

# Search'M – Search and Map

---



## Globalize Your Searches

*Search and view news that is important to you and others around the globe*

## Final Project Report

---

Krista Bacungan  
April 19, 2011

# Table of Contents

<b>Project Overview</b>	<b>2</b>
Summary	2
Problem	2
Approach	3
<b>Technical Design</b>	<b>4</b>
Overview	4
Implementation	5
User Interface	5
Application Controller	8
Backend Logic	9
News Indexer	9
Search Engine	10
Settings Manager	11
Preferences Files	11
Database Interface Module	11
Testing	11
<b>Project Reflection</b>	<b>12</b>
Overview	12
Challenges	12
<b>Lessons Learned</b>	<b>13</b>
<b>Future Work</b>	<b>13</b>

## Project Overview

---

### Summary

Search'M is a globalized news search engine, stressing the importance of understanding the trends in the news around the world. Search'M provides an RSS feed aggregator, search engine, and different visualizations for search results either in a traditional search result list or on a globe (Google Earth). Users can see where articles are located and how they relate to other articles by viewing their results on the globe. Search'M is a new way to browse and walk through one's news. It bridges the gap between local and global news by relating current events.

### Problem

According to the Newspaper Association of America, around 165.6 million American adults read the news (print and digital) every week. 53% of American adults get their news online and 65% of those adults use more than one news source to get their daily news. Visiting plenty of sites in order to get one's daily news can be tedious. A solution to this is a technology called Really Simple Syndication Feeds (RSS Feeds). Subscribing to an RSS Feed provides a user with a way to be alerted that a website has been updated and it is easily accessed through their Internet browser. Many sites provide multiple RSS Feeds, each for a different subject or topic (ex. CNN has a RSS Feed for politics, world news, business, art, ect.).

Search'M targets people who read their news online from news sources that provide RSS Feeds (almost all popular news sources and many independent sources). This application is aimed for personal as well as business use. Anyone that is interested in trends of topics in the news, including advertising agencies, business corporations, research facilities, students, sports fans, and many others will find this application to be a great tools.

There are many RSS aggregators available for free online and Search'M does not intend to duplicate or replace those services. Rather, Search'M goes one step

further than those applications and provides RSS Feed aggregator with features such as advanced search, in-depth automatic geotagging, the option to use a map or globe to view the results on, and the ability to explore relationships between articles in a visual manner. Since Google Earth is being used it is very likely that Search'M will appeal and pull in users that use or are interested in Google Earth.

### **Approach**

Search'M works by having the user specify RSS feed URLs from various news providers that they want to monitor. The articles from each feed are automatically parsed for metadata and content. Separate steps determine the primary location of an article for geotagging and generate keywords for that article. Geotagging is used for searching and viewing the results in Google Earth. Article keywords are used to determine how similar articles are to each other. The user can then search by certain filters (keywords, locations, dates) and view the results in a traditional search result list or visually in Google Earth. Viewing in Google Earth displays those news results and it's corresponding information on a globe in their respected geotagged locations. Through this interface, users can explore and determine the relationships between news articles from around the world.

When viewing the results on the globe, each article is represented as a place mark where the color depicts what topic that article is about and lines between articles tell the user that they are related some how. The relations between articles depend on the article's topic and content. In order to implement these features, each article is for it's metadata, primary location, topic, keywords, and how it is related to the other articles in the system.

Users can fully personalize which news is important to them through choosing RSS feeds from many different news sources. Being able to determine how they want to search for their news, choose which way to want to view their results, as well as decide how they want to see the relationships between articles and other options for customization that are available. Search'M gives the user both a purely personalized user experience as well as a globalized way of viewing their news.

