Exploiting geographical location information of web pages

Orkut Buyukkokten, Junghoo Cho, Hector Garcia-Molina, Luis Gravano*, Narayanan Shivakumar
Department of Computer Science
Stanford University, Palo Alto, CA 94305.
{orkut,cho,hector,gravano,shiva}@db.stanford.edu

Abstract

Many information sources on the web are relevant primarily to specific geographical communities. For instance, web sites containing information on restaurants, theatres and apartment rentals are relevant primarily to web users in geographical proximity to these locations. We make the case for identifying and exploiting the geographical location information of web sites so that web applications can rank information in a geographically sensitive fashion. For instance, when a user in Palo Alto issues a query for "Italian Restaurants," a web search engine can rank results based on how close such restaurants are to the user's physical location rather than based on traditional IR measures. In this paper, we first consider how to compute the geographical location of web pages. Subsequently, we consider how to exploit such information in one specific "proof-of-concept" application we implemented in JAVA.

1 Introduction

The world wide web (WWW) provides uniform access to information available across many web sites in several countries. Some web sites such as online stores [2, 4, 5] and banking institutions are of "global" interest to web users across the world. However, many web sites contain information primarily of interest to web users in a *geographical community*, such as the Bay Area or Palo Alto. In this context, a web search engine user usually finds sites of global interest, but finds it hard to quickly find information in or near a certain geographical region. For instance, finding restaurants, theatres and apartment rentals in or near specific regions is not a simple task in current web search engines.

Now consider the scenario we have a database with the geographical location (e.g., such as zip code or street address) of all "entities" (such as restaurants and theatres) with a web presence (i.e., has a web page or web site). We can then exploit such information in a variety of ways including the following.

- **Improving search engines:** When a web user queries a search engine for "Italian restaurants," the search engine first identifies where the user is from. For example, the user may have

*Currently affiliated with Computer Science Dept., Columbia Univ., 1214 Amsterdam Ave., New York, NY 10027.

pre-registered with the search engine his profile similar to `my.yahoo.com` or `my.excite.com`. In such a case, the search engine merely looks up a local database to identify the user's location. The search engine then uses this information to rank Italian restaurants (with a web presence) based on the distance of the restaurant from the user's location, rather than returning references to all Italian restaurants in the world. In case additional information such as the resturant quality or pricing information is available, the search engine can of course factor this into the ranking as well. We do not discuss these additional factors any further since the focus of this paper is on the as-yet-unexplored geography dimension rather than good ranking functions for restaurants.

Note that the above is not equivalent to the user querying the search engine for "Italian restaurant + Palo Alto" for the following two reasons. Firstly, this query will not return references to restaurants in adjacent areas such as Mountain View, Menlo Park or even in San Francisco. Secondly, ranking functions in current search engines are based on information-retrieval measures [3]. That is, the search engines will rank Italian restaurants based on the number of times the words "Italian restaurant" and "Palo Alto" occur on the web page. Clearly this is not equivalent to our approach of ranking restaurants based on the geographical distance.

- **Identifying "globality" of entity:** A web user often wants to find how "important" a web site or an entity is. For example, when a user is buying a book from a no-name bookseller on the Internet, the user wants to know if the bookseller is "big enough" so he can be comfortable giving his credit-card information. Also, the user often wants to visit the "authority" on a given topic, rather than a random web page on the same topic. For this reason, web search engines such as Google [1] count the number of distinct hyperlinks that point to a given web page as an indication of how important the web page is. The rationale for such a heuristic is that the larger the number of web users who made a hyperlink to the web page, the higher must be the importance of the page.

  While the above hyperlink count information is often useful for certain applications, we believe such a count is often not a good indication of the "global" relevance of a site. For instance, many Stanford University students and Palo Alto residents have links to Stanford Daily (the daily campus newspaper) since it covers news relevant to the local community. However, the number of hyperlinks to a "globally" important newspaper such as New York Times may be smaller simply because the concentration of web sites in Stanford University and Palo Alto is one of the largest in the world. We believe that by exploiting geographical information of web sites, we can estimate how global is a web entity. Conceptually, we can estimate the globality of an entity by computing the geographical distribution of the hyperlinks (e.g., how many states have hyperlinks to this entity?)

- **Data mining:** Market analysts are often interested in exploiting geographical information to target product sales. The web provides a new source of such information. For instance, if a market analyst identifies from web data that outdoor activities such as hiking and biking are popular among the residents of Bay Area and Seattle, the analyst may recommend opening additional sporting goods stores in these areas.

