Spatio-Textual Spreadsheets: Geotagging via Spatial Coherence*

Michael D. Lieberman† codepoet@cs.umd.edu
Hanan Samet† hjs@cs.umd.edu
Jagan Sankaranarayanan† jagan@cs.umd.edu
Jon Sperling‡ Jon.Sperling@hud.gov

ABSTRACT

The *spatio-textual spreadsheet* is a conventional spreadsheet where spatial attribute values are specified textually. Techniques are presented to automatically find the textually-specified spatial attributes that are present in spreadsheets. Once the spatial attributes have been identified, an accurate translation of the values of the spatial attributes to their actual geographic locations is needed (known as *geotagging*). The key observation is that spreadsheets with spatial data exhibit *spatial coherence* — that is, cells with spatial data that are nearby in the spreadsheet contain data that share spatial characteristics in the real world. These techniques also allow richer search engine results by returning actual tuples from spreadsheets instead of simply links to the spreadsheets. Moreover, when the search key is a particular location, results in proximity to the query can be provided rather than just exact matches.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.4.1 [**Information Systems Applications**]: Office Automation—*Spreadsheets*

General Terms

Algorithms, Design, Performance

Keywords

Geotagging, spatio-textual, spreadsheets, spatial coherence

*This work was supported in part by the National Science Foundation under grants IIS-08-12377, CCF-08-30618, and IIS-07-13501, as well as NVIDIA Corporation, Microsoft Research, Google, the E.T.S. Walton Visitor Award of the Science Foundation of Ireland, and the National Center for Geocomputation at the National University of Ireland at Maynooth.

†Institute for Advanced Computer Studies, Center for Automation Research, Department of Computer Science, University of Maryland, College Park, MD 20742, USA.

‡HUD Office of Policy Development & Research (PD&R), 451 7th St. SW, Washington, DC 20410, USA.

ACM GIS '09, November 4–6, 2009. Seattle, WA, USA

1. INTRODUCTION

Spreadsheets can be viewed as embodying both direct manipulation and abstraction, where columns and rows are transformed to create new information. There is no restriction on the type of data that can be stored in the spreadsheet. In the case of spatial attributes, there is an additional possible dimension of manipulation, namely through the use of a map and queries on it. In addition, the presence of spatial attributes means that we are no longer restricted to viewing the data using a table.

We use the term *spatial spreadsheet* to characterize spreadsheets that have spatial attributes and which also include operations to visualize the data. The concept of a spatial spreadsheet is quite broad and includes examples such as the spreadsheet for images proposed by Levoy [11]. In this case, the cells of the spreadsheet contain graphical objects such as images, volumes, or movies, as well as widgets such as sliders and buttons. The cells can be manipulated by a large class of operations such as filters and quantizers. A more limited interpretation is provided by the spatial spreadsheet of Iwerks and Samet [9, 10] where the cells are maps or spatial relations and the operations include unions, intersections, joins, and so on. The SAND Internet Browser [5, 14] is an even more limited variant of a spatial spreadsheet where the important feature is the ability to create new relations (with corresponding indexes, as appropriate) from existing ones by operations such as selection and spatial join.

In this paper we focus on the interpretation of a conventional spreadsheet as a spatial spreadsheet. In such a spreadsheet, the spatial attributes are specified textually, and we term this a *spatio-textual spreadsheet*. Our aim is to automatically determine the textually-specified spatial attributes that are present in a spreadsheet, with no user intervention. Our task is two-pronged. First, we must identify the spatial attributes, as well as their types (e.g., some strings correspond to names of cities or states, some numbers correspond to ZIP codes, etc.). Second, we must provide an accurate translation of the values of the spatial attributes to their actual geographic locations. In particular, at its narrowest interpretation, when the spatial entity is a point object such as a gas station or a city at a low zoom level, we seek to assign a pair of geographic coordinate values to each spreadsheet row, assuming that the geographic location can be appropriately described by the lat/long pair of some point within it. For data with spatial extent, such as line segments and regions (e.g., roads and counties), the situation is more complex, but we can make use of methods adopted for spatial databases (e.g., SAND [3, 4]) where pointers to spatial data structures are used that capture the spatial features. In the rest of this discussion, unless otherwise noted, we assume locations for ease of exposition.

