Mapping of Texas A&M Library Books
Fall 2013 Group Project GEOG 489/689
Elizabeth Rentschlar, Forrest Bowlick, Noe Saenz, Jacob Wooten, Alex Roman

1 Project Purpose

Library users do not always have a clear idea of the resources they need when beginning a library search. A map based search would allow users to search by geographic location for a product in addition to the traditional subject search. By providing a map interface where users may search by the geographic location and the subject of the product, we hope to improve the user's ability to interact and discover library resources. This approach would gives users a greater spatial and hands-on interaction with library resources, more than the current library search allows. We undertook this project at the request of Kathy Weimer at the Map and GIS Library. This project is a beta test of future functionality for the library website. If the end product meets the needs of the Map and GIS library, they may use it as a general model for further adaptation of the interface before allowing general availability to library patrons.

2 Interface

Users of this system of library access will require the kml (Google Earth) file, the associated Javascript code, and the necessary HTML. The HTML code runs best in Mozilla's Firefox web-browser. While the HTML code runs within the web-browser, it is not publicly published on the internet. Using a Texas A&M inspired color scheme and user interface, the user will select their preferred base map using the interface on the right side of the screen. This area also includes the panning and zooming options to interact with the general map layout. Users may also left-click on the map to move between locations. Scrolling on the mouse wheel allows zoom in or out of the map. They also have the option of using a panning menu and the zoom bar in the upper left hand corner of the map. In addition to these two options they may select which county they are interested in to zoom to that location. For this beta version, the United States and China have the option to zoom to specific States and Provinces.

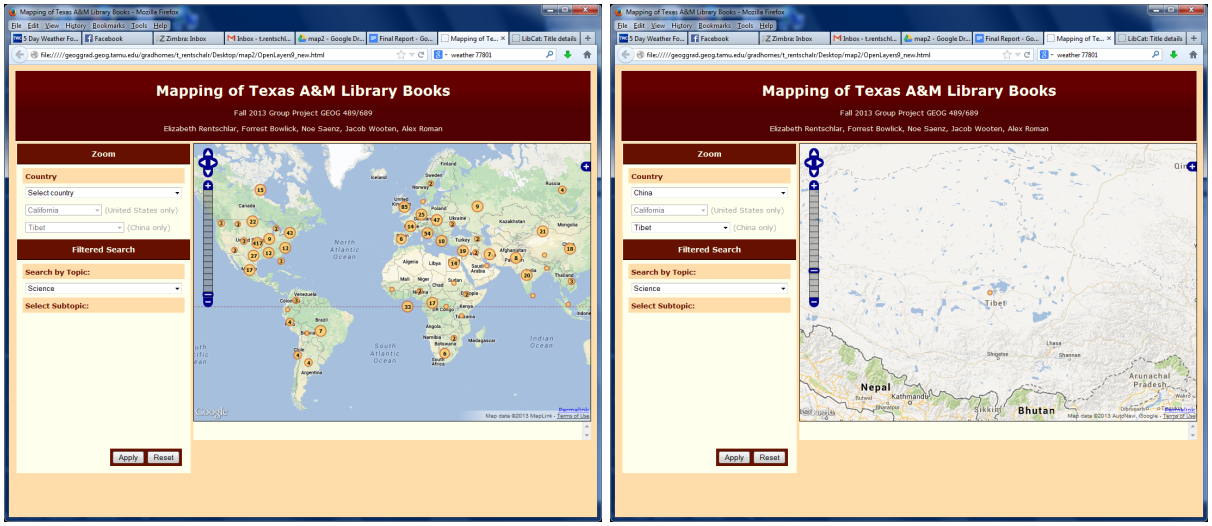

Figure One: Map Interface Displaying Books Geocoded Worldwide and in Tibet.

