Polytechnic
UNIVERSITY
Brooklyn · Long Island · Westchester

Design and Implementation of a Geographic Search Engine

Alexander Markowetz Yen-Yu Chen Torsten Suel
Xiaohui Long Bernhard Seeger

Department of Computer and Information Science

Technical Report
TR-CIS-2005-03
02/26/2005

Design and Implementation of a Geographic Search Engine

Alexander Markowetz [1] Yen-Yu Chen [2] Torsten Suel [2]
Xiaohui Long [2] Bernhard Seeger [3]

ABSTRACT

In this paper, we describe the design and initial implementation of a geographic search engine prototype for Germany, based on a large crawl of the de domain. Geographic search engines provide a flexible interface to the Web that allows users to constrain and order search results in an intuitive manner, by focusing a query on a particular geographic region. Geographic search technology has recently received significant commercial interest, but there has been only a limited amount of academic work in this direction so far. Our prototype performs massive extraction of geographic features from crawled data, which are then mapped to coordinates and aggregated across link and site structure. This allows us to assign to each web page a set of relevant locations, called the geographic footprint of the page. The resulting footprint data is then integrated into a high-performance query processor on a cluster-based architecture. We discuss the various techniques, both new and existing, that are used for recognizing, matching, mapping, and aggregating geographic features, and describe how to integrate geographic query processing into a standard search engine architecture and search interface.

1. Introduction

The World-Wide Web has reached a size where it is becoming increasingly challenging to satisfy certain information needs. While search engines are still able to index a reasonable subset of the surface web (i.e., excluding Deep Web content), the pages the user is really looking for may be buried under hundreds of thousands of less interesting results. Thus, search engine users are in danger of drowning in information. Adding additional terms to standard keyword searches often fails to drill the iceberg of results that are returned for common searches. A natural approach is to add advanced features to search engines that allow users to express constraints or preferences in an intuitive manner, resulting in the desired information to be returned among the first top results. In fact, search engines have added a variety of such features, often under a special *advanced search* interface, though mostly limited to fairly simple conditions on domain, link structure, or last modification date.

In this paper we focus on how to constrain web queries geographically. Geography is a particularly useful criterion, since it most directly affects our everyday lives and thus provides an intuitive way to express an information request. In many cases, a user is interested in information with geographic constraints, such as when looking for local businesses, locally relevant news items, or tourism information about a particular region. When taking up yoga, web sites of local yoga schools are often of much higher interest to the user than those of the world's ten biggest yoga schools.

We expect that *geographic search engines*, i.e., search engines that support geographic preferences, will have a major impact on search technology and associated business models. First of all, geographic search engines provide an extremely useful tool. They allow a user to express in a single query what might take multiple queries under traditional search engines. For example, when looking for a yoga school in or close to Brooklyn, a user of a traditional engine may end up trying queries such as

- `yoga ''new york''`
- `yoga brooklyn`
- `yoga ''park slope''` (a part of Brooklyn)

but even this might yield inferior results as there are many ways to refer to a particular location, e.g. numeric codes, and since the engine has no notion of geographical closeness, e.g., a result across the bridge to Manhattan might also be acceptable. Second, geographic search is a fundamental technology for *location based services*, including electronic commerce via cell phones and other mobile devices. Third, geographic search supports locally targeted web advertising, thus attracting advertisement budgets of businesses with a local focus, like pizza shops, that would otherwise be spent on leaflets and local print ads. Other opportunities arise from mining geographic properties of the web, e.g., for market research and competitive intelligence.