The ability to deal with spatial data of arbitrary type (i.e., with extent rather than just points) and, most importantly, the existence

of a fully automated algorithm to infer spatial attributes in spreadsheets is what enables us to have a scalable method to process the very large, heterogeneous collections of spatio-textual spreadsheets that are prevalent on the Web, and distinguishes our system from existing products and tools that aid the mapping of spreadsheets such as Spreadsheet-To-Map[1], Map A List[2], Excel To KML[3], and ExpertGPS[4]. In particular, these tools all require users to provide geographic coordinate values with each row, or at least to manually specify which column corresponds to a spatial data type (usually street address) which is then geocoded to lat/long coordinate values. As a result, these semi-automated tools require, at a minimum, manual interaction for each different spreadsheet schema, of which there are many on the Web. Moreover, they are primarily designed to facilitate the construction of mashups where the spatial data is usually constrained to be points rather than have extent.

The process of interpreting words as geographic locations and providing the coordinate values of the locations is known as *geotagging* [2]. Geotagging in general is difficult because recognizing references to locations is hampered by ambiguity in natural language (e.g., is "Jordan" a surname or a location?) and determining the correct interpretation of a given location name can be challenging (e.g., deciding which of over 140 interpretations of "Paris" is the correct one). The STEWARD system [12] is an example of the application of geotagging to a collection of unrelated documents such as those comprising the hidden Web. The NewsStand system [15] is another example of this approach where the data consists of dynamically changing RSS news feeds. In this paper we expand on our work in this area by devising an automated method of geotagging spatio-textual spreadsheets, which is further complicated by the fact that data and metadata are mixed, i.e., column headers are simply data values in spreadsheet cells, and so are harder to identify. Also, spreadsheets commonly suffer from errors such as missing, incomplete, or erroneous data [13].

Despite these challenges, we can take advantage of spatial data placement conventions used by human creators to effect correct geotagging. Our key observation is that spreadsheets with spatial data exhibit what we call *spatial coherence*. That is, spatial cells (i.e., cells with spatial data) that are nearby in the spreadsheet contain data that share spatial characteristics in the real world. In particular, spatial data in the same column are usually of the same spatial type (e.g., columns for "State", "County", and "City"), while spatial data in the same row usually exhibit a containment relationship (e.g., "Maryland", "Prince George's County", "College Park", and "20742"), termed *column coherence* and *row coherence*, respectively. Furthermore, spatial data in adjacent or nearby rows usually share containers. For example, a natural way to organize data for multiple cities in Maryland is to group city rows by county, which also reflects a sorting process where a primary-secondary key interpretation is imposed on the columns. It is important to note that existing work on automatic spreadsheet processing, such as header inference [1] and error detection [7, 8], has not considered spatial coherence as a source of evidence.

The rest of this paper is organized as follows. Section 2 presents some challenges in geotagging spreadsheets and our geotagging methods. Section 3 presents a visualization of one spreadsheet tagged with our methods. Finally, Section 4 offers future avenues of research and draws concluding remarks.

[1] http://gmb.bz/
[2] http://mapalist.com/
[3] http://earthpoint.us/ExcelToKml.aspx
[4] http://expertgps.com/

2. GEOTAGGING SPREADSHEETS

2.1 Spatial Cell Recognition

The first step in geotagging spreadsheets is recognizing spatial information within the spreadsheet's cells, essentially classifying certain cells as spatial and others as nonspatial. Our goal is creating a *location-augmented* spreadsheet from our input spreadsheet by associating spatial information in the form of geographic coordinates with each spatial cell. Note that these initial cell assignments are independent of values in other cells. In later steps we resolve inconsistencies among the cells by relying on the spatial coherence of spreadsheets. The spatial data types that we seek to recognize in spreadsheet cells include country, state, county, city, ZIP code, ZIP+4, street intersection, and street address.

Recognizing spatial data in spreadsheets involves many challenges. In addition to human errors such as missing, incomplete, and erroneous data, textual geographic data suffers from various forms of ambiguity and inconsistency. A given text phrase in a spreadsheet cell might be the name of many geographic places, or might not be a geographic reference at all. For example, "Washington" is the name of many cities, counties, and other geographic entities in the USA, but is also a very common surname. Furthermore, place names may be *underspecified* if their geographic context is assumed implicitly. That is, a spreadsheet containing a mention of "Prince George's County" may not be qualified with "Maryland", its container, if the entire spreadsheet consists of data about counties in Maryland. In addition, geographic *cue words* such as "County" may be omitted from place names if the spreadsheet is understood by readers to contain county data. Finally, spreadsheets vary in the number of columns used to store components of each row's spatial data. For example, a street address may be stored in a single column, or the spreadsheet may contain several columns, one for each part of the address.

