Structured Toponym Resolution Using Combined Hierarchical Place Categories*

Marco D. Adelfio Hanan Samet
Center for Automation Research, Institute for Advanced Computer Studies
Department of Computer Science, University of Maryland
College Park, MD 20742 USA
{marco, hjs}@cs.umd.edu

ABSTRACT

Determining geographic interpretations for place names, or toponyms, involves resolving multiple types of ambiguity. Place names commonly occur within lists and data tables, whose authors frequently omit qualifications (such as city or state containers) for place names because they expect the meaning of individual place names to be obvious from context. We present a novel technique for place name disambiguation (also known as toponym resolution) that uses Bayesian inference to assign categories to lists or tables containing place names, and then interprets individual toponyms based on the most likely category assignments. The categories are defined as nodes in hierarchies along three orthogonal dimensions: place types (e.g., cities, capitals, rivers, etc.), geographic containers, and prominence (e.g., based on population).

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Design

Keywords

Geotagging, toponym resolution, structured data extraction

1 Introduction

We present an algorithm for automatically categorizing and interpreting geographic place names (or *toponyms*) found in tables and lists. Many datasets and documents include informal references to places using toponyms, rather than explicit references using geographic coordinates, necessitating a transformation from toponym to geographic coordinates before any spatial processing can occur. By interpreting each place name as a specific geographic entity (a process known as *toponym resolution*), the document containing the list or table is *geotagged* with the locations it references. Toponym resolution and geotagging are common topics in current research but the results can be inaccurate when place names are not well-specified (that is, when place names are not followed by a geographic container, such as a country, state, or province name). Our aim is to utilize the context of other places named within the table or list to disambiguate place names that have multiple geographic interpretations.

Geographic references are a very common component of data tables that can occur in both (1) the case where the primary entities of a table are geographic or (2) the case where the primary entities of a table have geographic attributes. In many cases, the geographic references are not well qualified (for example, when "Paris, Texas, USA" is being referred to in a list of other towns in northern Texas, it may simply be presented as "Paris"). In order to resolve the location references to their intended geographic interpretations, we must make use of the context, which in this case is composed of the other toponyms in the same table column. In this sense, the table geotagging task is differentiated from the task of geotagging place names that are found in plaintext documents, as the place names in table columns are typically more homogeneous. In particular, we expect that there is an underlying *place category* which can describe the toponyms within a single column. Examples of place categories include *"states/provinces in North America"*, *"large cities in Bavaria, Germany"*, or *"airports in Italy"*. Figure 1 shows an example list of toponyms and potential place categories for different geographic interpretations of those toponyms. By identifying likely place categories for toponym lists, we can reduce the ambiguity of resolving individual place names.

Our method relies on a category formulation that we call *"combined hierarchical place categories"* to geotag tables that contain ambiguous place names with little or no qualifying context. Our approach uses a Bayesian likelihood model to assign geographic categories to toponym lists or individual table columns. This ensures coherence among the interpretations of toponyms that are expected to have a consistent theme (called *column coherence*). For example, assigning a coherent category to a list improves the odds of resolving "Washington" to mean "the State of Washington" when it appears in a list containing the values [Washington, Idaho, Oregon] (which are all names of American states) while resolving "Washington" to signify "Washington, DC" when it appears in a list containing the values [Washington, New York, San Francisco] (American cities).

At the core of our algorithm, we use a gazetteer (our implementation uses the GeoNames geographical database [4]) to identify possible geographic interpretations for each toponym in a table. For each of these possible interpretations,

*This work was supported in part by the NSF under Grants IIS-10-18475, IIS-12-19023, and IIS-13-20791 and by Google Research.

GIR'13, November 05-08 2013, Orlando, FL, USA Copyright is held by the owner/author(s). ACM 978-1-4503-2241-6/13/11$15.00.
http://dx.doi.org/10.1145/2533888.2533931

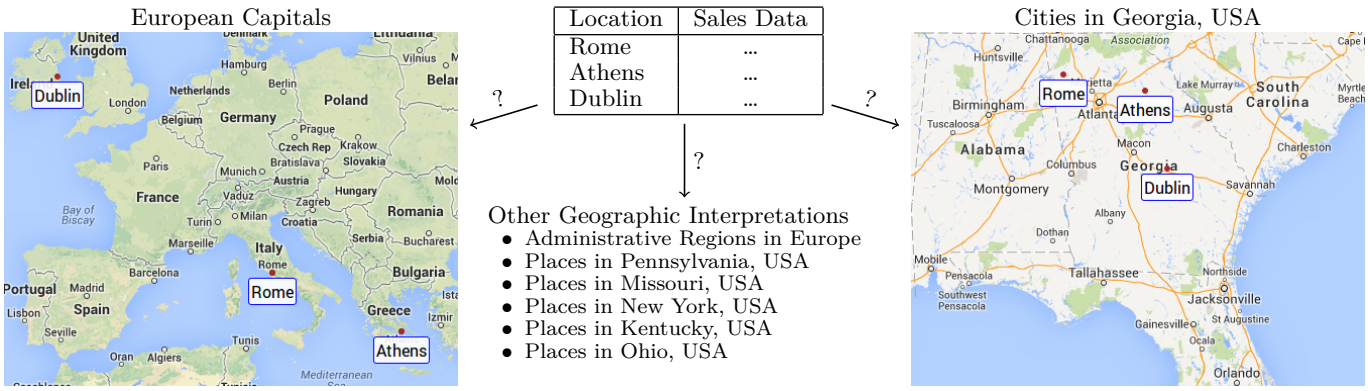

Figure 1: Plausible categories of geographic interpretations for a sample table. Many toponym sets, such as the one shown here, have several plausible geographic interpretations that fall in distinct place categories, whereas only one describes the true category of places that was intended by the table's author. In this case, each toponym has an interpretation as a capital city in Europe or as a large city in Georgia, USA. Other, less likely categories of geographic interpretations are listed. Our method assigns likelihood estimates to each category of geographic interpretations based on features of the categories and how well it fits the toponym list.