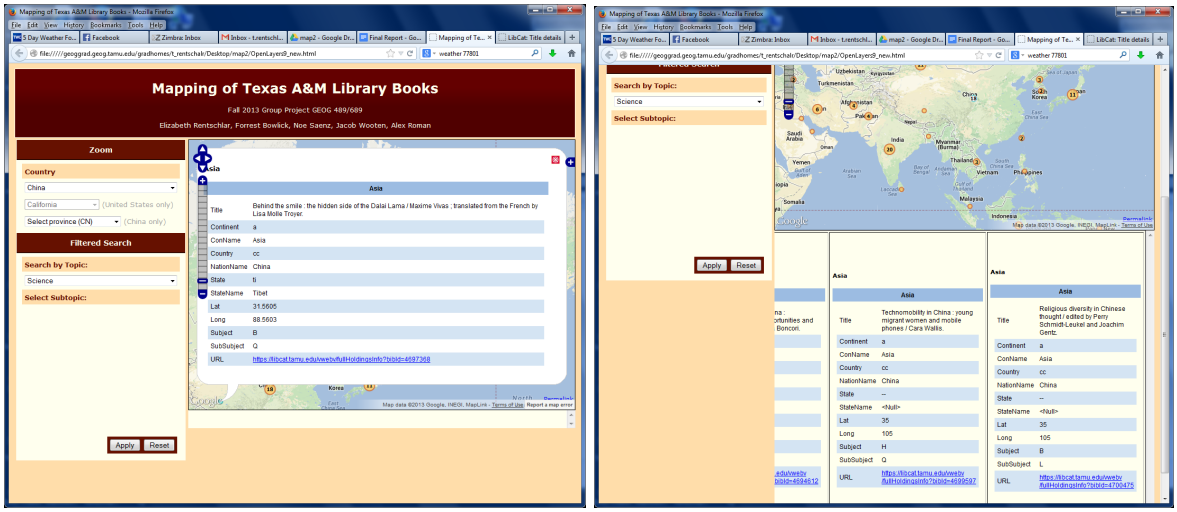

Figure Two: Details from Selected Books in the Map Interface.

In order to narrow the number of books that are displayed the user may use filters to specify which subject and sub-subject will display. However, the user must clear the filter between uses otherwise subjects will not fill in correctly. For single points, the user may click on the point to see the details associated with that selection. For a cluster of points, the user may hover the mouse over the cluster, and individual book details will display underneath the map. If there are more books than can be displayed, the user may use the pan bar at the bottom to navigate to more books. The records have a link to the main library catalog page about the book, where the user may find more details than are offered in the map. To make full use of this functionality, though, the user will already need to navigate to the library catalogue prior to clicking the links. Failing to do so will result in an error.

3 Methodology

Web scraping allowed us to obtain data from the library catalog. The library website provides a technical view that exists as library metadata, describing the various components of the library record in a manner that can be obtained via script. The Python language has an add-in named 'ScraPy' that can automatically extract structured data from websites. Since the Library metadata exists in this structured form, manipulation of ScraPy code allows us to extract the information for use in the geocoding system developed.

To begin our beta test, however, the Maps and GIS Library is the original source of our data. We acquired the data in two separate Microsoft Excel (.xls) files. The first data conversion resaved this data as tab delimited .txt files for ease of use and better manipulation in subsequent steps. We used a hard code python script to extract the elements we needed from the data. The script is dependent on the format of the input data. If the data format does not match the code format, the script will extract incorrect data. The individual values from the sample data are split using the tabs to create the index to extract the values. Because one of the extracted values is the title of the book, a coma was not an appropriate delimiter for the output text file. For this reason we chose to use tab, '\t', to delineate our output items. In order to determine the geographic location and the subject, we needed to identify specific characters from the metadata. We needed to use two indexes to extract the information of interest because it did not become separate via our split. The first index identifies which string we are trying to extract the information from, and the second index indicates which character inside the string represents our geographic locations, and which character represents our subject and sub-subject.