Our basic strategy in recognizing spatial cells is to search each subphrase of each cell's data for potential spatial data, employing several heuristic methods modeled after place recognition techniques for natural language text. To aid this search, we use a *gazetteer*, or geographic database of locations and their associated metadata, as an external source of geographic knowledge. We use the GeoNames[5] gazetteer, which contains over 7 million entries of locations around the world and their geographic coordinates. In addition to a lookup in the gazetteer, we search for explicit geographic cue words (e.g., "State of", "City of"), and append cue words for an additional lookup, since they may be absent in some spreadsheets (e.g., appending "County" to "Prince George's"). We also look for prominent places, such as large cities and political regions, as well as textually-specified hierarchies (e.g., "College Park, Maryland"). To recognize addresses, we use the Yahoo! address geocoding API[6]. For addresses broken across multiple columns, only the partial address will be available in a single cell, so we leave all possible address interpretations and defer resolution until later.

This process results in a set of spreadsheet cells with spatial information tagged to each cell, in the form of either gazetteer records or lat/long points. Note that our battery of recognition rules ensure that spatial cells are not missed, rather than being overly precise in marking spatial cells. That is, it is better to err on the side of generosity rather than caution when recognizing spatial data. Classifying nonspatial cells as spatial is not catastrophic, as we next use the spatial coherence properties of spreadsheet rows and columns to filter these errors.

[5] http://geonames.org/
[6] http://developer.yahoo.com/

2.2 Column Resolution

After assigning gazetteer records to each spatial cell, we now consider how to resolve the ambiguity introduced from having multiple gazetteer records or other types of spatial data assigned to a single cell. To see how, we note that most spreadsheet authors organize their data so that each data record corresponds to a row in the spreadsheet, and each data attribute corresponds to a column. In this way, data records (rows) all share the same attribute types (columns), including spatial attributes, if present. Furthermore, for spreadsheets with multiple spatial attributes, authors usually segregate different spatial data types into different columns, since this is a natural way to organize spatial information. For example, many spreadsheets contain multiple separate columns for different geographic hierarchy levels. Therefore, the first step toward assigning locations to rows, which we call *column resolution*, is first to classify each column in the spreadsheet as spatial or nonspatial, and second, for each spatial column, to assign the proper spatial data type. These spatial types serve as evidence for filtering the gazetteer records assigned to the cells in each column to only those that are consistent with the column's type, thereby ensuring *column coherence* for all gazetteer records assigned to cells in the column.

We note that over all cells in a single spatial column, there will usually be gazetteer records of a single spatial type that all cells have in common, which is the same spatial type used to construct the column. Likewise, for nonspatial columns, some cells might have been assigned gazetteer records, but the majority will not. In other words, we can loosely view the column's spatiality as a random variable, with each cell representing a sample. Thus, we may determine whether the column is spatial or not, and its spatial data type, by taking advantage of an averaging effect across all cells in the column. Also note that this averaging effect is robust to one or a few erroneous data cells in the column. Using these facts for column resolution, we classify a given column j as nonspatial if its cells contain relatively few gazetteer records, and remove all gazetteer records from its cells. Also, for each cell c in a spatial column j, we assign a spatial type to j by filtering out the gazetteer records assigned to c that do not share a spatial type with most of the other cells in j. This results in a *column-coherent* copy of the input spreadsheet, but does not necessarily mean that each spatial cell has been fully resolved.

2.3 Row Resolution

After producing a column-coherent spreadsheet, we next enforce *row coherence* among the remaining spatial data in the spreadsheet by removing conflicting gazetteer records from the cells in each row. To do so, we note that if the spreadsheet contains multiple spatial columns, they usually all aid a human viewer in grounding the row's location, and further that the spatial columns tend to exhibit a containment relationship. For example, a typical spreadsheet might include three spatial columns corresponding to State, County, and City, with a row such as Maryland, Prince George's County, and College Park. This row's location is actually College Park, with Maryland and Prince George's County serving as geographic context.

