Volume 3, No. 9, September 2012

Journal of Global Research in Computer Science

ISSN-2229-371X

RESEARCH PAPER

Available Online at www.jgrcs.info

EFFICIENT SEARCHING FOR GEOGRAPHICAL DATA GATHERING USING KNOWLEDGE MINING

Kuldeep Singh Jadon*[1], Satyam Maheshwari[2]

[1]M.Tech (Software System) ,SATI,Vidisha(MP)-India,
kuldeep788@yahoo.co.in

[2] Dept. of Computer Application , SATI,Vidisha(MP)-India
`satyam.vds@gmail.com

Abstract: We live in a mutual world. Perhaps, the most productive way of data collection is to recruit a large number of people for data entry. Internet provides a very powerful and adaptable medium to bring together such a large number of people With this in mind, we have developed a framework which enables novice users to build applications for social co-creation of geographic data, called Geogather. Users can create interactive maps with multiple layers to collect geographic information. Geographic information is any data which has a (latitude, longitude) value attached with it. An application can have multiple layers to differentiate features of different types, such as hotels and schools. Geogather allows user roles, thereby adding a social abstraction on top of the application, so users can control the process of data collection. We have used the OpenLayers JavaScript library to create maps, and the web2py web framework to control the application work flow.

INTRODUCTION

Geographic Data means any data which has a latitude and a longitude information attached with it. In recent times, computers have changed the way people handle geo-graphic data. There is now widespread use of Geographical Information Systems(GIS)[4] to handle such data.

A geographic information system (GIS), or geographical information system, is any system that captures, stores, analyses, manages, and presents data that are linked to location. In the simplest terms, GIS is the merging of cartography, statistical analysis, and database technology.[2]

However, most GIS focus on data storage and analysis rather than data collection. Data is generally collected using GPS devices, which provide raw data that has to be parsed and customized. Large scale collaboration for data collection is difficult. Some web based GIS do provide the means for collaboration in collecting data, but they do not provide a framework to build personalised applications to collect customized data. Comparing data from different sources is also difficult in such application. We believe that a framework which allows users to create applications on top of maps to collect data from the crowd will greatly enhance the ease of collecting geographic data.

Information for Geoinfomer:

Adding location information to data can greatly enhance its usability. It can help in generating statistics about different regions and finding nearest features such as ATMs and Hotels. We are trying to build a platform which enables novice users to create applications to collect such data. Geographic data can be used in several interesting ways, some of which are

a. We can collect information about ATMs of various banks in the region. With this information, users can be easily directed to the nearest ATM.
b. We can collect reviews of restaurants in a region. A user can query the highest rated restaurant within a certain radius.
c. If we collect the information about NGOs working in a region, in the event of natural disasters, help can be contacted efficiently. If we also mark the road blockages, we can find the best routes to connect the affected areas with the NGOs and the relief work will be much more efficient.

Epigrammatic Prologue to Geogather:

Geogather is a web based framework with which people can create applications to collect geographic information. Geogather allows user roles, thereby adding a social abstraction on top of the application, so users can control the process of data collection. Geogather supports multiple layers to an application, allowing the users to separate Geogather supports multiple layers to an application, allowing the users to separate different types of features, such as Hotels and Restaurants.

ARCHITECTURE DETAILS OF GEOGATHER

There currently exist many collections of “points of interest”. However, most of them are proprietary, and all of them follow a pre-defined data format (the attributes for each POI). Applications which allow users to add data to such collections number far less.

A popular example is google maps , which allows users to create personalized maps and add points and roads on them. But it is not suitable for our objective because of the following reasons.

a. Collaborators have to be invited to be able to edit the map, making large scale collaboration extremely difficult.
b. Another drawback is that the only attribute google maps allows users to add is description, and hence the attribute information cannot be controlled.

c. If we use a single map for all types of features, it is difficult to segregate them.
d. If we use different maps for different types of features, overlaying them is not possible, making it difficult to compare them.

We have tried to overcome these shortcomings and create a framework which is easy to use, allows large scale collaboration, allows users to customize the feature attributes, and allows users to compare different types of features by overlaying different layers on top of each other.