We chose to use 'if else' statements to produce the geocoding. This was not the most efficient means, as a dictionary would be more efficient, but the dictionary form did not undergo consideration until the 'if else' statements existed in code already. The latitudes and longitudes for the continents, nations, and states are hard coded. The 'if' statement checks if the index that is the continent matches the Library of congress code for North America. If they match, then the continent name, latitude, and longitude will be set as those appropriate for North America. The 'if' statement will also set the Nation name and the State name to null values. Next, it will go into the North American Nations 'if' statements. If the Nation indexed item from the input file matches the Library of Congress's code for the United States, the earlier null values for the Nation name, lat, and long will be overwritten with the United States information. The reason that the Nations and States were set to nulls during the continent identification is that there are not Nations and States associated with every field from the input files.

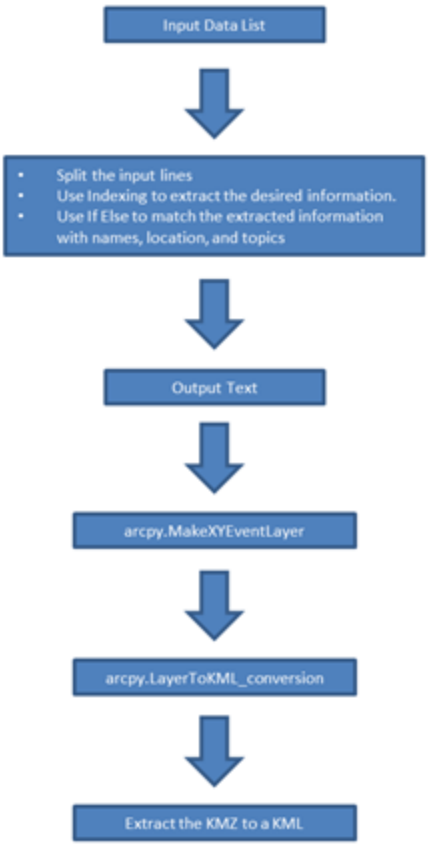

Figure Three: Workflow for Data Conversion.

The new text file will have fields for continent code, continent name, nation code, nation name, state code, state name, latitude, longitude, book topic, book subtopic, the title of the book, and the library url for the book. This is not the most efficient database design, because of the blank countries and states, but it is intuitive to work with and we will not need to run code on it often, so we were not concerned with the speed of processing. Future designs of this database may need to address this issue.

The new text file can be read by arcpy as a table, due to the prior use of a tab delineation. Using Model Builder within ArcGIS to create our python code, we then convert the table with latitude and longitude into a layer event and then convert that layer into a KMZ. This conversion uses only two tools:

arcpy.MakeXYEventLayer_management and
arcpy.LayerToKML_conversion.

Both have a large number of input variables that use ArcGIS default values. The built in file format for OpenLayers is not a KMZ but a KML, so we needed to unzip the KMZ. We imported the zipfile module to add this functionality to the code. We used the extract command which outputs a doc.kml.

We imported the os module to rename the doc. kml using the os.rename() command and so that we could delete the kmz that was created by arcpy. We had initial planned to have three different layers of point data for the Continent, Nations, and the States, but we instead opted to geocode all of our data to the smallest geography available from the metadata codes.

OpenLayers has a javascript and a html component. The html creates the shell from which the javascript runs in Mozilla Firefox. The OpenLayers Library Documentation website gave walkthrough steps regarding setup and implementation of the OpenLayers viewer html code. There are two main components to the OpenLayers API: 'Map' and 'Layer'. Once the 'Map' has been created a 'Layer' may be added. To display the 'Layer' in the 'Map', a center and zoom level must be set. There are two types of layers in OpenLayers: base layers and overlays. OpenLayers has base layers that are easily used, from recognizable sources such as Google, ArcGIS, and OpenStreetMap. The map may have as many base maps as the user would like to include, but only one may be opened at a time. We selected "Google Streets", "Google Hybrid", and "Open Street Maps" as our base layers. We only have the point overlay of the kml that was created with the python code.

