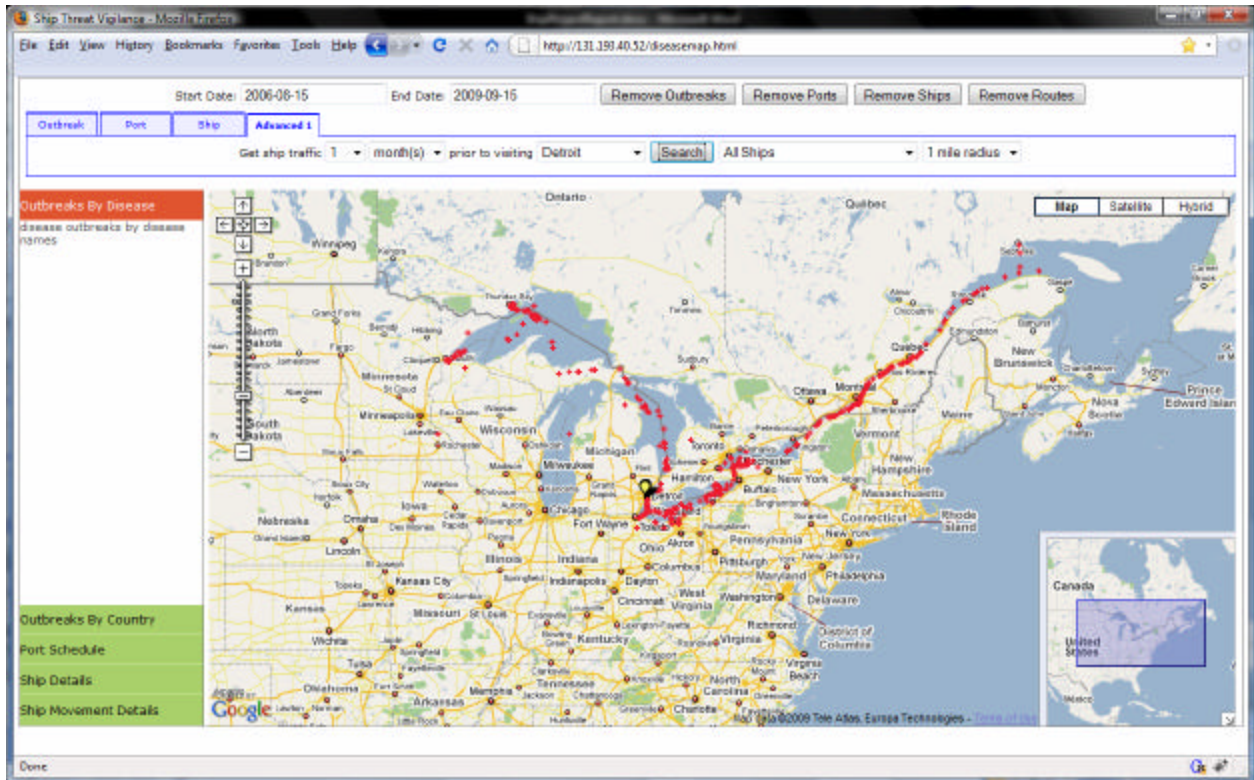


Figure 6. Ship traffic prior to visiting the port of Detroit.

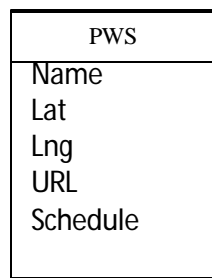
## Database Schemas for Tables

### *Port Schema Design*

Ports with schedule

Field	Type	Null	Key	Default	Extra
lat	float	YES		NULL	
lng	float	YES		NULL	
name	varchar(100)	YES		NULL	
url	varchar(500)	YES		NULL	
schedule	varchar(500)	YES		NULL	

### *Class Diagram*

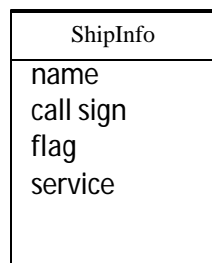


### *Ship Info Schema Design*

shipinfo

Field	Type	Null	Key	Default	Extra
name	varchar(100)	YES		NULL	
call sign	varchar(100)	YES		NULL	
flag	varchar(100)	YES		NULL	
service	varchar(100)	YES		NULL	

### *Class Diagram*

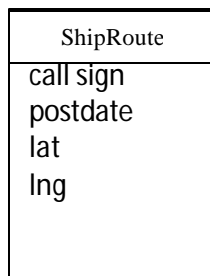


### Ship Route Schema Design

shiproute

Field	Type	Null	Key	Default	Extra
call sign	varchar(100)	NO	PRI		
postdate	datetime	NO	PRI	0000-00-00 00:00:00	
lat	float	YES		NULL	
lng	float	YES		NULL	

### Class Diagram



### Schedule Schema Design

shipsinport

Field	Type	Null	Key	Default	Extra
portname	varchar(100)	YES		NULL	
shipname	varchar(100)	YES		NULL	
call sign	varchar(100)	YES		NULL	

### Class Diagram



*Overall Database Design*