Application Architecture:

Geogather allows users to create their own geographic applications. Each application can have several layers. Each layer is meant to collect and display features which have common attributes. Attribute values are stored within the features themselves.

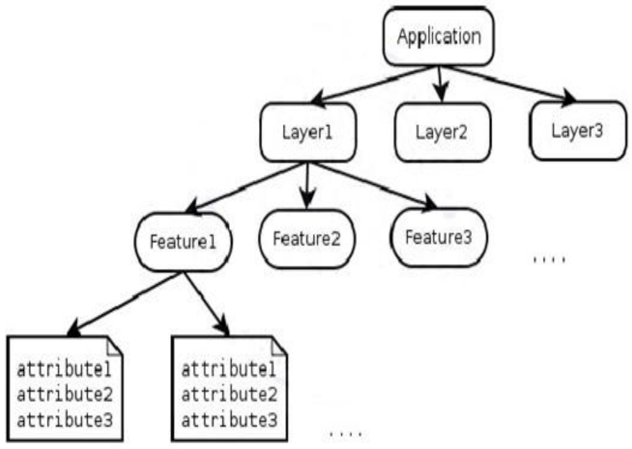


Figure 1. Application Structure

If a layer has proper permissions, it can be used by multiple applications. This allows for collaboration between multiple parties for data collection and comparison

Layers *:*

A layer represents a virtual layer on the map which is a collection of features of the same type. Each layer's attributes and permissions can be customized individually. For each layer, the application administrator can set the following options

a. Publicly Viewable - Controls if unauthenticated users can view the map.
b. Publicly Editable - Controls if unauthenticated users can edit the map.
c. Feature Moderation - Controls if new features added to the layer should be moderated.
d. Attribute Moderation - Controls if new attributes added to the features in the layer should be moderated.
e. Number of Attributes - Sets the number of attributes to be shown for each feature in the layer.
f. Marker - Sets the marker for the layer

The application administrator can assign several users as moderators for different layers. If feature moderation is set, new features will not be shown on the map until moderated. If attribute moderation is set, new attributes will not be shown until moderated by the moderators.

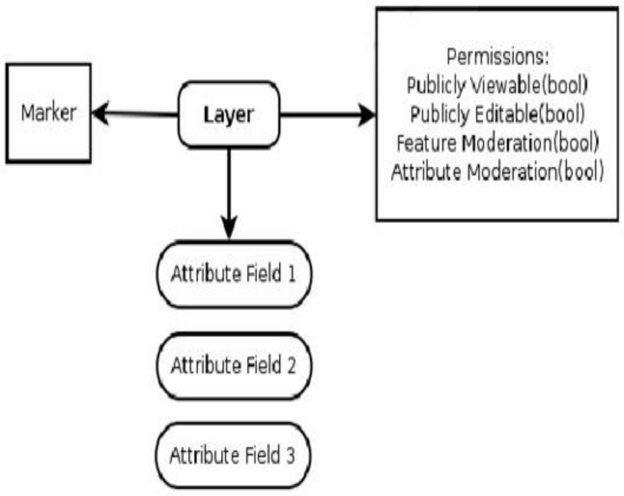


Figure 2. Layered Structure

Attributes:

The application administrator can customize the format of information to be collected for each layer. This is achieved by customizing the fields for that layer. Each field represents a feature attribute. A layer can have any number of attribute fields, and each field can be of the type String, Integer or Float. The administrator can also select if the field is required or not.

The attributes may be single entry, meaning only one attribute value per feature, or multiple entry, meaning multiple attribute values per feature. For example, a contact number of an organization should be a single entry attribute, whereas reviews of a restaurant can be multiple entry attribute.

Moderation:

The administrator may assign users as moderators for the layers that he created. The moderators can see the features and attributes which are not yet moderated and have the option to either accept, or delete them. Once a moderator accepts a feature and or attribute, it can be viewed by normal users.