We extensively used example tools from the web site to work on our javascript such as: kml-layer, zoom, filter-strategy, popup, strategy-cluster-extended, and highlight-feature. Some of the examples provide block of code that we copied while other examples only told the name of the function being applied to overlay. We have several methods of zoom available to map users, as they may use the scroll wheel on the mouse, the zoom bar present on the left of the map, or the drop down boxes that will take the user to specific Countries and to specific States or provinces in the United States and China. These zooms do not change the number of points that are accessible from the map. The zoom dimensions are based on the coordinates that surround a particular boundary. For the states, these coordinates are exports obtained from OpenStreetMaps. The coordinates for the Chinese provinces are shapefile extracts from ArcMap, via execution of the "minimum bounding geometry" tool on each province.

There are two drop down filters that limit the number of points that are displayed on the map. These filters are based on the call numbers from the original data. The main topics follow those outline by the library of congress, when the main topic is selected the sub topic infills automaticly with the appropriate sub topics. There are several topics that do not have subtopics like American History, as the call number codes work differently than they do for the other subjects. The filters are currently empty shells. The backend code gathers the user's topic and sub-topic input and compares these to all the features description fields "subject" and "sub-subject". All matching features will be returned and displayed on the map respectively.

4 Limitations

The first issue that will need to be addressed is the implementation of the filters. We will need to

complete the filters before we pass on the beta test to the Map Library because this was one of the main features that they had requested.

We did not geocode all of the possible states and provinces, Canada and Australia were not included despite having possible results. There were even a few countries that we missed in the Caribbean and Central America. The nations inside of the UK were geocoded but we did not create zooms because they are so small it seemed unnecessary. In the future it might even be possible to geocode cities inside of the states based on the titles. For example: "Seat of empire : the embattled birth of Austin, Texas / Jeffrey Stuart Kerr" could be coded as located in Austin instead of the more generic Texas. Other titles, such as: "Cultural resources investigations of portions of the proposed 40-mile Duke Energy-Wilbreeze pipeline project, Jackson and Lavaca counties, Texas / prepared by Michael N. Smith, Kevin A. Miller, and Mercedes C. Cody ; principal investigator, Kevin A. Miller", would be more difficult to spatially pinpoint. Consider, would the title be located in Jackson or Lavaca county? The limits of this beta test locked our geocoding to the location that were already outlined in the library of congress meta data.

The versatility of the code could be improved with the removal of the 'if else' statements used for geocoding and the use of dictionaries in their place. This would shorten the code significantly which would make it simpler to read and work with. Currently, the reliance on these statements increases the overall run time of the application, which for a large scale implementation of the overall script would be inadvisable.

Reading the current map can be confusing when the user is zoomed into a location like Slovakia. It appears to have over 40 associated book titles, but when the user hovers over the cluster, the books are all general European books. and not about Slovakia at all. This could be improved if there was a way to have the continent level titles fade out once the map was zoomed in to a closer level, as the scale adjustment from continent to country does not apply to the underlying country geography when displayed at the continent level. There are also titles floating in the ocean that did not have geographic codes associated with them, despite having names like "Collapse of British power" and "Open veins of Latin America : five centuries of the pillage of a continent / Eduardo Galeano ; translated by Cedric Belfrage." It is unclear if these titles should have been excluded from the kml or if we could have used another method to place their locations.

The map interface would be more aesthetically pleasing if the cluster book display started at the top instead of the bottom of the table underneath the map. This adjustment would remove the stair step effect that occurs when a book with a long title is next to a book with a short title. The authors could be pulled from the title by splitting the string at the '\' mark. The Author field could then be made into a search field of its own. Our current filter could still be improved by incorporating the subtopics of the topics that had different call number styles that were left out of the map filter. This would require reworking of the python code to differentiate between an index value that what a letter or a number.