For spreadsheets with multiple spatial columns, the above observation can be used to achieve row coherence by checking for containment among the gazetteer records in spatial cells in a single row. However, many spreadsheets only have a single spatial column, which precludes the use of containers within a single row. Furthermore, if a spatial cell's data is not recognized as a location, a needed container might be missed, which will prevent the row from being assigned a location. To overcome this limitation, we exploit another common property of spatial spreadsheets, namely that adjacent rows with locations usually (but not always) exhibit geographic proximity in terms of a hierarchy or geographic distance. This organization, while not required of spreadsheet data, is natural as it makes intuitive sense for human viewers, and hence spreadsheet authors use this organization frequently.

Our strategy to achieve row coherence follows from the above spatial coherence observations. We begin with the coarsest possible resolution for each row, and successively refine the location to ever finer resolutions based on additional spatial columns, as long as the finer resolutions are consistent (in the sense of containment) with our current notion of the row's location. In addition, where even a coarse location is unavailable to test containment for a given spatial cell c, we gather potential containers from nearby cells above and below c in the spreadsheet, to take advantage of the row grouping usually imposed by human creators and described above. Aside from resulting in row-coherent spatial data, this approach has the added benefit that even if no consistent finer resolutions are discovered, we can fall back to a coarser, approximate resolution of the row's location, which can aid in determining the spreadsheet's overall geographic focus. Like our column resolution method, it is also robust to possibly missing spatial data.

3. EXAMPLE

We now present an example of spreadsheet data with geographic attributes, tagged using our methods. Figure 1 shows project funding requests received by the Washington State Department of Ecology Water Quality Program[7] for 2010. This display was generated from a spreadsheet containing columns for the project title and amount requested, and a single spatial column with textual data values consisting of geographic cue words along with each spatial entity (e.g., "Spokane, City of", "Seattle, City of", "Spangle, Town of"). Prior to mapping the data, we aggregated requests by geographic entity and summed the amounts requested. Each data point thus represents a set of projects proposed by a single geographic entity (i.e., county, city, or town), with the point's size indicating the total monetary amount requested for all projects. In the figure, the largest amounts stand out clearly and correspond to the largest cities in Washington, such as Spokane ($52 million), Snohomish ($31 million), and Everett ($27.5 million). Also, most projects requested fall in the Seattle-Tacoma geographic area. Somewhat more surprising are the multitude of relatively small cities requesting large amounts, such as Soos Creek ($13.5 million), Vancouver ($13 million), and Yakima ($12.5 million), which might be indicative of major improvements to those areas and warrant further investigation. Clearly, mapping the spreadsheet affords users an intuitive geographic display for spatially-relevant information.

4. DISCUSSION

Automated geotagging of spreadsheets offers potent opportunities for geospatial exploration and retrieval applications. However, even better exploration could be accomplished with a robust method of inferring spreadsheet headers. Doing so creates associations between spatial data and its meaning, which can allow a search engine to return actual tuples from the spreadsheet rather than simply a link to the spreadsheet, thereby serving as a geographic "fact-finding" engine. For example, a search for "American Recovery and Reinvestment Act" near "College Park, Maryland" could return relevant rows from the appropriate spreadsheet, rather than forcing users to manually examine the spreadsheet to find the relevant information. Existing work [1, 6] on automatic header inference does not make use of spatial data.