The following principal problems arise when we try to exploit geographical location information of entities (such as restaurants):

1. **How to compute geographical information?** First we need to compute geographical information of entities that have a web presence. We discuss in Section 2 some techniques for this problem. These techniques primarily differ in how *easy* it is to compute location information versus how *accurate* the corresponding information is.

2. **How to exploit this information?** Once we have a database of location information, the second question is how to use this information. We discussed earlier a few applications in which we can use such information. In Section 3 we concentrate on one such application and discuss an initial visualization prototype we developed for this specific task.

2 Computing geographical information

Ideally, we want to identify all "entities" with a web presence and their corresponding geographical locations. The simplest approach to computing geographical location of all entities is to have a human "hand-classify" entities in a variety of domains (such as companies, entertainment centers and restaurants). However this approach may take too long or may be too expensive. Automatically extracting entities from web pages is also a difficult task since it is hard to specify precisely what an entity is. In this context, we now consider the problem how to tag web pages and servers based on their geographical location approximately, but automatically.

- **Data mining:** We can automatically analyze web pages to identify geographic terms (e.g., Palo Alto), area codes or zip codes. When a user types in some search terms into a search engine, the engine can denote the geographical location in "close textual proximity" to the search terms to be the location of the search terms. For instance, consider some web page with the words "Florentine's Italian Restaurant, Menlo Park." If a web user in Palo Alto searches for Italian restaurants, we can associate Menlo Park to be the "closest" geographical location of the search terms "Italian restaurant" for that particular web page. The search engine can then rank the web page to be "close" to the user in Palo Alto.

  While the above idea is conceptually simple, it is often hard to implement for a variety of reasons. Firstly, it is often hard to automatically parse and extract geographical information with few false-positives and false-negatives. Secondly, this information requires us to download a web page to parse and extract geographic locations (as opposed to the techniques we consider next). Thirdly, many web pages have multiple restaurant names in many regions or discussing many geographical locations. Choosing the "right" geographical location may be tricky in some of these scenarios.

- **Network IP address:** Now consider the case we have a database with the location of each web site in the world. We can then mark each web page in that site to be relevant to the corresponding location. For instance, consider the web page at `http://www-db.stanford.edu/~shiva`. From the URL, we can query the database and assume the web page author and the content of the page is relevant to Palo Alto or Stanford regions. Unfortunately, we have not found such a database despite repeated requests to INTERNIC and ICANN, the Internet domain registries. However, we can use UNIX tools such as `whois`, `nslookup` or `traceroute` to identify `www-db.stanford.edu` to be located in Palo Alto, California.

This approach suffers from three problems. Firstly, the above heuristic does not work in all cases. For instance, a web page located in `www.aol.com` often has no correspondence to where the web server is located at. Secondly, while `http://www-db.stanford.edu/~shiva` is located in Stanford, the content may be globally relevant. (We will consider one way of handling this problem in the next section.) Finally, IP addresses are often allocated in an ad-hoc fashion and there is usually no inherent connection between an IP address and its physical location. In this context, tools like `whois`, `nslookup` and `traceroute` do not always give us the location of a site [6]. Also these tools are slow and typically take between a second and ten seconds to respond per web site. Such tools will not scale when we need to find the locations of hundreds of thousands of web sites. In the next section, we show how we can approximate some of the location information accurately and efficiently (especially for educational institutions with a `.edu` extension).

3 Prototype application

We discussed in Section 1 a variety of applications that benefit from geographical location information. As proof of concept, we developed a prototype to test our hypothesis that geographical information is indeed a good indicator of the "globality" of a web site. We built our prototype using the following databases that we downloaded from the Internet. Recall from previous section that tools like `whois` are too slow when we need to compute the geographic locations of hundreds of thousands of web sites.

1. **Site-Mapper:** We downloaded from `rs.internic.net` a database that has the phone numbers of network administrators of all Class A and B domains (in case of routing problems with that domain). From this database, we extracted the area code of the domain administrator and built a `Site-Mapper` table with area code information for IP addresses belonging to Class A and Class B addresses. That is, the original database contained a phone number with a (650) area code for the administrator of the 171.0.0.0 (Stanford) Class A domain addresses. Based on this information, we maintained in `Site-Mapper`, the area code information for all the `171.*.*.*` IP addresses. Similarly for Class B addresses.