It would be nice if opening the map automatically began a LibCat search session, as the user would not have to perform a search before the links on the map worked. This functionality would require more integration than the current beta test allows.

The default scale of the displayed map is dependent on the shape of the web browser. See the screenshots below for an example of this issue. Finding a manner to standardize the size of the map window, or aggregate results over a fixed area, would reduce the severity of this problem.

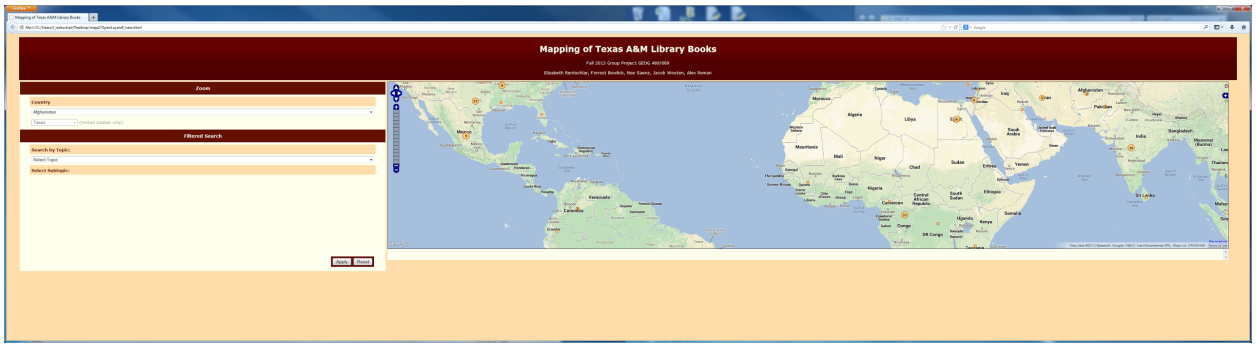

Figure Four: Map Display on Two Monitors.

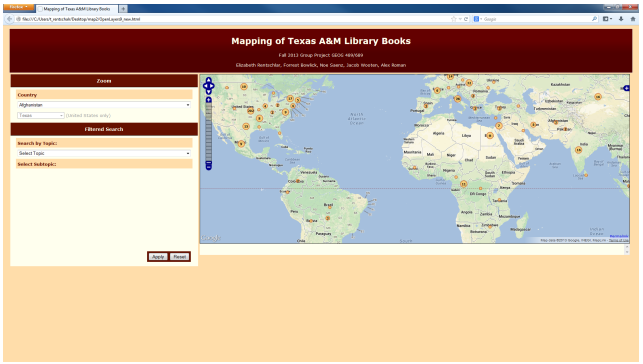

Figure Five: Map Display on One Monitor.

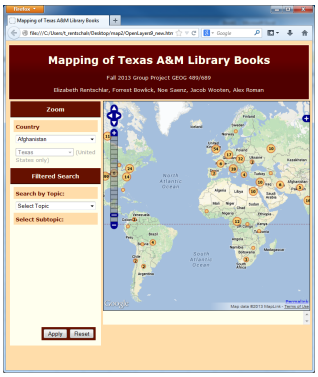

Figure Six: Map Display on Half of a Monitor.

Iterating through the descriptions of the books to pull the subject and sub-subject information in this test is done with a 'for' loop that checks the descriptions, iterates the scraped metadata, and provides a functional way to obtain the information. However, as dataset size increases, the length of time this loop takes to run also increases, resulting in a time-consuming and awkward process to extract the necessary data for the project. Future versions of this program must overcome the data extraction difficulties to be a functional, wider use example of this capability. While sufficient for the beta test, future versions should form a different construction of the iteration process to remove the inherent awkwardness of a bevy of 'for' looping.

5 References

OpenLayers documentation available online at: http://openlayers.org/.

ScraPy documentation available online at: http://scrapy.org/.

Texas A&M Map and GIS Library website: http://library.tamu.edu/about/collections/map-gis-collections-services/.