the gazetteer provides several descriptive attributes, such as the place type (e.g., "capital city"), geographic containers (e.g., the country in which the interpretation is found), and the population of the place. We create what we call a "category taxonomy" that represents all possible combinations of these three attributes. The values of each attribute are encoded in a tree where the root can describe all interpretations found in the gazetteer, whereas lower levels can only describe subsets of the interpretations. Using the category taxonomy, we can find all possible categories that describe an individual place name interpretation by finding each attribute of the interpretation in the attribute trees and taking the Cartesian product of all possible path prefixes of each attribute. For example, the city of Washington, DC can be described as a "place on Earth with population ≥ 0", but that is not a very useful category. It could also be described as a "city in North America with population ≥ 100", which is a slightly more discriminating category. There are many other ways to describe it within our system of categorization that lead up to the most discriminating category for Washington, DC, which is "Capital of an independent political entity, located in the District of Columbia, USA, with population $\geq 100{,}000$". The key concept is that many categories within our taxonomy can be used to describe multiple interpretations from our gazetteer, but for a given list or column of place names, there is one category that is simultaneously discriminating and broad enough to describe an interpretation for each place name.

The Bayesian approach of our method involves measuring certain characteristics of a small sample of training data and using that data to identify characteristics that are more likely to occur in true categories for a list or table column. As a concrete example, the statistics taken from our training data show that interpreting a set of toponyms in a way that they all have populations greater than 10,000,000 is about 18% more likely to be the expected way of interpreting them, rather than interpreting them as places that all have populations greater than 1,000,000, when the population is looked at as an isolated feature. The algorithm computes several such likelihood values and combines them for an aggregate likelihood score that a specific category of interpretations leads to the expected geotagging results.

The rest of this paper is organized as follows. Section 2 surveys prior geotagging work for unstructured and structured documents. Section 3 presents the geotagging algorithm, while Section 4 describes experiments showing our method's categorization and toponym resolution accuracy. Section 5 contains some concluding remarks.

2 Related Work

We have done considerable work on indexing spatial and temporal data [5, 6, 17, 18, 19] and similarity searching in the serial domain [16, 20, 21], as well as in a distributed domain [22]. The spatial data can also be expressed textually. Traditional systems that use geotagging, such as Web-a-where [2], STEWARD [10], and NewsStand [23], along with general systems for mapping Web content [13], accept plain-text documents or web pages as the input for a geotagging algorithm. Geotagging plain-text documents involves some level of natural language processing (NLP) to accurately identify individual toponyms and reason about the relationships between them. In particular, the geotagging accuracy of these systems improves when incorporating the assumption of coherence between place names in several ways. For example, some systems attempt to infer a geographic focus of individual document sources, known as a *local lexicon* that can be used to resolve otherwise ambiguous toponyms [15]. In other work, incorporating the interpretation of toponyms that appear close together in text was shown to improve toponym resolution accuracy [9]. Additionally, some work has shown that sentence structure, such as place names appearing in comma-separated groups, can be utilized to improve accuracy [12]. However, all of these methods apply fairly loose definitions of consistency because plain-text documents are unstructured and heterogeneous.

In contrast to toponyms in plain-text documents, the toponyms that appear within a list or table column are much more likely to have strong consistency among their types, geographic containers, or prominence, or a combination of all three. Several systems support geotagging data from a spreadsheet or list, such as Google Fusion Tables[1] and Wolfram Alpha[2], along with special purpose systems such as MapAList[3] and BatchGeo[4]. Each of these systems performs well when fed documents with well-specified locations, such as addresses. But the results are poor for individual to-

[1] `http://tables.googlelabs.com/`
[2] `http://www.wolframalpha.com`
[3] `http://www.mapalist.com`
[4] `http://www.batchgeo.com`

Figure 2: Table with a location column containing L = [Alexandria, Arlington, Springfield, Vienna], geotagged by Wolfram Alpha. Wolfram Alpha interprets each toponym as the most populated place with the name, so "Alexandria" is associated with "Alexandria, Egypt", "Vienna" with "Vienna, Austria", "Arlington" with "Arlington, TX, USA", and "Springfield" with "Springfield, MO, USA".

Figure 3: List L, geotagged by our algorithm, which recognizes that the list of toponyms in L are likely to refer to a cluster of nearby cities in the American state of Virginia.

ponyms, as shown in Figure 2, which demonstrates the result of resolving each toponym individually, as done by Wolfram Alpha. This can be improved by categorizing possible interpretations, as shown in Figure 3. Additionally, there is some prior work that investigates methods for geotagging spreadsheets [11], by employing heuristics to determine if a collection of toponyms can be viewed as either (1) all prominent, (2) all nearby, or (3) all similar place types. This method uses thresholds to determine which places were prominent or nearby, which we aim to eliminate using a probabilistic Bayesian method.

3 Geotagging Lists and Tables

The table geotagging problem can be formalized as follows. Given a grid of data cells, each containing a character string, determine which cells contain geographic references (the *toponym recognition* task) and provide the most likely geographic interpretation for each selected cell (*toponym resolution*) in the context of the other cells in the grid.