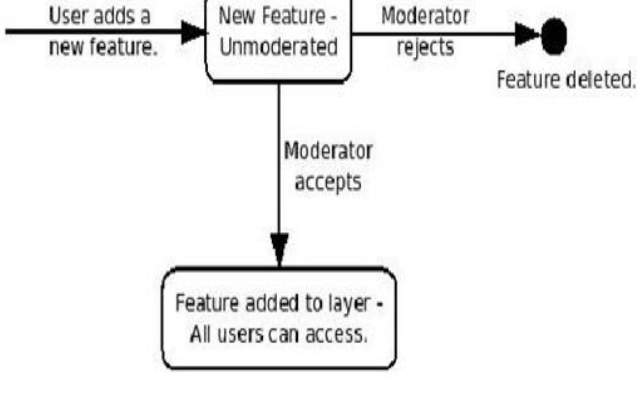


Figure 3. Moderation

Framework Architecture:

We divide the framework architecture in two broad categories front-end and back-end. The front-end, which displays the map and handles the user interaction with it, and the back-end which handles the logic, the user authentication and authorization, and the interaction with the database.

JavaScript is used extensively for the front-end. Map display, interaction, feature loading, form submission are handled with JavaScript ,web2py framework handles the back-end.
The components are as:

Load Layer:

Initially, all the layers in the application are loaded in the available layers section. When a layers is selected from them, an asynchronous call is made to the back-end function. A vector layer is added to the map and the features from the KML file are added to it. Finally the layer name is removed from the available layers section and added to the added layers section.

Remove Layer:

When the user clicks on a layer in the added layers section, the selected layer is removed from the map and the destroyed. The layer name is removed from the added layers section and added to the available layers section. If the user wants to reload the layer,he has to select the layer name from the available layers section again, and the KML file will be generated and the layer reloaded.

Add Feature:

To add a feature to a layer, first we have to select the desired layer as the active layer.The layer added to the map most recently is the active layer. There are two ways to add the feature, either by clicking on the map at the feature's location, or by entering the feature address in the search box. In both cases, a marker is added at the desired location, and a popup is shown with a form for the feature name.

Add Attribute:

When the click control is off, we can select features to view and add attribute infor mation. On selecting a feature, a popup is opened which gives the feature information. It lists the feature name, and attributes. The number of attributes shown is controlled by the number attributes field of the layer .The popup also contains a link to add more attributes to the feature. Clicking this link adds a form to the page containing a form for the attributes of selected feature. When an attribute is added to a feature, the changes will only be visible only after reloading the layer.

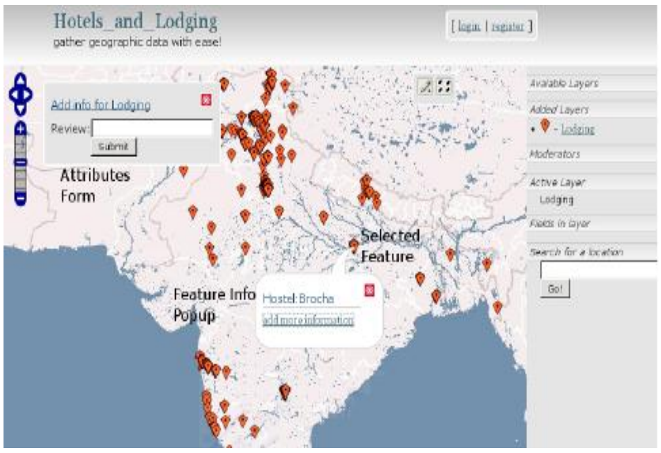


Figure 4. Add new attribute

Get Nearest Feature:

To get the features within a certain radius of a point, we first select a layer and then activate the get nearest features control from the toolbox. By clicking on the point we are interested in, a popup is opened which asks us the radius to consider, and returns all the features on the layer within the specified radius.

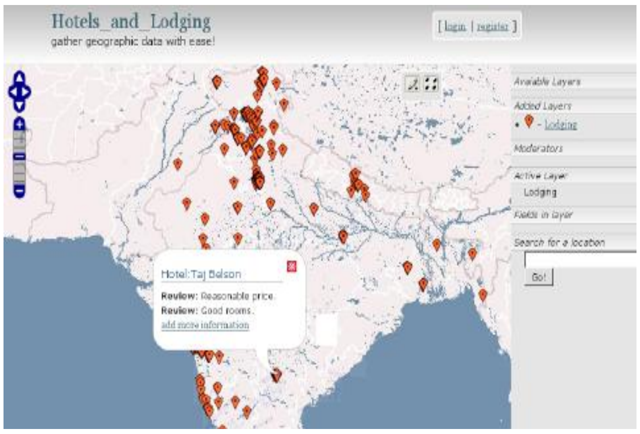


Figure 5.Get Nearest Feature

PERFORMANCE FACTS

We use MVC (Model view controller) model for this framework[10], in this for an application the steps of storing file for better performance is as