# Technical Design

---

## Overview

### Use Case

Only one type of user – General user and own administrator

- Add RSS Feeds to monitor
- Search for articles through filters
  - View results in result list
  - View results in Google Earth
    - See relations between articles
    - View description on why these articles are related
  - View full articles (Click on URL)
- Manage Search'M options and preferences

### Operating Platform

- Front end
  - HTML, CSS, Javascript
  - Javascript embedded Google Earth<sup>1</sup>
- Back end
  - Java
  - JDBC<sup>2</sup>
  - MySQL<sup>3</sup> database
  - Apache Tomcat<sup>4</sup>
- Google Code Repository and Bug Tracker<sup>5</sup>

### Design Pattern

- Strict model view controller (MVC) design pattern

---

<sup>1</sup> <http://www.google.com/earth/index.html>

<sup>2</sup> [http:// dev.mysql.com/usingmysql/java/](http://dev.mysql.com/usingmysql/java/)

<sup>3</sup> <http://www.mysql.com/>

<sup>4</sup> <http://tomcat.apache.org/>

<sup>5</sup> <http://code.google.com/p/searchnmap/>

- Model – Back end logic (news indexer, search engine, settings manager)
- View - Front end (user interface, Java Servlet, Google Earth)
- Controller – Application controller
- Resources used by back end (controller and model)
  - Preferences files stored on disk
  - Database interface module
  - MySQL database
  - Apache Tomcat Server

User only interacts with the front end of the application.

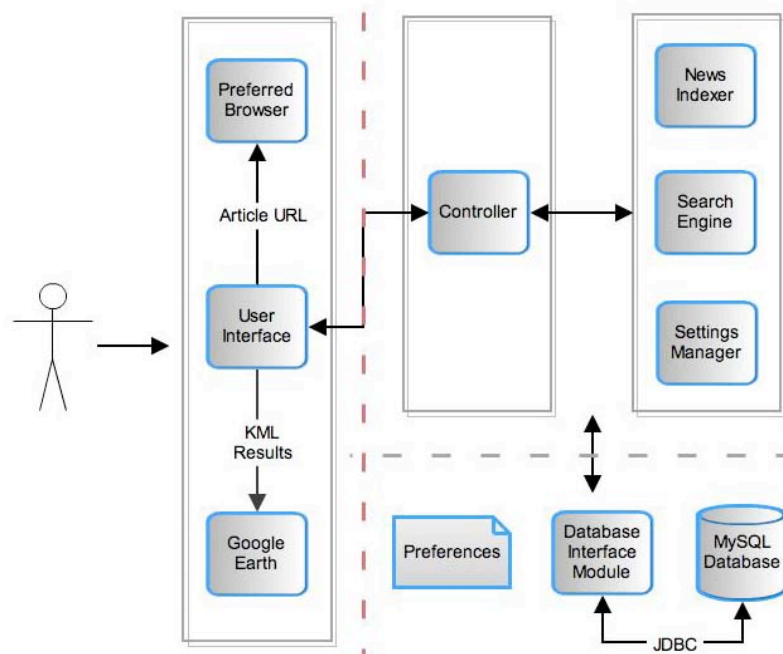


Figure 1 – Design Diagram

## Implementation

### User Interface

Search'M application runs on a Tomcat Server as a Java Servlet and is opened by typing in the servlet URL into a preferred Internet browser. Here, the user sees the embedded globe where geotagged results get displayed, the options for

viewing and hiding features on the globe (Fig. 3), the search box, and the traditional result list for all results, even non-geotagged articles.

This GUI must be extremely user friendly and intuitive. The design is sleek and minimalist. The traditional result list is populated by generated XML and the globe is populated by generated KML for each search. Both the results on the globe (Fig. 5) and in the result list display (Fig. 2) the following information: title, author, URL, data published, article summary, categories, keywords, location, and thumbnails of images.

There are three types of KML that are generated for each search: article KML, category KML, and relational KML. The article KML (Fig. 4) creates place marks for each article at their primary geotagged location. Each category of articles (ie. Politics, world news, fashion, ect.) have a different color for it's place mark. When the place mark is clicked on, the article information described above is displayed in a text bubble. The category KML (Fig. 6) displays lines that connect all articles of the same category and the color of the lines is the same as the specified color for that category. Lastly, the relational KML (Fig. 7) connects articles that are related to each other. The thickness of the line depicts how related they are (more related, thicker lines). The user has the options of displaying and hiding the category KML lines and the relational KML lines.