Our method is motivated by the following model of how table authors construct lists and tables that contain geographic information. First, the author recognizes that one or more geographic locations are associated with each entity in the table. The entity for each row may itself be a geographic entity, but this situation need not be treated as a special case. After deciding on the geographic entities that will appear in the table, the author includes a column with the most descriptive geographic references for each entity. In some cases, the author includes multiple columns per geographic reference, with the additional columns providing geographic containers for entities in the primary geographic column. In other cases, the author assumes the contrast of nearby values can be used to disambiguate the toponyms [24]. In the table setting, the context comes in the form of coherent categories of toponyms within the geographic columns, such as "provinces in Canada", or "parks in Texas" or "prominent cities in Europe". There are multiple dimensions to these categories: the *feature type*, *geographic container*, and *feature prominence*.

Unfortunately, many places share names, a well-known geotagging challenge known as entity-entity ambiguity [14] or geo-geo ambiguity [2, 3]. Some place names, such as "Victoria", "Rome"/"Roma", and "San Antonio", are reused for hundreds of places in dozens of countries. Additionally, even within a small geographic area, a name can be used to describe a variety of places, such as "Rappahannock", which describes a county, cemetery, mountain, and river in the American state of Virginia. The process of resolving this ambiguity is known as toponym resolution [2, 11] and in this section we describe a method for toponym resolution in the context of tables. This is in contrast to the related problem of toponym recognition [8], where we are interested in determining whether a reference is to be interpreted as a toponym or not (i.e., is "Jordan" the name of a person or a location).

3.1 Problem Definition

We are given a data table $\mathcal{D}$ from a spreadsheet or HTML document that includes one or more columns of place names. The table contains a two-dimensional grid of data values, where $d_{i,j}$ represents the character string in the i-th column and j-th row of the data values. A gazetteer $\mathcal{G}$ is used to identify place names from the table. Each geographic entity $g_i \in \mathcal{G}$ is associated with several attributes by the gazetteer: *name*, *alternate names*, *feature type*, *geographic container*, *population*, and *coordinates*. The goal is to discover the mapping $F : \mathcal{D} \rightarrow \mathcal{G} \cup \varnothing$ that resolves each $d_{i,j} \in \mathcal{D}$ to a geographic entity $g_i \in \mathcal{G}$ or to nothing (indicating that the string value is not a reference to a place).

3.2 Proposed Method

To formalize our discussion, we define terms as follows. A toponym d is a character string with one or more possible geographic interpretations. The set of possible interpretations, $Geo(d) = \{g \in \mathcal{G} \mid g \text{ is a geographic interpretation of } d\}$, is determined by the entities of the gazetteer $\mathcal{G}$. For example, the interpretations of the string "Washington" include the city of Washington, D.C., the American state of Washington, the city of Washington, England, along with dozens of other, less prominent interpretations. The exact collection of interpretations depends on how strictly names are matched, such as whether gazetteer entries for "Mount Washington" or "Washington County" are included as interpretations for the string "Washington". Our method does not currently support loose matches such as these.

3.2.1 Extract Data Rows

In order to improve the quality and consistency of the data that our method is run on, we pre-process the input tables using a previously developed method based on Conditional Random Fields [1]. This pre-processing method makes it possible to handle spreadsheets and HTML tables that contain structures such as multiple header rows, sub-total rows, and notes/footnotes or non-relational rows. The input to the pre-processor is a table in either spreadsheet format (.xls) or an HTML file that includes TABLE elements. The pre-processor outputs a list of row classes that describe the functions of the rows within each table, which are then used to isolate the cells that contain data values from the other

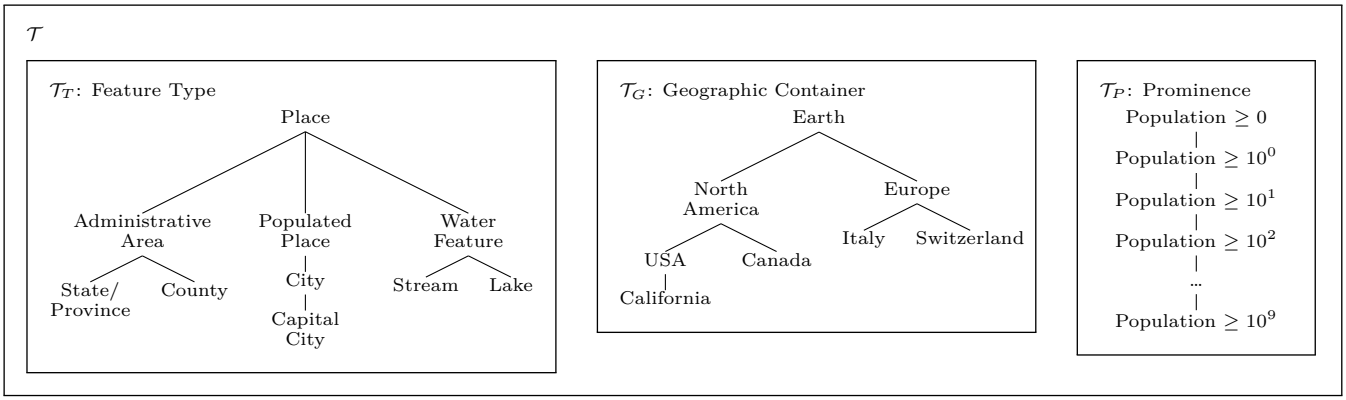

Figure 4: Simplified fragments of $\mathcal{T}$, the taxonomy for geographic entities that we derive from a gazetteer. The taxonomy is divided into three dimensions, $\mathcal{T}_T$, $\mathcal{T}_G$, and $\mathcal{T}_P$, which describe the feature type, geographic container, and prominence for geographic entities, respectively. Every geographic entity in the gazetteer belongs to a category $c \in \mathcal{T}_T \times \mathcal{T}_G \times \mathcal{T}_P$.

components of the table. The remainder of our method is applied to the data cells only.