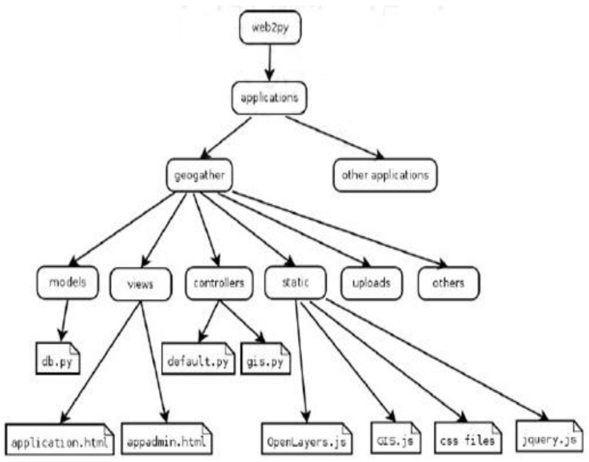


Fig 6.Directory Pecking Order

Model:

The database design resides in the Models section. The tables are defined in the file using DAL (Data Abstraction Layer). Geogather allows one layer to be attached to several applications. For this, we have a separate table for applications and a separate table for layers, and a table defining the relationships between the layers.

For example, a tourism company may create a public application with layers for hotels and tourism destinations, and another user may create a private application for all the places he wants to visit. The second user can include the public layer for hotels in his application, and plan his travel bookings easily.

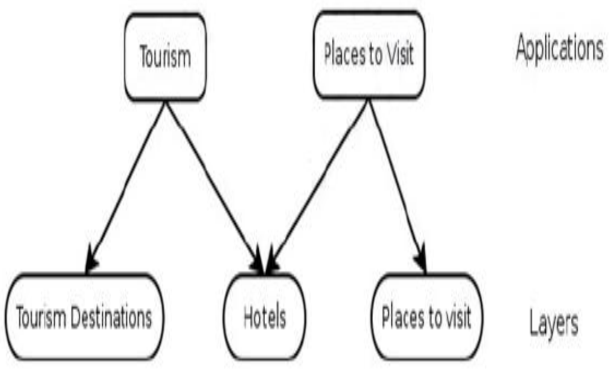


Fig 7 .Example Of Layer And Application Relationship

Geogather allows users to customize the information they want to gather for each layer, resulting in a dynamic schema for each layer. This is achieved by creating the layer attribute tables dynamically depending on the fields requested by the user. Schema information for each layer is stored in the table fields. When a new layer is created, an entry is made in the layers table corresponding to the new layer. When the user adds a new field to the layer, corresponding entry is made in the fields table

View:

Views contain the display styles and rules. The application page, displaying the map and layer controls. The controls depend on the user role, which may be administrator, moderator, authenticated user or anonymous user.

An unauthenticated user will only be able to view layers which are publicly viewable. An authenticated user will be able to view and edit all layers. Moderators can view and edit all the layers. Moderators also have the option to see the unmoderated features on the map, and accept or reject them. The administrator will get a link to the administration page. Only the administrator can access this page. It allows the administrator to add/delete layers from the application, modify the added layers, and add/delete moderators from the layers on the map.