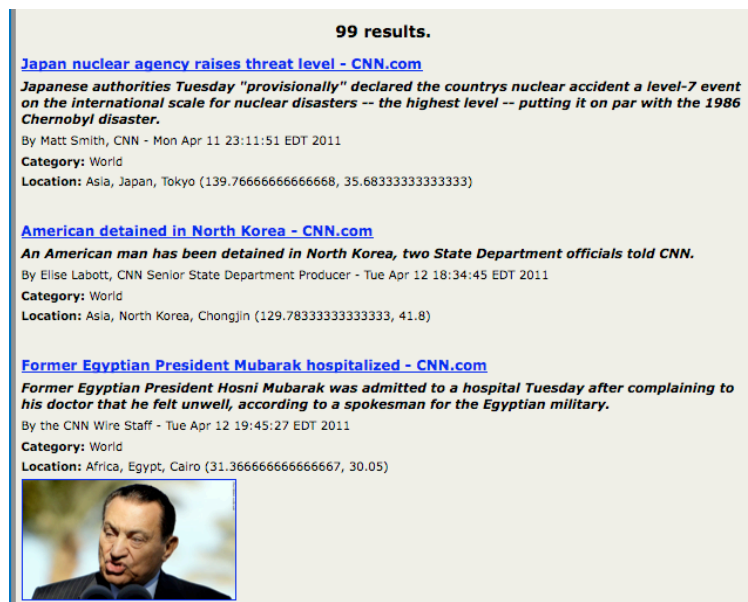


Figure 2 – Search result list

### Google Earth Options

Show Article Relations:  
Show All Categories:

World News:  
Politics:  
Travel:

Health:  
Crime:

### Search News Articles

Title: 
Author:

Date: 
Location:

Content: 
Category:

Figure 3 – Options for globe and search box

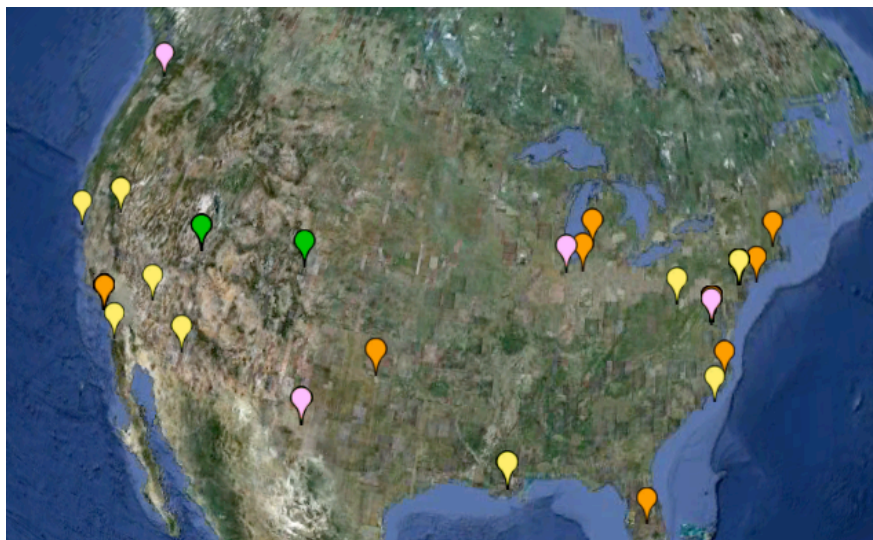


Figure 4 – Article KML




**Frontier Airlines reduces fees - CNN.com**  
[http://rss.cnn.com/~r/rss/cnn\\_travel/~3/bvAzafKEDRQ/index.html](http://rss.cnn.com/~r/rss/cnn_travel/~3/bvAzafKEDRQ/index.html)  
 Authors:By  A. Pawlowski, CNN  
 Wed Apr 13 12:29:57 EDT 2011  
**Air travelers are used to airlines raising or adding fees, so they may be shocked to hear that one carrier is actually reducing some of them.**  
 Category: Travel  
 Location: North America, United States, Colorado, CO, Denver