3.2.2 Taxonomy for Geographic Entities

The properties that are provided by the gazetteer are used to generate a taxonomy $\mathcal{T}$ for describing geographic entities, which is partially depicted in Figure 4. The taxonomy is defined by hierarchies that represent properties along three orthogonal dimensions.

- **Feature Type**. The feature type describes the class of objects that an entity belongs to, such as "Capital City" or "Park" or "Stream" or "County". Feature types belong to a type hierarchy $\mathcal{T}_T$.
- **Geographic Container**. The geographic container is an administrative region in which the entity appears. Geographic containers belong to a geographic container hierarchy $\mathcal{T}_G$, in which counties or minor regions are contained by states or provinces, which in turn are contained by their countries. For example, a category could describe entities that are in South Africa, or in Shanghai, China.
- **Prominence**. For our purposes, an entity's prominence is derived from its population. Our formulation uses the $log_{10}(pop)$ as the prominence for a place with population *pop*. We view the prominence hierarchy $\mathcal{T}_P$ as having multiple levels, but no branches.

The tree structure of these hierarchies is not directly available from the GeoNames database. However, it can mostly be generated from the attributes of individual places. For the place types, we use GeoNames's *feature class* and *feature code* attributes and add an additional level by grouping similar types. We also augment the standard geographical containers with a level representing continents as the parents of country containers. We note that many geographic categorizations are handled by the geographic container dimension, but properly recognizing proximity relationships that cross borders is limited in the current approach. Proximity relationships lack the regular structure of a proper container hierarchy, which leads to an intractable number of possible categories (e.g., *"cities in Europe within 200 miles of Berlin"* where Berlin could be replaced with any location on the globe and the distance threshold could also vary). However an extension to our method that utilizes proximity relationships should be considered in future work.

We define $\mathcal{T}$ to be $\mathcal{T}_T \times \mathcal{T}_G \times \mathcal{T}_P$ and an element $c \in \mathcal{T}$ is called a *category*, which has three components, one for each dimension of $\mathcal{T}$. Each entity $g \in \mathcal{G}$ has a specific category. For example, "Franklin County" is a county in Ohio, USA with population > 1,000,000. The category for this entity, denoted $Cat(g)$, is <County, Ohio, Population $\geq 10^6$> with the English description *"counties in Ohio, USA, with population* $\geq 1,000,000$*"*. In addition, this entity could *satisfy* many other, less-restrictive categories, such as c' = *"places in USA with population* $\geq 10,000$*"*. The boolean function $Sat(g, c)$ is defined to be true if and only if entity g satisfies category c in this way.

3.2.3 Features

Next, we define two measures that can be used to estimate how well a list of toponyms is described by a specific category.

Coverage. The *coverage* of a category c over a set of column values D is defined as the fraction of values in the column with interpretations that satisfy the category.

$$Cov(D, c) = |\{d \in D \mid \exists g \in Geo(d) : Sat(g, c)\}| \,/\, |D| \quad (1)$$

For example, for D = [Washington, New York, Miami] and c = <City, United States, Population $\geq 10^6$>, we have $Cov(D, c) = 1.0$ because all entries in D are names of large cities in the United States. For c' = <State, United States, Population $\geq 10^6$>, $Cov(D, c') \approx 0.67$ because there is no state of Miami in the United States, so no interpretation of Miami satisfies the category.

Ambiguity. One way to differentiate between categories for describing a set of place names D is to estimate how specifically each category describes D. For example, the category <Place, Earth, Population ≥ 0> is satisfied by any valid set of place names. However, this is not a very specific category, and results in a lot of ambiguity when trying to resolve place names within the category. To encapsulate this concept quantitatively, we define the *ambiguity* of a category c over a set of string values D as the average number of interpretations for each string value that satisfy the category. That is, the ambiguity is equal to the total number of possible combinations of interpretations, normalized over $|D|$ (i.e., the geometric mean).

$$Amb(D, c) = \left(\prod_{d \in D} |\{g \mid g \in Geo(d), Sat(g, c)\}| \right)^{1/|D|} \quad (2)$$

As an example, the cities of Conway, Lockhart, Oakland, and Oak Ridge are suburbs of Orlando with populations greater than 1,000. However, there exist larger cities that share those names across the United States. In particular,

there are three cities named Conway, along with two each named Lockhart, Oakland, and Oak Ridge in the United States with populations greater than 10,000. Thus there are $3 \cdot 2^3 = 24$ possible combinations of interpretations for these place names in this category, resulting in an ambiguity value of $24^{1/4} \approx 2.21$. By contrast, if we look at the suburb interpretations, there is exactly one city of population greater than 1,000 with each of those names in Orange County, Florida, so that category has an ambiguity value of 1. Intuitively, we expect that the intended categories for a toponym lists have low ambiguity, all else being equal. However, rather than enforce the direction of the correlation, we leave it to our Bayesian model.