[7] `http://ecy.wa.gov/programs/wq/`

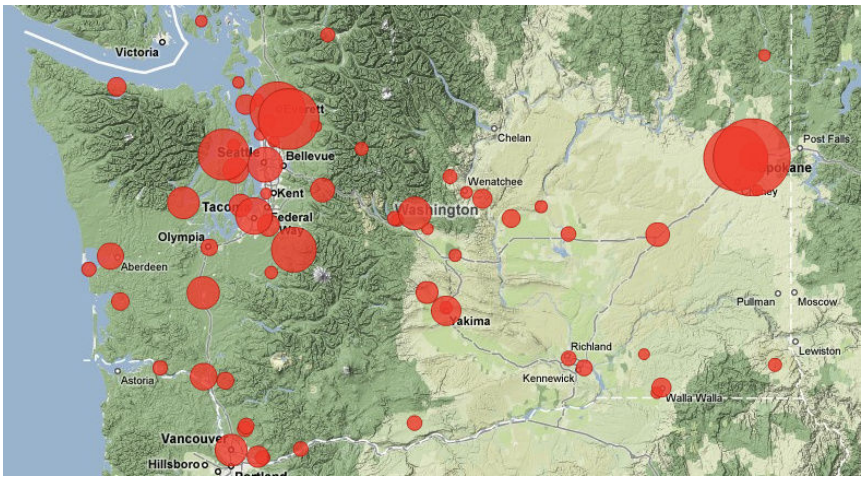

Figure 1: Washington State Department of Ecology Water Quality Program funding requests for 2010, aggregated by location. Data point size indicates the total size of all requests from the geographic area.

In addition, further investigation is warranted toward appropriate user interface controls that allow simple exploration of mapped geospatial spreadsheet data. Many spreadsheets contain a large number of columns, most of which might not be of interest to any particular user. Therefore, it is important to settle on appropriate interface choices that allow for filtering or expansion of particular attributes. Other controls might allow the filtering of spreadsheet rows from the map, or sorting rows by a given attribute and changing each row's corresponding map marker according to its rank.

5. REFERENCES

[1] R. Abraham and M. Erwig. Header and unit inference for spreadsheets through spatial analyses. In *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC'04)*, pages 165–172, Rome, Italy, Sept. 2004.

[2] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-Where: Geotagging web content. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*, pages 273–280, Sheffield, UK, July 2004.

[3] W. G. Aref and H. Samet. An approach to information management in geographical applications. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 589–598, Zurich, Switzerland, July 1990.

[4] W. G. Aref and H. Samet. Extending a DBMS with spatial operations. In *Proceedings of the 2nd International Symposium on Advances in Spatial Databases (SSD'91)*, volume 525 of *Lecture Notes in Computer Science*, pages 299–318, Zurich, Switzerland, Aug. 1991. Springer-Verlag.

[5] F. Brabec and H. Samet. Hierarchical infrastructure for internet mapping services. In J. T. Sample, K. Shaw, S. Tu, and M. Abdelguerfi, editors, *Geospatial Services and Applications for the Internet*, pages 1–30. Springer-Verlag, New York, 2008.

[6] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the web. In *Proceedings of the 34th International Conference on Very Large Data Bases (VLDB'08)*, pages 538–549, Auckland, New Zealand, Aug. 2008.

[7] M. Clermont. Heuristics for the automatic identification of irregularities in spreadsheets. In *Proceedings of the 1st Workshop on End-User Software Engineering (WEUSE'05)*, St. Louis, MO, May 2005.

[8] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger. Automatic generation and maintenance of correct spreadsheets. In *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*, pages 136–145, St. Louis, MO, May 2005.

[9] G. S. Iwerks and H. Samet. The spatial spreadsheet. In *Proceedings of the 3rd International Conference on Visual Information Systems (VISUAL'99)*, pages 317–324, Amsterdam, The Netherlands, June 1999.

[10] G. S. Iwerks and H. Samet. The internet spatial spreadsheet: Enabling remote visualization of dynamic spatial data and ongoing query results over a network. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems (ACMGIS'03)*, pages 154–160, New Orleans, LA, Nov. 2003.

[11] M. Levoy. Spreadsheets for images. In *Proceedings of the 21st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'94)*, pages 139–146, Orlando, FL, July 1994.

[12] M. D. Lieberman, H. Samet, J. Sankaranarayan, and J. Sperling. STEWARD: Architecture of a spatio-textual search engine. In *Proceedings of the 15th Annual ACM International Symposium on A dvances in Geographic Information Systems (ACMGIS'07)*, pages 186–193, Seattle, WA, Nov. 2007.

[13] R. R. Panko. What we know about spreadsheet errors. *Journal of End User Computing*, 10(2):15–21, 1998.

[14] H. Samet, H. Alborzi, F. Brabec, C. Esperança, G. R. Hjaltason, F. Morgan, and E. Tanin. Use of the SAND spatial browser for digital government applications. *Communications of the ACM*, 46(1):63–66, Jan. 2003.

[15] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACMGIS'08)*, pages 144–153, Irvine, CA, Nov. 2008.