Figure 5 – Placemark pop-up text bubble for an article



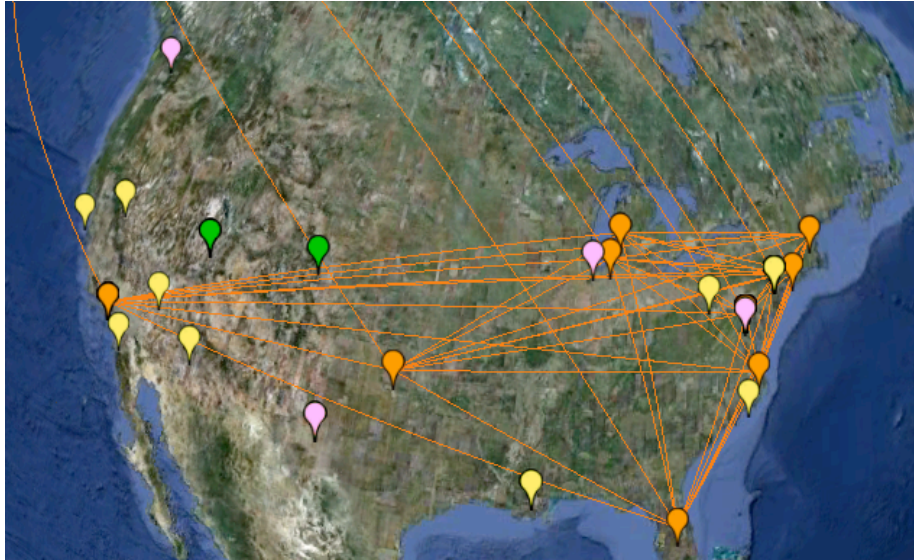


Figure 6 – Category KML

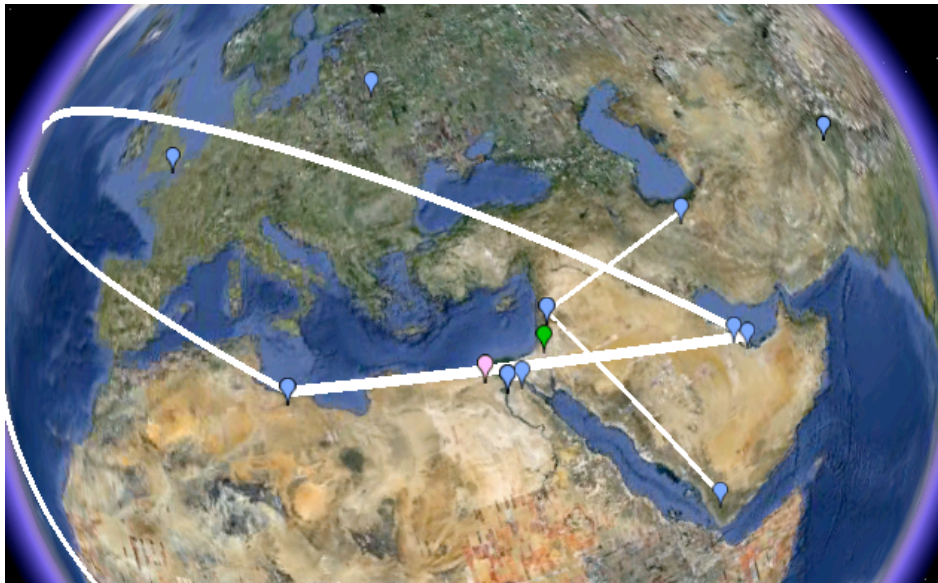


Figure 7 – Relational KML

## Application Controller

The Application Controller is the Java Servlet decouples communication between User Interface, and back end logic. It handles all the events and updates from the User Interface, as well as returns the search results from the back end logic to the front end interface. In starting up, the controller loads in all user preferences that were stored or any default preferences. It also checks the status of the application database in case files have been corrupted or deleted. In that case, the controller resets tables and restores the files. If stored articles are expired (they

have been in the system longer than specified by the user), the controller purges them from the system.