Algorithm 1 Given a column of toponyms, return set of place categories with coverage and ambiguity values.

```
 1: procedure FindCategories(D)
    input: List of toponyms D
    output: Set of categories C
 2:     C ← ∅
 3:     Initialize ColumnCounts_c ← empty list for all c ∈ 𝒯
 4:     for d ∈ D do
 5:         Initialize CellCounts_c ← 0 for all c ∈ 𝒯
 6:         I ← {i | i is a geographic interpretation for d}
 7:         for i ∈ I do
 8:             (c_T, c_G, c_P) ← GetSpecificCategory(i)
 9:             p_T ← GetAncestors(c_T) ∪ {c_T}
10:             p_G ← GetAncestors(c_G) ∪ {c_G}
11:             p_P ← GetAncestors(c_P) ∪ {c_P}
12:             for c ∈ p_T × p_G × p_P do
13:                 CellCounts_c ← CellCounts_c + 1
14:             end for
15:         end for
16:         for c ∈ 𝒯 where CellCounts_c > 0 do
17:             Append CellCounts_c to ColumnCounts_c
18:         end for
19:     end for
20:     for c ∈ 𝒯 where ColumnCounts_c is not empty do
21:         c.cov ← |ColumnCounts_c|/|T|
22:         c.amb ← (∏ ColumnCounts_c)^(1/|T|)
23:         C ← C ∪ {c}
24:     end for
25:     return C
26: end procedure
```

Our algorithm for computing the coverage and ambiguity of place categories for a given list of toponyms is shown in Algorithm 1. It takes a list of toponyms D as input, and returns a list of candidate place categories C, augmented with the coverage and ambiguity measures defined above. The algorithm begins by initializing the return set C (line 2) and ColumnCounts lists (line 3), which will accumulate a sequence of values for each category. The purpose of these values is to count the number of interpretations of each toponym that satisfy the category. The algorithm proceeds by iterating through each string value in D and determining how many interpretations of each string value satisfy the possible place categories. For each interpretation of a string value, the function GetSpecificCategory(i) returns the most specific satisfying category based on the gazetteer attributes and the components of that category are stored separately (line 8). Each component represents a node in one of the hierarchies of the taxonomy $\mathcal{T}$, and the function GetAncestors(n) returns all ancestors of a given node within the taxonomy (lines 9 to 11). The collection of ancestors within each dimension are combined using the Cartesian product to get the set of all categories that are satisfied by the interpretation being examined (line 12). This step makes use of the important property of our taxonomy that anything that satisfies a node n in one of the dimension hierarchies will also satisfy Parent(n). Once the full list of satisfying categories is computed, the number of interpretations is counted (line 13) and the counts are accumulated (lines 16 and 17). Finally, the accumulated counts are used to compute the coverage and ambiguity of each category with respect to the values in D (lines 20 to 23) and the resulting set of categories is returned (line 25).

3.2.4 List Categorization

Having computed $Cov(D, c)$ and $Amb(D, c)$ for every possible place category, we proceed by identifying which of those categories is most probably the correct one. A Bayesian classifier determines the estimated likelihood that each category c is the intended category for a set of string values D. Namely, for each $c \in \mathcal{T}$, we compute an estimate of $p(C_D = c \mid c, Amb(D, C), Cov(D, c))$, where C_D represents the intended category for the toponyms in D. In practice, many categories are not satisfied by any interpretation of any of the string values, so the space of possible categories is a small portion of the full category space. The example toponym list [Rome, Athens, Dublin] has at least one interpretation in 2,141 categories, which appears to be a large number for a three toponym list, but is dwarfed by $|\mathcal{T}|$, the total number of possible categories. To estimate each category's likelihood, the components of the category $c = <c_T, c_G, c_P>$ are treated separately so we have $l_c(D) = p(C_D = c \mid c_T, c_G, c_P, Amb(D, c), Cov(D, c))$. We want to separate the influence of each term on the final likelihood estimate, however the effect of each term is very dependent on the coverage value, so we apply a assumption of independence between all conditions except for $Cov(D, c)$. This allows us to estimate the joint probability as the product of four factors, $p(C_D = c|c_T, Cov(D, c)) \cdot p(C_D = c|c_G, Cov(D, c)) \cdot p(C_D = c|c_P, Cov(D, c)) \cdot p(C_D = c|Amb(D, c), Cov(D, c))$. We can employ the chain rule of probabilities to show that $p(A|B, C) = p(C|A, B)p(A|B)/p(C|B)$, which we apply to the individual factors of the likelihood estimate to transform them into a form that we can approximate using relative frequencies found in a training dataset.

Some transformations are performed to increase the generality of the training instances. First, the depth within $\mathcal{T}_G$, rather than the node itself, is used to match a category candidate with categories in the training data, which avoids geographic bias in our model. Second, the values of $Amb(D, c)$ are discretized in order to emphasize categories that are completely unambiguous (i.e., when $Amb(D, c) = 1.0$). Finally, the likelihood of a category coverage value, given one of the category components or the ambiguity value, is modeled as a truncated normal distribution over the [0, 1] interval, whose mean and standard deviation are computed from training data [7]. For our system, training data comprised a randomly selected set of 20 toponym lists along with their proper categories. The actual number of training instances is actually much larger than this, as the 20 lists each have interpretations in a large number of categories (between 280 and 6,097 categories per toponym list), where all but one serve as negative training examples.

As shown in Figure 5, the result of this process is a collection of potential categories with corresponding likelihood values. The final steps of our algorithm are to select the most likely category c based on the computed likelihood values and to resolve each toponym $d \in D$ by selecting the most prominent interpretation $g \in Geo(d)$ such that $Sat(g, c)$.

Location	Other Columns
Rome	...
Athens	...
Dublin	...

Category	Coverage	Ambiguity	Normalized Likelihood
country capitals with population $\geq$ 100,000 in Europe	1.00	1.00	70.13%
county seats with population $\geq$ 10,000 in Georgia, USA	1.00	1.00	15.07%
administrative regions with population $\geq$ 100,000 in Europe	1.00	1.26	13.88%
populated places with population $\geq$ 100 in Pennsylvania, USA	1.00	1.00	0.60%
populated places in Ohio, USA	1.00	2.15	0.05%
places in Missouri, USA	1.00	1.00	0.04%
farms in Limpopo, South Africa	1.00	2.47	0.04%
administrative regions with population $\geq$ 1,000,000 in Europe	0.67	1.41	0.03%
third-order administrative divisions with population $\geq$ 100,000 in Europe	0.67	1.00	0.03%
...	...	...	...