Given these possibilities, it comes as no surprise that over the last year leading search engine companies such as *Google* and *Yahoo* have made significant efforts to deploy their own versions of geographic web search. Our approach and focus differs, both in the way geographic information is extracted from the web and, as far as we can tell, in the way it is integrated into query processing. In particular, the commercial engines appear to focus on matching pages with data from business directories in order to support search for local businesses and organizations. While this is an important part of geographic search, we focus on the use of geographic constraints in more general information requests. A user may not just be interested in finding businesses listed in yellow pages, but may have broader interests that can best be satisfied by private or at least non-commercial web sites, such as local news and cultural events, or the history or geography of a certain area. In order to facilitate such queries, we try to extract geographic markers, such as addresses or phone numbers, from all web pages, independent of their listing in a business directory. To extend search capabilities to those pages that contain no such markers, we employ a combination of new and previously proposed techniques based on link and

[1]Department of Computer Science, University of Science and Technology, Kowloon, Hong Kong. alexmar@cs.ust.hk

[2]Department of CIS, Polytechnic University, Brooklyn, NY 11201. {yenyu@photon.poly.edu, suel@poly.edu, xlong@cis.poly.edu}. Research supported by NSF CAREER Award NSF CCR-0093400 and the New York State Center for Advanced Technology in Telecommunications (CATT) at Polytechnic University.

[3]Department of Mathematics and Computer Science, Philipps Universität, Marburg. seeger@informatik.uni-marburg.de

Technical Report TR-CIS-2005-03, Department of Computer and Information Science, Polytechnic University, Brooklyn, NY 11201.

site structure.

Before continuing, we briefly outline our basic approach. Our system is a crawl-based engine that starts by fetching a subset of the web for indexing and analysis. In our case we focus on Germany and crawl a subset of the `de` domain; in cases where the coverage area does not correspond well to any particular domain, smarter focused crawling strategies may be needed. Afterwards, a standard text index is built. In addition, data extraction and mining is used to assign a set of relevant locations to each page, called a geographic footprint. Finally, search queries consisting of textual terms and geographic areas are evaluated against the index and footprint data using an appropriate ranking function. The goal of this project is to put to practice and further develop some ideas that had been laid out in earlier work such as [24, 13, 23], by building a complete prototype for testing and evaluating the various approaches. Our contributions are:

- We provide the first description in the literature of an actual implementation of large-scale geographic web search. The current prototype, which is close to completion, is based on a crawl of over 30 million pages in the `de` domain, with plans to expand to a larger set soon.
- We combine multiple previous and new techniques for deriving geographic data from the web, using features such as town names, zip codes, phone numbers, link and site structure, or external databases such as `whois`.
- We map the set of geographic features in each page to a simple and highly compressible geometric representation that is used during link and site analysis and query processing.
- We provide the first discussion of efficient query execution in large geographic search engines.
- We discuss a range of geo coding techniques, with a special focus on disambiguation, and describe their implementation and interplay. We particularly cover compound town names that are common in Germany but also other countries.

This paper is organized as follows. Related work is discussed in the next section, and Section 3 describes the web crawl data and external geographic databases that we used.

In Section 4, we introduce the *geographic footprint* of a document, and show how to create this footprint in a process called *geo coding* over several steps. In the *geo extraction* step, we extract features that are likely to have geographic meaning from the documents. The mapping of these terms to actual geographic entities, such as towns or regions, and their coordinates in the next step is called *geo matching*. This gives us an initial geographic footprint for a good fraction of the documents, which we store in a simple and highly compressible geometric format. We then perform *geo propagation* across link and site structures to increase the quality and coverage of the results.

In Section 5 we show how to use our geo-coded web crawl to build a geographic search engine. We address efficient query execution as well as user interface issues. Finally, we provide some concluding remarks and discuss future work on geographic search and geographic web mining.

2. Related Work

We now discuss related work. We start with *geo coding*, the process of determining the set of locations a web page is about. We then describe existing geographic search engines in Subsection 2.2, and discuss approaches based on the Semantic Web in Subsection 2.3. Finally, we discuss the role of the physical locations of server and client devices in geographic search.

2.1 Geo Coding

There have been several previous papers on geo coding. A good first overview on geographic hints that are commonly found in web pages is provided by McCurley in [24], which introduces the notion of *geo coding*. It describes the various geographic indicators found in web pages, such as zip codes or town names, but does not give any detailed solutions on how to extract them and in particular resolve the resulting ambiguities and other issues when mapping names to actual towns.

