Krista Bacungan

Search'M – Functional Specifications and Use Cases

Search'M is an application that allows users to search and view news that is important to them. The search results can be viewed in different representations, allowing the user to see how the news they are interested in has affected the rest of the world. It 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 processed for categories, images, related articles' URLs, and a description. Separate steps are taken to determine the keywords and locations in an article's text versus the articles metadata. The program then are stores the information in the system's database.

The user searches through their news using keywords, locations, dates, authors, categories, or a combination of those options. The search results are displayed in a search result list containing each result's title, URL to the article, authors, date, and thumbnails of images in the article. The user can to view the results in Google Earth by clicking on the earth icon in the search result list. This is a feature users will use so that they can view the news based on its location around the globe and see the relationship of that new article to others. For each search result, a placemark is displayed at the location that the article identifies with along with its corresponding information. Hovering over a placemark displays lines that connect related articles together, representing similar keywords, categories, dates, ect.

All articles can be viewed fully in a browser by clicking on the article URL. Users can modify what RSS feeds are being monitored and have the option to clear all the stored data. This system can be used whether or not connected to the Internet. If offline, new articles cannot be processed but all other features remain. Once the system is connected to the internet again, Search'M will start its RSS feed ingest again.

Core Requirements

- Parse RSS feeds
    - Retrieve RSS stream from RSS feed URLs.
    - Parse through HTML of each article, process text, and store discovered keywords, locations, dates, and categories to be associated with article.
- Search
    - By keywords, dates, categories, and locations, or a combination of filters.
- Display results

    - In a result list (designed like commonly used search engine results).
    - In Google Earth using KML.
        - Each result is a placemark pin.
        - Connecting lines between article placemarks represents that the connected articles are related.
- Manage Search'M
    - Add, delete, or modify RSS feed URLs.
    - Clear system of all data.
- Use Search'M
    - Connected to the internet
        - All features able to run
    - Disconnected from the internet
        - All features able to run except no new articles will be added to the system

List of Functions:

- Process articles
    - Determine which sections of the HTML is the article content (TagSoup API and Java SAX API).
    - Get article dates, authors, categories, and automated keywords from article's metadata.
    - Process the article content for keywords and locations.
    - Pick out embedded images.
        - Create thumbnails.
- Retrieve articles from RSS Feed
    - Stream articles one at a time and read HTML (ROME API).
    - Automatically pull in articles as they are updated to the RSS feed.
- Create results
    - Determine relationships between articles. This is not the same as related articles which are URLs found in the article. Instead this subfunction must connect articles together based on common dates, locations, keywords, or categories.
    - Create KML to show results in Google Earth (Java SAX API).
    - Create KML to show the relating lines between articles. (Java SAX API)
- Store and retrieve information from database

- Create and design database (Java Database Connectivity, JDBC).
- Store the article content, authors, description, thumbnail of embedded images, keywords, categories, URLs of related articles, dates, locations, and URL of the article.
- Create efficient queries to get results for searches.

Use Cases:

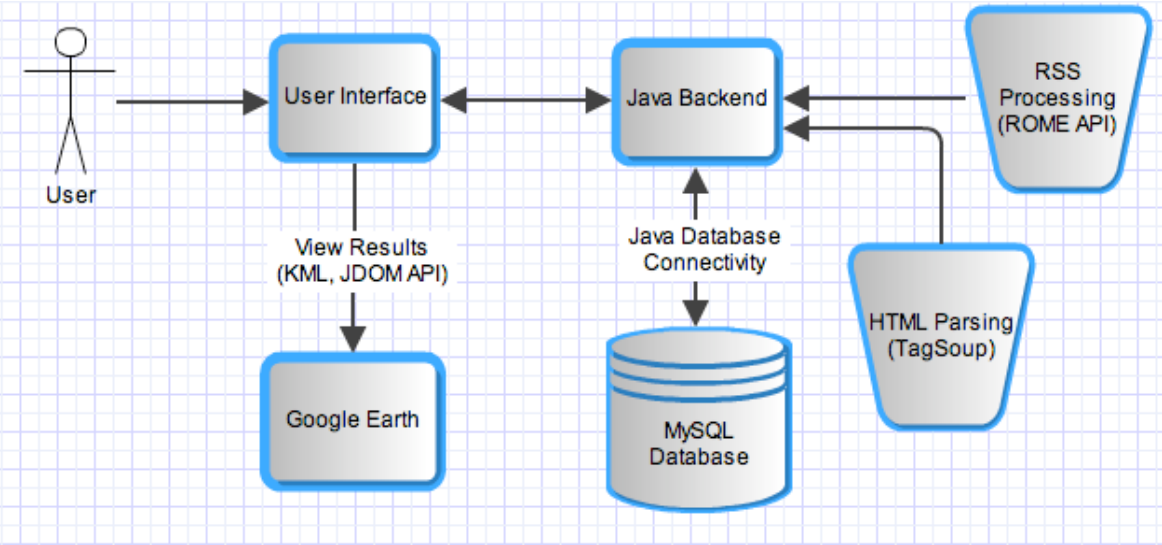
- General Users
  - Select RSS feeds to monitor.
  - Search the system by keyword, category, date, location, author, or a combination of filters.
  - View results in result list
    - View results in Google Earth (Click on icon).
      - View related article paths.
  - View full articles (Click on URL).
  - Clear RSS feeds and database.
  - Add, delete, or update RSS feed URLs.
  - Use system on or off the internet.

Data Structure:

This application is developed in Java and interfaces with many external libraries and APIs. The ROME API is being called to read in RSS feed URLs and retrieve the articles that are represented in that feed. Then Jericho HTML Parser API and Java SAX API are used to parse the HTML for each article. This parsing also grabs the information for each article such as dates, authors, categories, images, and the body of the article. All of that information is being stored in a MySQL database and uses JDBC to communicate between the database and the base Java application. XAMPP is being used to manage the MySQL database. The results are displayed through Java in the user interface and in a KML file for Google Earth. The system calls the Javax.xml library to create the result KML file. There is a location hierarchy tree that is used for determine the primary location of article. This tree was created with external text files found on the internet that contained a locations city, state/province if applicable, country, and longitude and latitude coordinates. A small application reads in those files and creates an XML using the Javax.xml library to store the location data. The

front end is done in Java as a stand-alone desktop application and opens the primary internet browser when viewing article's original file. When the user wants to view the results in Google Earth, the KML file that is generated from the search results is opened in the current Google Earth browser or in a new browser if one does not exist already.

Diagram:



Algorithms:

Since keyword generation is very complex, I am still in the process of finding an appropriate algorithm. Word frequency combined with a don't-include-list of non-keyword words (such as conjunctions and common nouns) could be a possible solution. As described above, the location hierarchy tree will be used to find locations in the text. Determining the primary location of an article is a similar problem to keyword generation and will most likely use the location that is most frequent. Lastly, determining which articles are related to each can be solved by correctly designing the database to allow similarities between articles to be determined at the time of the search.

Changes in Writing-2:

Introductory paragraph was edited to make it less technical and to fix the tenses. More details were added to the data structure section since progress has been made in that area since the last writing assignment. Overall, the tense of this writing has been corrected as well as run on sentences.