2. **Area-Mapper:** We downloaded from `www.zipinfo.com` a database that mapped cities and townships to a given area code. In some cases, entire states (e.g., Montana) correspond to one area code. In other cases, a big city often has multiple area codes (e.g., Los Angeles). We wrote scripts to convert the above data into a table with entries that maintained for each area code the corresponding set of cities/counties.

3. **ZipCode-mapper:** We downloaded from `www.zipinfo.com` a database that mapped each zip code to a range of longitudes and latitudes. We also downloaded a database that mapped each city to its corresponding zip codes.

From the above databases, we computed for each IP address the "average" latitude and longitude of where the web site was located. For instance, when an area code covers an entire state, the longitude and latitude for an IP address in that state is the longitude and latitude averaged across the extent of the state. As we discussed earlier, the above process does not always yield the correct

answer for arbitrary IP addresses. However we found that this technique works very well in practice for educational institutions (with `.edu` extensions) since universities have all the IP addresses in a Class A or Class B domain allocated to web sites within the campus. Hence by restricting our database to educational institutes, our database has accurate geographical information for web pages located in these institutes.

Given the above geographical database, we built a GUI in JAVA that performed the following tasks on a map of the United States.

- Given any latitude and longitude, the prototype places a point at the corresponding location on the map.
- Given a city name, the prototype places points on the corresponding locations on the map. (Many cities have the same name.)
- Given an IP address or the corresponding textual URL, the prototype places the site on the corresponding geographical location on the map.

With the above infrastructure in place, we built our prototype on top of the Google web search engine [1]. When a user types in a URL into our system, the prototype issues the "link:" query to Google through `http GET` requests, to retrieve the set of pages in educational sites that have a hyperlink to the given URL. For instance, when a user types in `www-db.stanford.edu`, the prototype extracts from Google all web pages (from `.edu` sites) that have a hyperlink to the given URL. Our prototype then maps each page so extracted onto its corresponding geographical location on the map. In Figure 1 we see a sample screenshot from our system when the given URL is `www.sfgate.com`, the online website for the San Francisco Chronicle. Our prototype draws "filled" circles for each geographical location with links to `sfgate.com`, where the radius of the circle indicates the number of links from that location normalized to the total number of links to `sfgate.com`. When the circles exceed a certain maximum radius, we draw these circles as "unfilled" so we can see other smaller circles that may otherwise be covered by the larger circle. From the screenshot, we see that the Chronicle is popular primarily in the Bay Area and California and has a few readers distributed across other states. In Figures 2 and 3 we show similar screenshots for the New York Times and the daily campus newspaper in the University of Minnesota. We can see from these screenshots that the New York Times is a more "global" newspaper than the Chronicle given its readership across more states, while the daily newspaper in Minnesota is a very popular community newspaper in Minnesota.

We tried similar queries on many online newspapers and observed similar results. Also we issued queries to identify which regions pointed to our Database group web page, and could visually identify other database research groups across the country. Similarly, we observed "close correlations" between `www.stanford.edu` and `www.berkeley.edu`, presumably due to the close interactions between the two universities. We also tried other queries in other domains such as `www.playboy.com` and a few Gay and Lesbian web sites. In some of these cases, we observed a few states with unusually high number of links to these sites.

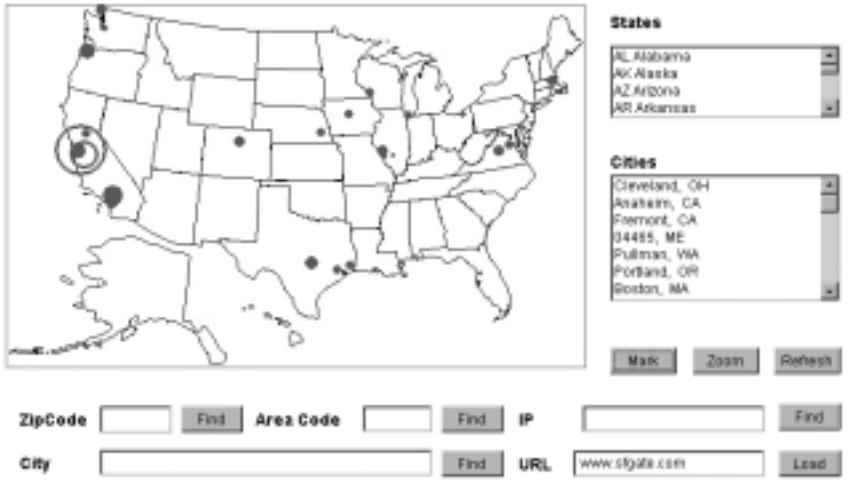

Figure 1: Geographical distribution of hyperlinks to `www.sfgate.com`.