The authors of [4, 13] introduced the idea of a web page's *geographic scope*, i.e., the area it addresses in terms of readership. In the first step, their technique assigns a position to every web site based on its `whois` entry. In the second step, a fixed hierarchy of administrative regions is superimposed on the entire area, and link analysis is performed with respect to these regions. If a reasonable number of links from a region point to a web site, and if these links are homogeneously distributed, then the region is included in the geographic scope of the site. The approach was applied to the United States using states, counties, and cities as the objects in the geographic hierarchy. This technique is related to the approach that we propose in Section 4.6, which is essentially a generalization and refinement, but differs in several ways. First, the density measure that is used assumes that a site has a fairly significant number of incoming links, and thus it does not work well for pages and sites that have a more moderate in-degree. In general, the approach is fairly coarse-grained as it focuses on sites instead of individual pages and on relatively large geographic regions. The experimental evaluation in [13] is limited to pages in the `edu` domain where `whois` provides a good first estimate of the location of a web page; thus [13] does not address how to deal with more noisy location data for arbitrary pages by analyzing additional features such as page content.

The approach that is closest to us is the very recent work in [1]. It uses a hierarchical gazetteer with 40,000 towns and exclusively considers these town names. It is designed to perform geo coding for the entire globe, looking for names of towns with more than 500,000 people. Decreasing the minimum size for recognizable towns to 5,000 is reported to have a positive effect. Similar to our *nearby town* measure, it uses the gazetteer's hierarchy for disambiguation when there are several towns of the same name, but the size of towns is not considered for this case.

Similar to our geographic footprint, [1] focuses on a document's *geographic focus* rather than the more specialized *geographic scope* of [13]. In contrast to our system, the geographic focus of a page is not given in geographic coordinates, but tied to a node in the hierarchical gazetteer. There can be several foci for a document, although the authors explicitly seek to avoid these situations by grouping, while we prefer to preserve such "fuzziness" until query processing. The system is evaluated on 20,000 documents by comparing it against entries from DMOZ [26], and also on three smaller hand-tagged corpora, resulting in accuracies between 60% and 80%.

Many approaches, such as [13] or our work, exploit data from the `whois` directory. By comparison against a business directory, [33] showed that `whois` entries mostly can be relied on to accurately point to the address of the business that registered the domain.

2.2 Geographic Search Engines

There are several geographic search engines that are already available online. Some are small-scale academic prototypes, based either on small specialized collections or a meta search approach. In particular, [19] performs automatic geo coding of documents based on the approach in [13]. Most other small prototypes, such as [8], require pages either to carry special markup tags or to be manually

registered with the search engine.

There are several lesser-known geographic search engines by commercial players. Some, such as the extension to the *Northern Light* engine by *Divine* [14], have already disappeared again. Others such as [16] rely on geographic meta data in the pages, or query internet directories such as *DMOZ* [26]. The scalability of these approaches in terms of user population is doubtful as they are highly susceptible to manipulation through misleading meta data. Of all current geographic search engines, the Swiss *search.ch* engine [29] is closest to our approach. It has been around for quite a while and allows users to narrow down their initial search by specifying a more and more focused location, over several hierarchical levels such as cantons (states) and cities within Switzerland.

As mentioned already, geographic search has recently received a lot of attention by the major search companies. Both *Google* and *Yahoo* have introduced localized search options [18, 32]. The two approaches appear to be similar to each other and quite different from what we propose. Both seem to rely on an intermittent business directory, where a user first retrieves entries for businesses that satisfy certain key words and are close by, and then can extend the search to actually retrieve pages about these businesses or about the area they are located in. The exact algorithms used by these services are not publicized.

There are two main differences to our approach. First, the intermittent business directory narrows the search results to information relating to commercial or other entities that are registered or known to the business directory. This eliminates a lot of private noncommercial content, unless it directly relates to one of these entities (in which case it can be retrieved in the second phase). Second, the location is modeled down to very precise coordinates, i.e., the street address of the businesses, which can then be displayed on detailed street maps. However, there seems to be no mechanism to model geographic footprints of pages that cover larger areas, such as a county or state, which do not map to this business-oriented model.

2.3 Geographic Semantic Web

Extending the Semantic Web to a *Geographic Semantic Web*, such as proposed in [15], seems like a natural approach. Every web page would simply contain some meta data to explicitly store its geographic footprint. There are a number of simple models such as [3] or [9] that are already available. Other models from the GIS community, such as GML from the Open GIS Consortium [7], could also easily be adapted.