## Backend Logic

### *News Indexer*

#### RSS Feed Module

This module handles all RSS feed processing including adding, editing, and deleting monitored RSS feeds. The ROME API<sup>6</sup> provides methods for parsing each RSS feed for individual articles. Only articles that are within the user specified time range are indexed. The user specifies this time range within the Search'M preferences. Since multiple RSS feeds are monitored at the same time, a thread pool design pattern is used to handle the concurrency.

#### Article Processing Module

This module handles the processing for each individual article. The Jericho API<sup>7</sup> provides methods for extracting the article metadata via tags in the HTML. Article metadata includes the title, authors, date published, and categories. The article content needs to be parsed out from the HTML so that it does not include external advertisements, ect. This is done with Java SAX<sup>8</sup> along with extracting the images from the article.

To determine the main location of an article, a location hierarchy tree (Fig. 3) is used to decide which coordinates are to be geotagged. This location hierarchy tree is stored in the Search'M database with the name of all the popular locations in the world mapped to their coordinates. If the article does not contain a city, but contains a broader location, the coordinate of the first child in that branch is geotagged. Example: Article contains United States - the coordinate 385342N 770212W is tagged. The biggest complication with this algorithm occurs when multiple locations are present in an article or when nouns are ambiguous with location names. In order to handle these situations, the position of the location in

---

<sup>6</sup> <https://rome.dev.java.net/>

<sup>7</sup> <http://jericho.htmlparser.net/docs/index.html>

<sup>8</sup> <http://www.saxproject.org/>

the article text and the frequency of that location are used to determine the primary location.

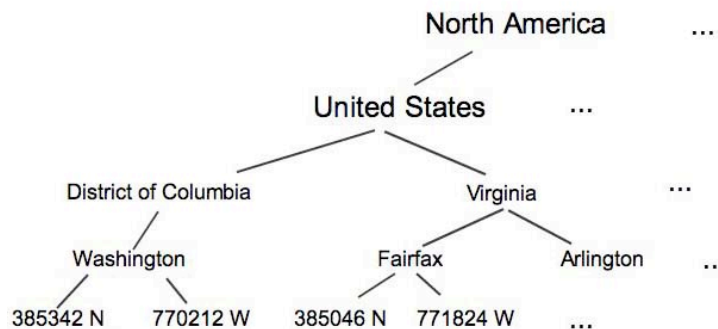


Figure 7 – Location hierarchy tree

In order to compare articles, Search'M needs to determine the keywords of each article. Search'M utilizes the Stanford Natural Language Part-Of-Speech Tagger<sup>9</sup> to find all of the nouns in an article. Each noun is given a relevance percent that tells us how important that noun is to that article. Taking the frequency of each noun and dividing by the sum of nouns in the article finds the relevance percent. Then the nouns and their relevance percents of two articles are compared to relate the article. If both articles have a certain noun, those noun's relevance percents are averaged and added to the similarity score for those two articles. That final similarity score is used in displaying the relations between articles.

## Search Engine

### Search Module

When the user performs a search, the parameters from that search are fed to the Search Module. The Search Module uses the Apache Lucene API for quick and efficient searching. By communicating with Database Interface Module, Search'M can query for results.

### Results Module

The raw results are handed from the Search Module to the Results Module. This module generates the result KML to be used in the front end of the application.

---

<sup>9</sup> <http://nlp.stanford.edu/software/tagger.shtml>

Streaming the generated XML and KML to the front end of the application saves I/O resources. All relational KML needs to be generated and sent to the front end at the same time. This ensures that the relational KML for a newly selected option does not dynamically get created, saving the user interface from lagging..