Figure 5: A sample table (left) and the resulting ranked list of column categories (right). The set of possible categories, along with their coverage and ambiguity values, is computed using the FindCategories algorithm. The likelihood values are computed by our method using a Bayesian classifier.

4 Evaluation

To evaluate the effectiveness of our table geotagging algorithm, we developed a system to process tables from the Web. In this section, we describe our dataset, analyze its spatio-textual composition, evaluate the effectiveness of our method for categorization, and compare its accuracy to that of alternative methods for table geotagging.

4.1 Dataset

Our experimental table dataset was selected from a large corpus of tables that were extracted from spreadsheet documents (in Microsoft Excel format) and HTML tables. The source documents were found on the Web using targeted search terms in search engines to find documents that were likely to contain relational tables (i.e., tables that contain data). The search terms made use of the "filetype" predicate to target Microsoft Excel XLS files and were appended with various terms such as "data" and "statistics". The search terms also contained random chaff values such as combinations of numbers and letters in order to uncover documents that might otherwise have been far from the front page of search results. The pre-processing phase described in Section 3.2.1 discards non-relational tables, such as spreadsheets containing calendars and forms or HTML layout tables, which allows us to focus on relational data tables.

We sampled 20,000 spreadsheets and 20,000 HTML tables from the table dataset. To avoid biasing our dataset towards large documents with many tables (i.e., spreadsheets with multiple worksheets or HTML documents with many TABLE elements), at most one table was selected from each document. As an initial filter, we identified columns that contain text values that match place names in GeoNames. To do so, we discarded any column containing fewer than three toponyms that matched GeoNames entities, within the first 100 values in the column. We then applied our algorithm to the remaining columns in order to identify those that can be described by a category from our taxonomy. This resulted in a collection of 12,861 columns from 8,422 tables. The automated processing of these columns by our algorithm results in a large array of categories, which we describe here.

The distribution of the place types (from $\mathcal{T}_T$) of columns in our dataset is shown in Figure 7 in the Appendix. As expected, the most common place types found in the columns of our table corpus are populated places (i.e., cities) and administrative areas (i.e., countries, states, provinces, counties, etc.). Other, less common place types we observed include schools; airports; country, state/province, and region capitals; continents; hospitals; and rivers and streams.

Many columns were classified with non-leaf types, which occurs when places in the column have a variety of specific types. For example, in American Baseball, there are some teams that represent states (e.g., Texas and Colorado, which are identified as administrative regions in GeoNames) and others that represent cities (e.g., New York and Chicago, which are identified as populated places), so a column containing these values would be categorized using the root node of the place type hierarchy. The root node represents generic places and was part of the assigned category for 1,640 columns according to our classifier. Another example of a generic place column is from a spreadsheet in our corpus that describes the itinerary of a trip to China, containing cell values such as "Great Wall of China", "Beijing", "Yangtze River", and "Shanghai".

Table 1: Dataset characteristics.

	Spreadsheets	HTML
Documents	20,000	20,000
Data column count	234,776	108,795
Data row count	11,984,929	992,309
Data cell count	122,291,632	5,053,901
Geographic tables	5,117	3,305
Geographic columns	7,072	5,789

Many place types are not found or are found only rarely in our table dataset. In most cases, these place types are relatively rare in Web tables, but in others the places are commonly represented in shorthand that is not found in GeoNames and consequently they are not identified by our algorithm. For example, some tables in our dataset contain information about National Parks, but use the park name without the qualifier, for example, "Yosemite" instead of "Yosemite National Park", and the GeoNames gazetteer does not contain this as an alternate name.

A large number of distinct nodes in the geographic container hierarchy $\mathcal{T}_G$ were present in the categories assigned to columns in our dataset. In total, 361 different geographic containers were chosen. The full distribution is too large display in full, but we note that 39.7% of all geographic containers were "Earth", 9.8% were at the continent level, 41.6% were at the country level, 7.4% were at the state/province level, and 1.5% were at the country/region level. One hundred seventeen countries were part of at least one geographic container, so the dataset was geographically diverse.

The final component of our taxonomy, the $\mathcal{T}_P$ prominence hierarchy, contains only 10 nodes, with the distribution shown in Figure 8 in the Appendix.

4.2 Category Accuracy

Next, we sampled from the full dataset to obtain a smaller dataset that could be hand-annotated by human judges and evaluated for correctness. To ensure that a wide variety of geographic columns were evaluated, columns with a variety of place types were chosen randomly. In total, 200 columns were chosen for evaluating the category classifications, balanced over different areas of the place type dimension $\mathcal{T}_T$.

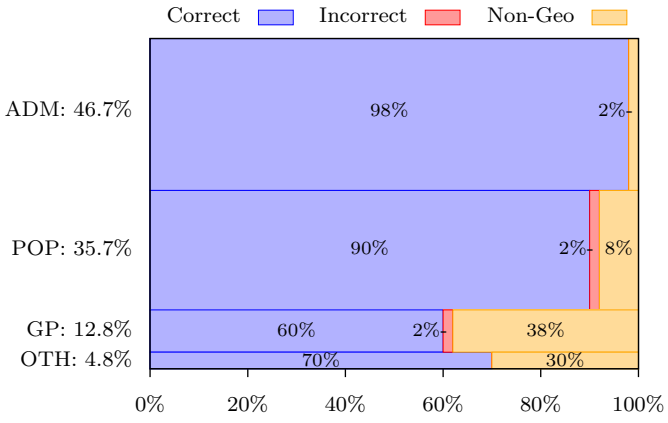

Figure 6: Accuracy of our algorithm for categorizing columns of toponyms. Bars are scaled horizontally to reflect the proportion of results within each group and scaled vertically to reflect the prevalence of each group within the full dataset.