There are however two major problems, both inherent to the Semantic Web as such, that we believe render this approach infeasible for general web search applications, though it may be useful for other more controlled scenarios. First, this approach runs into a *chicken and egg* dead-lock situation. Web authors will only include meta information if search engines make use of them. Search engines however will wait for a sufficient amount of web authors to provide meta information, before building any services based on such meta information. Second, Web authors are not to be trusted, as they frequently provide misleading information to manipulate search engines. For this reason current search engines already pay little attention to existing `meta` tags.

2.4 Geo Coding Based on Hardware

There are a number of mechanisms for estimating the physical location of a web server or a client; however, this is only of very limited relevance to our scenario. On the server side, most web pages today are hosted in server farms often hundreds of miles away from their author and with no relationship to the geographic regions they relate to. In the case of the `de` domain, a large percentage of the entire content, including most small-business sites, is served by two large hosting companies located in Berlin and Karlsruhe. On a very high level, there is some correlation between the location of the server and the geographic focus of a web page, since a site in the `de` domain is more likely to be hosted in Germany than Romania. But on a regional level there seems to be less correlation, with the exception of larger organizations such as universities.

One mechanism, which one could call "semantic hardware", proposes annotating DNS with geographic information regarding the positions of hosts [10]. On the experimental side, the authors of [28] performed several studies where they try to infer the geographic position of a host from the network topology, using data such as host names, IP addresses, and round trip time measurements. Several commercial players such as [12] use similar technology to trace the geographic positions of end users, mainly to enable content providers to comply with different national legal restrictions or language requirements, or for more efficient content distribution.

Other solutions exist for locating mobile clients in wireless and cellular networks. In our application, a user specifies the geographic scope of a query via some mechanism in the user interface, either through keywords or by clicking on a map. In a mobile scenario one might assume that the user is interested in the area surrounding their current location, as determined by the mobile provider. However, this is entirely an interface issue and does not impact geo coding or query processing.

3. Underlying Data

Any crawl-based search engine is limited to the underlying data. We now briefly describe the data that was available for our prototype, both the crawl and geographic databases for Germany.

Using the *PolyBot* web crawler [30], we acquired about 31 million distinct web pages from the `de` domain over a few weeks in April 2004, using Yahoo's German portal as seed pages. One problem for Germany are *Umlauts* which `HTML` can represent in multiple ways that had to be standardized in a preprocessing step.

Apart from the personal background of some of the team members, we chose the `de` domain for two reasons. First, it is the right size both geographically and in terms of number of pages. It is quite dense with about 7.8 million `de` domains registered within the relatively small area of Germany. It is also reasonably self-contained in terms of link structure due to language issues. Thus, the domain provides a very nice test bed that is large enough to be meaningful and challenging, but also not outside the reach of an academic research prototype. The `de` domain was estimated in 2000 at 3.67% of the entire web [2]. This would translate to about 150 million pages to achieve the same density of coverage as the 4 billion pages of the *Google* engine (as of November 2004) on the entire web; this number is within reach of our current hardware setup.

A second reason was the availability of geographic databases for extracting and matching geographic terms. In particular, the `whois` entries for `de` domains are usually complete and well structured and enabled us to extract the necessary information with little effort. We retrieved 680,000 `whois` entries for all the domains our crawl had touched; many of the 7.8 million registered domains do not actually have a live web server. We also had access to several other sources of geographic data for Germany, and even more important an understanding of the language, administrative geography, and conventions for referring to geographic entities.

We use two geographic data sets for Germany, one for 5,000 telephone area codes, the second for 82,000 German towns. The first maps every area code to *one* city and also to the coordinates of

the centroid of the region that the code covers. Since an area code may sometimes span multiple cities, this information was not completely accurate but still useful. The second mapped zip codes to all the towns they cover, and these towns to their geographic centroids. If the town was a village, it was also mapped to the associated city. This data set originated from a GIS application, where geographic positions are the database keys and town names are only used for display to the user. The names were therefore often misspelled or abbreviated in various nonstandard ways, requiring painstaking manual cleaning over several days.