Controller:

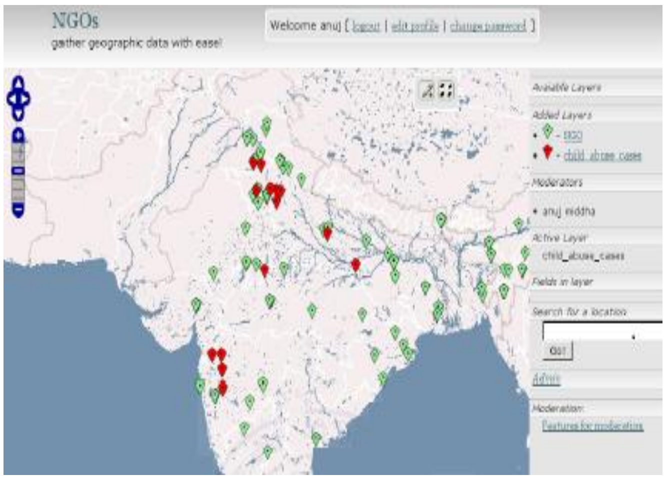
All the application logic reside in the controller, we have different function by using them we can controlee the flow of the application. By using these function we can work only on logics without interfering in the model and view part

Other helper functions are provided for tasks such as checking authorization, generating and submitting forms, adding and deleting entries from the database and such.

FRAMEWORK APPLICATION

NGOs:

Information about Non Government Organisations(NGOs) in India is spread across different websites, with no website giving an exhaustive list of NGOs. Moreover, these lists are rarely updated, and new information is difficult to find. One main reason for this is that since the content is not user generated, it is a difficult job for one person to gather and update a large amount of data. By providing a social co-creation framework, the work is distributed among the many users, and as a result, we can get a large amount of good, updated data.For the NGOs application, we have created a layer for the NGOs themselves. The layer permissions are set for public viewing, with feature and attribute moderation.


NGOs
gather geographic data with ease!
Welcome anuj [logout | edit profile | change password]

Available Layers
Added Layers
- NGO
- child abuse cases

Moderators
- anuj siddha

Active Layer
child_abuse_cases

Fields in layer

Search for a location

Go!

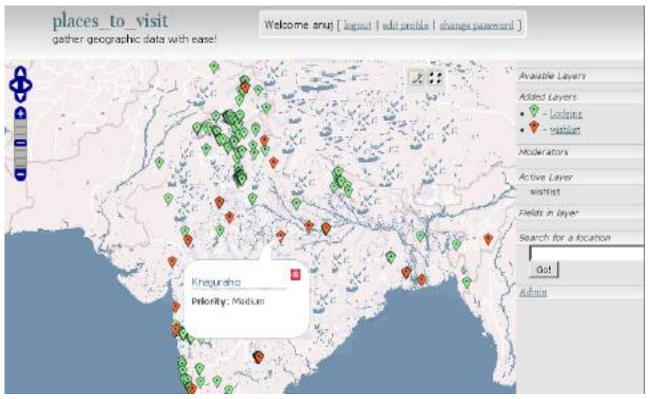
Admin

Moderation:
Features for moderation

Figure 8. NGOs Information

Another advantage of using Geogather for collecting information about NGOs[12] is that during times of emergencies such as natural disasters, we can easily locate the nearest NGOs from the disaster area and mobilise them in a very short time. If we want to study the impact of NGOs on social problems, we can create layers for the relevant problems such as child abuse and corruption, and by overlaying the two layers on top of each other, we can get the statistics to study the impact.

Place To Visit:


places_to_visit
gather geographic data with ease!
Welcome anuj [logout | edit profile | change password]

Available Layers
Added Layers
- Lodging
- wishlist

Moderators

Active Layer
wishlist

Fields in layer

Search for a location

Go!

Admin

Khajuraho
Priority: Medium

Figure 9. Place to visit

This application demonstrates how we can create personal applications, and use data from public layers with them. We create a layer "wishlist", which is a list of places we want to visit, and set its permissions such that its neither publicly viewable nor editable. We then import the "Lodging" layer from the previous application, and add that to our application. The attributes have only one field, "Priority". Now the user can stack the "wishlist" layer on top of "Lodgin" layer and by using the get nearest features option, get information about hotels and plan his travel.

CONCLUSION

We studied the existing applications for "web based geographical data collection", and found that they did not meet our requirements. Geinformer is designed in a way so that it is easy to use for novice users, as it provides an abstraction for geographic data in the form of layers and attributes, and hides the intricacies of database design and management from the user. At the same time, it is designed to provide flexibility and control to the user by providing permission and moderation settings for individual layers.

The user is also provided complete control over the attribute schema, making it easier to customize layers according to individual needs.