### ***Settings Manager***

All user preferences and settings are handled by the Settings Manager component. The setting manager relays any changes in the user preferences to the other back end components. The ideal settings that can be set include deleting and modifying RSS feed URLs, editing how long articles stay in the system, editing how far back in time articles should be archived, pausing news indexing, selecting the preferred browser, and resetting application. For the current release of Search'M these features were not developed.

### ***Preferences Files***

Preferences files include the application default settings, user specified settings, database table statements, database backup files, and location hierarchy tree. These files are loaded at the startup of the application via the Application Controller.

### ***Database Interface Module***

This module connects to the MySQL database through JDBC. All MySQL statements are stored in this module allowing the other components to access the database for obtaining or storing information. An article data access object is found in this module. This DAO provides components with a persistent object that encapsulates all the information about a specific article. This significantly lessens the number of direct queries to the database, making the application faster.

## **Testing**

Search'M has in-line source code tests during compile-time and run-time for each module of the application. This ensures that the each level of the program architecture is monitored. An elaborate logging system is in place to help with

debugging run-time issues. With these test mechanisms in place, all of the functional specifications can be tested and confirms the usability of the application for demo purpose.

The logging system gets created in the Application Controller and logs out to disk in the Search'M folder. There is a separate log file for each run of the application and is named with the date and time to distinguish the files. The logging system being used is the Java `java.util.logging.Logger` package. Logging is done in every single source file of the application to ensure that nothing is being overlooked. The logs include both errors and success messages to provide a detailed run through of what has been done.

## Project Reflection

---

### Overview

Planning and development for this project started in September 2010. Reading in RSS feeds, reading individual articles and retrieving their metadata, basic KML and XML search results, and the search components were completed by December. By January, the algorithm for geotagging articles was developed. The Java Servlet and user interface was finalized in March. Relating articles, viewing the relational KML on Google Earth, and finalizing the code was delivered mid-April.

### Challenges

Challenges that occurred during the development of this project included working with unstructured news articles, deciphering RSS Feeds, seamlessly interfacing with Google Earth, determining the primary location of an article, keyword generation, and relating articles to each other. Each news provider structures their website and articles differently. An algorithm was developed so that Search'M can find an article's metadata no matter the structure of the website as well as an algorithm to decipher RSS Feeds from different sites.

Originally the user interface included three separate modules: the Search'M interface built from Java Swing, the Google Earth application, and the user's preferred Internet browser. This proved to be very unsightly and tedious. Now, Search'M is opened in the users preferred browser as a web service and Google Earth is embedded in the same screen as the Search'M interface using Javascript.

The biggest challenges were geotagging articles, generating the keywords, and comparing two articles to each other. The algorithms for these challenges were described in the Implementation section. These challenges took up most of development from January to April. A lot of research had to be conducted in finding algorithms that were efficient and exactly what Search'M needed

## Lessons Learned

---

During the Fall semester, the boot camp phase and beginning development on Search'M went smoothly. Most of the time deliverables were accomplished early and demos excelled expectations for that deadline. However, it is evident that during the Spring semester, time management was not as scheduled as it should have been. Requirements were met for the final demonstration, but Search'M is not at the level that it was planned to be at. A good lesson is to always get ahead of schedule whenever you can so you can balance the effects when you fall behind later.

Another lesson is to talk to other classmates and research deep into existing products or similar products. If this was carried out before, a lot of time would have been saved and some decisions made for Search'M would be different.

## Future Work

---

Development on Search'M will not stop here and it will eventually be released as a free product. With more time, features that are missing from the current release of Search'M will be developed. This includes: developing and

perfecting the algorithm for determining how related articles are, allowing for users to correct incorrectly tagged articles or generated keywords, implementation of a machine learning algorithm so the accuracy of the generated keywords and relations between articles will be improved, and implementing the missing preference features.

Search'M can be extended to include media other than news. Blogs, images, desktop documents, and other files can be indexed and processed similarly to the news articles being used now with different types of tagging