4. Geo Coding

The process of assigning geographic locations to web pages that provide information relevant to these locations is called *geo coding*. A document can be associated with one or a number of locations, for example when a company web page contains addresses of several different outlets. We call this collection of locations the page's *geographic footprint*. For every location found in a page, an integer value is assigned that expresses the *certainty* with which we believe the web page actually provides information relating to the location. (Thus, a complete address at the top of the page might result in a higher certainty than a single use of the town name somewhere in the text of the page.)

In our approach, we divide geo coding into three steps, *geo extraction*, *geo matching*, and *geo propagation*. The first step extracts all elements from a page that might indicate a geographic location, including elements in URLs. The second step tries to make sense of these by mapping them to actual locations, i.e., coordinates, and leads to a first initial geo coding of a body of documents. For these first two steps, we make use of databases of known geographic entities such as cities or zip codes, as described in Section 3. Our architecture also allows us to incorporate information from third party directories such as DMOZ or yellow pages. In the third step, we perform *geo propagation* to increase the quality and coverage of the geo coding through analysis of link structure and site topology. We take a conservative approach during the extraction and matching steps: if in doubt, we rather drop a hint for a location than include it. The underlying assumption is that for the same area and most common search terms, there is an abundance of other documents that contain the same key words and are more likely to be relevant to the location. Before we proceed with the description of our geo coding process, we first introduce our representation of a document's geographic footprints.

4.1 Geographic Footprints of Web Pages

As for all Geographic Information Systems, the first basic design decision has to be made between a vector data model and a raster data model that maps all data onto a discrete grid. A document may contain several geographic hints, some of which refer to point positions, while others such as zip codes refer to polygonal areas. Thus, our data model has to handle both types.

After some consideration, we decided to use a raster data model that represents geographic footprints in a bitmap-like data structure. In comparison to the alternative, we lose some precision by pressing the information into the grid. If we make the grid fine enough however, then the degree of imprecision is small, especially when compared to other uncertainty factors in the data extraction process. In our case, we superimposed a grid of 1024×1024 tiles, each covering roughly a square kilometer, over Germany, and stored an integer value (certainty) with each tile that expresses the likelihood that the document is relevant to this tile.

Such a representation provides two advantages. First, it allows us to define and efficiently implement some basic aggregation operations on footprints. Thus, if a page contains several geographic features, then the footprint for the page is defined by combining the footprints for the individual features, i.e., adding up the amplitudes of the corresponding tiles, possibly after some normalization. Such operations will be very useful during geo propagation and query processing. Second, since for most documents only a few of the tile values are non-zero, we can efficiently store the footprints in a highly compressed quad-tree structure. Moreover, we can use lossy compression (smoothing) on such structures to further reduce their size with limited loss in precision, which will be important for efficient query processing. We note that our compressed footprint structures are also reminiscent of two-dimensional histograms as studied in the context of database systems.

We implemented a small and highly optimized library for basic operations such as footprint creation, aggregation, simplification (smoothing), and intersection (for query processing) based on quad-trees. Concerning the need for more precise locations such as exact addresses of businesses that can then be displayed to the user on a street map, we note that such information can always be stored separately for this purpose. Our focus here, as discussed earlier, is not on simple yellow page operations but more general classes of geographic search operations. Our grid model is particularly useful for the geo propagation and query processing phases, where exact locations are not that helpful.

4.2 External Databases

In addition to city names, addresses, and phone numbers extracted from page content, we could use other external databases as sources for geo coding. These fall into three categories: business directories, web directories, and the `whois` directory.

The first category maps businesses and thus their associated web sites to addresses, which in return map to geographic positions. Some geographic search engines such as those of *Google* and *Yahoo* [18, 32] appear to make heavy use of business directories. The main problem of business directories is also their biggest advantage. They cost money, and therefore usually list commercial companies exclusively, ignoring private web sites or web sites of nonprofit organizations. The registration fee however also results in less spam and higher data quality.