Geogather provides an interface to easily collect geographic data. But little work is done in the field of data representation and data reckoning. A possible development can be to add a statistics generator, which generates statistics based on different attributes. Another possibility is to divide the map into regions, and calculate the statistics based on regions. This can help in applications where users want to compare some statistics, such as crime rate or child mortality rate, of different states or districts. Currently Geogather supports layer wide permission settings. This can be improved by adding attribute wide permission settings, which will give users much more control over their applications.

REFERENCES

[1]. Drozd A., Benford S., Tandavanitj N., Wright M., Chamberlain A., 2006, Hitchers: Designing for Cellular Positioning, Ubicomp, Springer Lecture Notes in Computer Science, Vol.4206, pp. 279-296.

[2]. Kiefer, P., Matyas, S., Schlieder, C. (2006a). Systematically Exploring the Design Space of Location-based Games, In: Strang, Th., Cahil, V., Quigley, A. (eds.): Pervasive 2006 Workshop Proceedings, Poster presented at PerGames2006, 07. May 2006, Dublin, Ireland, ISBN 3-00-018411-2, pp. 183-190.

[3]. Egenhofer, Max. 2002. Toward the Semantic Geospatial Web. In Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems, 1-4. McLean, Virginia: ACM Press.

[4]. http://en.wikipedia.org/wiki/Web Map Service.

[5]. Kang, J., and Naughton, J. F. 2003. On Schema Matching with Opaque Column Names and Data Values. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), 205-216. San Diego, California: ACM Press.

[6]. LICGF. 2003. Community Planning Resources Web site of the Land Information and Computer Graphics Facility, University of Wisconsin-Madison, www.lic.wisc.edu.

[7]. Naughton, J. F.; DeWitt, D.; Maier, D.; and others. 2001.The Niagara Internet Query System, IEEE DataEngineering Bulletin 24(2): 27-33.

[8]. http://maps.google.com.

[9]. Anders K. (2003), A hierarchical graph-clustering approach to find groups of objects, working paper for ICA generalization commission workshop, available at http://www.geo.unizh.ch/ICA/docs/paris2003/papers03.html

[10]. http://mapserver.org.

[11]. Berkhin P. (2004), Survey of clustering data mining techniques, Datanautics, Inc. Research Papers, available at http://www.accrue.com/products/rp_cluster_review.pdf

[12]. http://mapnik.org.

[13]. Wiegand, N.; Zhou, N.; and Cruz, I. F. 2003. A Web Query System for Heterogeneous Geospatial Data. In Proceedings of the Fifteenth International Conference on Scientific and Statistical Database Management (SSDBM), 262-265. Cambridge, MA: IEEE Computer Society.

[14]. Halkidi M., Batistakis Y. and Vazirgiannis M. (2001), On clustering validation techniques, Journal of Intelligent Information Systems, Kluwer Academic Publishers. 17: 2/3, pp. 107 – 145.

[15]. Han J, Kamber M. and Tung A. K. H. (2001), Spatial clustering methods in data mining, in: Miller H. J. and Han J. (2001, eds.), Geographic data mining and knowledge discovery, Taylor & Francis, London and New York, pp. 188 – 217.

[16]. Jiang B. and Claramunt C. (2002), A Structural Approach to Model Generalisation of an Urban Street Network, presented at the 4th AGILE Conference on Geographic Information Science, 24-27 April 2002, Mallorca, Spain (a revised version of this paper has been published in GeoInformatica, **8** (2): 157-171, June 2004).

[17]. Jiang B. and Harrie L. (2003), Selection of Streets from a Network Using Self-Organizing Maps, ICA Generalization Workshop, Paris, April 28 - 30, 2003 (a revised version of this paper has published in Transactions in GIS, Blackwell Publishers, Vol. 8, No.3, pp. 335 - 350.)

[18]. Cruz, I. F.; Rajendran, A.; Sunna, W.; Wiegand, N. 2002. Handling Semantic Heterogeneities Using Declarative Agreements. In Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems, 168-174. McLean, Virginia: ACM Press.

[19]. OpenGIS Consortium, 2001. Geography Markup Language (GML) version 2.0. http://www.opengis.net/gml/01-029/GML2.html.