Fifty columns each were chosen from the following groups:

- **ADM**: Administrative Features (or a descendant),
- **POP**: Populated Places (or a descendant),
- **GP**: Generic Places (i.e., the root of $\mathcal{T}_T$),
- **OTH**: Other places types (e.g., schools, airports, etc.).

For each column/category, a human judge determined whether the category that was returned by our algorithm matched the values in the column. Results are shown in Figure 6. Overall, 148 of the 200 columns were correctly categorized, with 2 mis-categorized, and 50 that were wrongly chosen as geographic columns. Columns that were categorized as Administrative or Populated Places achieved the best accuracy rates, with 49 and 45 of the columns given the proper categories, respectively.

Since the four groups make up different portions of the full table dataset, we can extrapolate the overall accuracy rate by incorporating the relative prevalence of each group throughout the larger dataset. We measured the fraction of the full dataset that was categorized into each of the four groups, and found that 46.7% of column categories are administrative (ADM), 35.7% are populated places (POP), 12.8% are generic places (GP), and 4.8% are other place types (OTH). Weighting the accuracy results by these proportions lets us estimate an overall accuracy rate of 88.9%.

Most errors were due to non-geographic columns being assigned a place category, which suggests that we can improve upon our toponym recognition phase to filter out words that are ambiguous. The most common non-geographic words that were interpreted as toponyms were proper names of people (15 columns). However, the results for columns that did contain toponyms were promising, with a total of only two incorrect categories out of 148 columns.

4.3 Toponym Resolution Accuracy

Our final experiment measures our toponym resolution accuracy. From the 200 categorized columns analyzed in Section 4.2, we selected the 148 true geographic columns. From each column, one cell value was picked at random for inspection. A human judge was presented with the other values in the column for context and was asked to choose the most likely interpretation out of all available interpretations for the string that were present in the gazetteer. If none of the interpretations were valid, but the string was indeed a toponym, the string was marked as an "unmatched toponym". Or if the string was not a toponym, the string was marked as a "non-toponym".

The results were compared to the toponym resolution output of three algorithms. The first, Prom, considers only the prominence of the possible interpretations when resolving each toponym. The second, 2D, is a combination of three classifiers that each only uses two of the dimensions in our taxonomy $\mathcal{T}$. Each classifier is trained separately with a subset of our feature set. To arrive at a resolution for toponyms, we choose the category with the highest likelihood value out of the three and pick the most prominent interpretation within that category. Finally, the third method, 3D, is our full method, which considers features from all three dimensions of the place taxonomy. The results of this experiment are shown in Table 2.

Table 2: Toponym Resolution Results.

Method	Accuracy
Prom	101/148 (0.682)
2D	130/148 (0.878)
3D	144/148 (0.973)

As expected, considering more dimensions improves the toponym resolution accuracy. The Prom method manages to resolve over two-thirds of the toponyms correctly, which is possible due to the large number of country, state, and metropolis occurrences in our dataset. The 2D variant improves upon this to achieve nearly 90% accuracy, since the addition of other attributes allows this method to recognize coherent types and geographically contained columns. Our full algorithm increases the accuracy rate further, where in all but 4 of the cases, the interpretation selected matched the interpretation that was assigned by our algorithm. This represents a 97.3% accuracy rate (144/148) for the toponym resolution task on geographic columns. In both columns that were assigned incorrect categories, the assigned interpretation did not match the ground truth. And in two other cases, the category was correct, but the toponym was still ambiguous within the category and a less prominent interpretation was the correct one (whereas out method chooses the more prominent interpretation in the face of ambiguity within a category). This result demonstrates the value of using our full taxonomy of hierarchical place categories for toponym resolution.

5 Conclusions

We introduced and studied the utility of combined hierarchical place categories for identifying and resolving toponyms in structured datasets. Lists and table columns containing spatio-textual references can be difficult to geotag correctly because standard contextual clues, such as geographic containers, are sometimes omitted when the table author expects the interpretation of the references to be clear from context. However, making use of the context that is present in a list or column of similar places has not been thoroughly studied before. Here, we take the approach that the common thread of a list of toponyms can have varying specificity over multiple dimensions, namely the place type of the locations, their geographic container, and their prominence. To address this, we showed how to construct a list of possible categories that can be used to describe the list of toponyms, along with several measures of each category's applicability. A Bayesian classifier is used to identify the most likely category based on observations made from a training data set. Our experimental analysis shows that the algorithm is effective at categorizing and resolving toponym lists that come from a large dataset of tables from the Web. As future

work, we plan to expand the category taxonomy by incorporating additional prominence information (including prominence measures for unpopulated places) as branches in the prominence hierarchy. Our work will also explore adding a proximity dimension to the category taxonomy in a way that searches for relevant proximity relationships, such as *"parks within 30 miles of Washington, DC"*, while maintaining a tractable number of candidate categories.

6 REFERENCES

[1] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *PVLDB*, 6(6), 2013.

[2] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: Geotagging web content. In *SIGIR*, pages 273–280, Sheffield, United Kingdom, July 2004.

[3] I. Bensalem and M. K. Kholladi. Toponym disambiguation by arborescent relationships. *Journal of Computer Science*, 6(6): 653, 2010.

[4] GeoNames. `http://geonames.org/`.

[5] G. R. Hjaltason and H. Samet. Speeding up construction of PMR quadtree-based spatial indexes. *VLDBJ*, 11(2):109–137, Oct. 2002.

[6] G. S. Iwerks, H. Samet, and K. Smith. Maintenance of spatial semijoin queries on moving points. In *VLDB*, pages 828–839, Toronto, Canada, Sept. 2004.