Web directories such as Yahoo [31] or the Open Directory project [26] can also provide geographic information about web sites, since most of them also categorize sites by region. Such directories are challenging to create and maintain, and often out of date and not as reliable as commercial directories. However, they can be useful as an additional ingredient in disambiguation and in geo propagation via link structure.

The `whois` directory turns out to be a very good source of geographic information, though not as accurate as the above business directories. The `whois` directory is an integral part of the Internet infrastructure and freely accessible, though usually only via many individual lookups. For every registered domain, it contains three sections with contact information for the content provider, server administrator, and network administrator. For most commercial sites, the section with the contact regarding content (`admin-c`) points to the address of the company that registered the domain. An earlier study [33] showed a high accuracy for `whois` entries.

The other two sections of the `whois` entries are rather useless for our purpose as we are not interested in finding out where servers are located. In Germany, roughly 70% of all `de` domains are hosted in just two server farms; thus at least 70% of all `whois` entries differ only in the `admin-c` section. We point out that `whois` entries for different top-level domains differ greatly in quality. For the `de` domain, they are highly structured and usually complete, with pre-

cise addresses and even phone numbers in the different sections. Entries for the uk domain in contrast usually contain very little of such information and are often not very structured. Entries for the com domain tend to be highly unstructured, but generally quite rich in information.

In Section 4.6.1 we show where to plug information from such databases into our geo coding process.

4.3 Germany's Administrative Geography

Effective geo coding requires some understanding of a country's *administrative geography*, and of conventions for referring to addresses and other geographic entities. Thus, one has to know how names are composed, how different towns relate to each other, what the role of the states and counties is, and where postal or area codes fit into the picture. Since every country is organized differently, the rules presented for Germany in this section will likely have to be adapted for geo coding pages in other countries and languages. In the United States, for example, most addresses contain the state, which can be used to resolve ambiguity between towns of the same name. In German addresses, states are never mentioned. German telephone area codes and zip codes are highly clustered, i.e., codes with a common prefix tend to be in the same region. Large companies might have their own zip code, but we could infer their position from the positions of similar zip codes.

We give a brief summary of Germany's administrative geography. States play little role in daily life and are not mentioned in addresses; we therefore ignore them. Counties and districts are treated similarly, since many Germans may not even know in which county or district they live. Area codes and zip codes are distributed in clusters. At least all entries with the same first digit are clustered. There is no simple relation between these numeric codes and towns. A town might cover several of these codes or several towns might share the same numeric code.

Towns fall into two categories, cities and villages or boroughs. There is a one-to-many relationship between the two. Every village or borough is associated with exactly one city, and every city might be associated with several villages. Villages are often mentioned in conjunction with their cities. German town names consist of up to three parts:

1. an optional *descriptive prefix*, such as *Bad*, consisting of a single term
2. a mandatory *main name*, such as *Frankfurt* or *Göttingen*, usually consisting of a single term.
3. Any number of *descriptive terms*, such as *bei Weimar*, *am Main*, *Thüringen*[1]

Descriptive prefixes and large parts of the descriptive terms are often dropped or abbreviated in various ways. The city of *Frankfurt am Main* might be abbreviated as *Frankfurt M.*, *Frankfurt/Main*, *Frankfurt a.M.*, or just *Frankfurt*.

4.4 Geo Extraction

This process reduces a document to the subset of its terms that actually contain geographic meaning. If there is any uncertainty whether a term is used in a geographic meaning or not (called *geo-nongeo* ambiguity by [1]), then this is resolved during this step. We extract only those geographic markers from pages that we know how to map to geographic positions, in our case town names, phone numbers, and zip codes. We first focus the discussion on extraction from page content, and later show how to adapt this to extract markers from URLs.

[1] near the city of Weimar, on the river Main, in the state of Thuringia

4.4.1 Town Names

When parsing pages to extract terms that might refer to towns, we could simply write out all terms that appear in some position of some town name. However, if we do not resolve geo-nongeo ambiguity in term usage at this point, then we would produce a lot of garbage, since many terms that appear in town names are also common German or English words and many town names are also common surnames. Filtering these cases out at a later point is much more tricky. To resolve geo-nongeo ambiguity, we therefore manually divided the set of all terms that appear in any town name into:

- 3,000 *weak terms* that are common language terms.
- 55,000 *strong terms* that are almost uniquely used as town names, except when used as a last name.