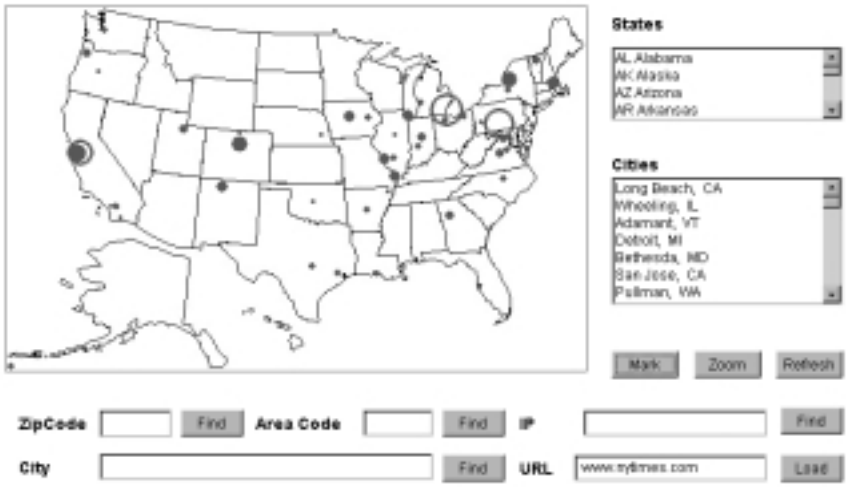

Figure 2: Geographical distribution of hyperlinks to `www.nytimes.com`.

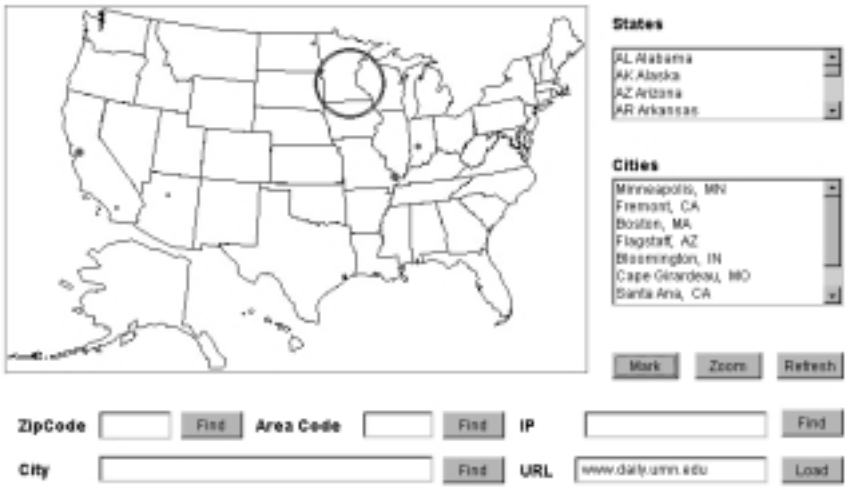

Figure 3: Geographical distribution of hyperlinks to `www.daily.umn.edu`.

4 Conclusions

In this paper, we make a case for exploiting geographical information to improve web search engines and for data mining applications. We discussed a few approaches to compute the geographical location of web sites, and discussed our prototype that visually indicates the "globality" of web sites based on the geographical distribution of web pages that point to the web site. From this prototype, we see that we can build useful applications that exploit geographical information in a meaningful fashion, even if we have geographical information of only educational institutes.

In the future, we plan to explore how to identify geographical locations of web pages and entities using the content-based data mining approaches we mentioned briefly. Specifically, we plan to compare the data mining approach against the network IP address based approach we adopted until now, and understand how the techniques differ. For this, we have built a web crawler in our research group that has an archived repository of over 30 million web pages. We plan to implement and run our geography extractors on this data soon.

References

[1] S. Brin, L. Page. Google Web Search Engine. `www.google.com`

[2] Amazon Bookseller. `www.amazon.com`

[3] G. Salton. Introduction to Modern Information Retrieval. *McGraw-Hill, New York*, 1983.

[4] Barnes and Nobles Bookseller. `www.barnesandnobles.com`

[5] Jango Comparison Shopper. `jango.excite.com`

[6] Uri Raz. Finding a host's geographical location. http://t2.technion.ac.il/~s2845543/IP2geo.html