[7] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *UAI*, pages 338–345, Montreal, Quebec, Canada, 1995.

[8] M. D. Lieberman and H. Samet. Multifaceted toponym recognition for streaming news. In *SIGIR*, pages 843–852, Beijing, China, July 2011.

[9] M. D. Lieberman and H. Samet. Adaptive context features for toponym resolution in streaming news. In *SIGIR*, pages 731–740, Portland, OR, Aug. 2012.

[10] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling. STEWARD: Architecture of a spatio-textual search engine. In *GIS*, pages 25:1–25:8, Seattle, WA, Nov. 2007.

[11] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling. Spatio-textual spreadsheets: Geotagging via spatial coherence. In *GIS*, pages 524–527, Seattle, WA, Nov. 2009.

[12] M. D. Lieberman, H. Samet, and J. Sankaranayananan. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *GIR*, pages 6:1–6:8, Zurich, Switzerland, Nov. 2010.

[13] K. S. McCurley. Geospatial mapping and navigation of the web. In *WWW*, pages 221–229, Hong Kong, May 2001.

[14] D. Nadeau, P. Turney, and S. Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Canadian AI*, Quebec City, Quebec, Canada, June 2006.

[15] G. Quercini, H. Samet, J. Sankaranarayanan, and M. D. Lieberman. Determining the spatial reader scopes of news sources using local lexicons. In *GIS*, pages 43–52, San Jose, CA, Nov. 2010.

[16] H. Samet. K-nearest neighbor finding using MaxNearestDist. *PAMI*, 30(2):243–252, Feb. 2008.

[17] H. Samet and M. Tamminen. Bintrees, CSG trees, and time. *Computer Graphics*, 19(3):121–130, July 1985.

[18] H. Samet, A. Rosenfeld, C. A. Shaffer, and R. E. Webber. A geographic information system using quadtrees. *Patt. Recog.*, 17 (6):647–656, November/December 1984.

[19] 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. *CACM*, 46(1):63–66, Jan. 2003.

[20] J. Sankaranarayanan, H. Alborzi, and H. Samet. Efficient query processing on spatial networks. In *GIS*, pages 200–209, Bremen, Germany, Nov. 2005.

[21] J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Computers & Graphics*, 31(2):157–174, Apr. 2007.

[22] E. Tanin, A. Harwood, and H. Samet. A distributed quadtree index for peer-to-peer settings. In *ICDE*, pages 254–255, Tokyo, Japan, Apr. 2005.

[23] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *GIS*, pages 144–153, Irvine, CA, Nov. 2008.

[24] S. Winter and C. Freksa. Approaching the notion of place by contrast. *JOSIS*, 5:31–50, 2012.

APPENDIX

Figures 7 and 8 depict the distribution of node values in the type hierarchy $\mathcal{T}_T$ and the prominence hierarchy $\mathcal{T}_P$ for column categories in our evaluation dataset. The geographic container hierarchy $\mathcal{T}_G$ contains too many nodes to display in full; its distribution is discussed in Section 4.1.

```
Places ..... 1640
  Administrative ..... 693
    Political Entities ..... 540
      Independent ..... 1393
    Regions ..... 214
      Divisions ..... 15
      Level 1 ..... 2754
      Level 2 ..... 287
      Level 3 ..... 113
      Level 4 ..... 1
  Hydrological ..... 29
    Lakes ..... 4
    Reservoirs ..... 2
    Streams ..... 18
      Intermittent ..... 2
  Land Features ..... 38
    Areas ..... 10
    Continents ..... 10
    Parks ..... 3
    Reserve ..... 1
    Region ..... 1
      Economic ..... 1
  Populated
    Populated Places ..... 4348
      Capitals of Political Entities ..... 25
      Populated Localities ..... 6
      Seats of Level 1 Admin Regions ..... 89
      Seats of Level 2 Admin Regions ..... 106
      Seats of Level 3 Admin Regions ..... 8
      Sections of Populated Places ..... 6
  Roads or Railroads
    Streets ..... 6
  Spots, Buildings, or Farms ..... 157
    Air Transport Facilities ..... 26
      Airports ..... 18
    Buildings ..... 14
    Churches ..... 1
    Cemeteries ..... 2
    Farms ..... 7
    Gas/Oil Plants ..... 3
    Houses ..... 1
    Hospitals ..... 12
    Hotels ..... 10
    Military Installations ..... 2
    Libraries ..... 4
    Hydroelectric Plants ..... 1
    Railroad Stations ..... 1
    Schools ..... 215
    Triangulation Stations ..... 2
  Topographic ..... 13
    Islands ..... 6
    Mountain Ranges ..... 1
    Peaks ..... 1
    Ridges ..... 1
```

Figure 7: Place type distribution over nodes in $\mathcal{T}_T$ in the categorized table dataset. Values on the right indicate the number of columns that had the corresponding place type as part of their assigned category.

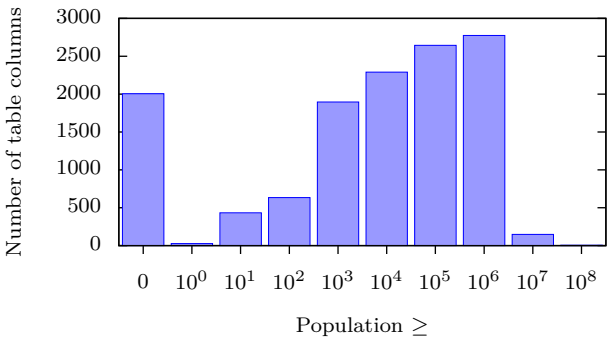

Figure 8: Prominence distribution over nodes in $\mathcal{T}_P$ in the categorized table dataset.