When parsing web pages, we first try to extract all strong terms. Next, we look for all weak terms with which any of the extracted strong terms appear in the same town name. The underlying idea is that we try to find a town's main names first and then parse for weak descriptive terms to resolve any ambiguity.

We also assigned a distance d to each weak term. A weak term would only be recognized if it appears within less than d positions from an associated strong term. Thus, if we have found the strong term *Frankfurt*, we would accept the weak term *Main* anywhere on the page ($d = \infty$), or the weak term *Oder* within a distance $d = 2$ since it is a much more common term.[2]

To further increase the precision of the extraction, we introduced *killer terms* and *validator terms*. These are defined by mappings from strong terms to other terms. If a strong term is mapped to a killer term with a distance d, then any appearance of the strong term will be ignored if the killer term appears within this distance. If a strong term is mapped to a validator term with a distance d, then any appearance of the strong term will be ignored unless at least one validator term appears within this distance. These mappings allowed us to deal with towns where the main term is also commonly used in everyday language. We additionally introduced a list of *general killers* such that any strong term within some distance of a general killer will be ignored. This list was filled with about 3,500 common first names and titles such as *Herr*, *Frau*, or *Professor*, thus trying to avoid mistaking surnames for town names.

We filled all these tables manually, which is a one-time effort since they can be reused for future crawls. Of course, other more sophisticated techniques, e.g., from natural language processing, could also be applied in this context. The simplicity of the architecture allows the database to be easily extended if additional types of geographic data are to be extracted. Derived geographic features, such as the fact that "Oktoberfest" typically correlates with locations in Munich, could be determined using data mining techniques and then added to the database to improve geocoding. Computational costs during geo extraction are limited to a few table lookups and thus not a major concern.

4.4.2 Numeric Indicators

We also extracted telephone area and zip codes. We checked every number we encountered against a list of all telephone area and zip codes. Every detected zip code also had to qualify through the presence of a validator term, in our basic implementation any strong term that appears in the name of a town that shares the zip

[2] *Main* indicates the river Main, while *Oder* indicates the river Oder; however, *Oder* is also the German word for *or* and thus extremely common. Its appearance on a random position of a page most definitely has no geographic meaning.

code, since zip codes are rarely used without an accompanying town name. Formatting is more useful to distinguish phone numbers from other numbers, as there are certain typical ways in which phone numbers are usually represented.

4.4.3 Parsing URLs

The challenge in extracting geographic terms from URLs is that a URL is a single string where words are often not clearly separated, as opposed to words in pages that are usually separated by blanks or punctuation marks. Of course, we can start by breaking the full URL into several sections for TLD, domain, sub domains, host, path, and file, in order to reduce the problem. But the original problem still exists in each of these parts, particularly the domain, where constructs such as `bostonyoga.com` or `beachesofsouthwalton.com` are quite common. We address this problem by introducing the concepts of *soft* and *hard delimiters* that divide a string into substrings that are likely to be terms. For every part of the URL, we compare all substrings against the set of strong terms from our list of cities, and for any match check if it *qualifies*. If so, it is written out and we search for corresponding weak terms. In order to qualify, we require a term to be bounded by at least one hard and one soft delimiter. Hard delimiters are all characters except upper and lower case letters. Soft delimiters are all common German and English words, which we took from the dictionary of OpenOffice [27]. This identifies most cases. Consider the following examples:

- `www.fitness-frankfurt.de` contains *frankfurt* with two hard delimiters
- `www.fitnessfrankfurt.de` contains a *frankfurt* with one hard and one soft delimiter (the word *fitness*)

Our approach ignores most false cases, where some random substring happens to resemble a town name, for example

- `www.registrierkasse.de`[3] where the town name *Trier* has only a single soft delimiter, *Kasse*.

4.5 Geo Matching

The previous step reduced documents to sets of terms that carry a geographic meaning. This step maps these terms to actual towns and thus to geographic locations. The problem we encounter is that some terms can point to several town names, a situation called *geo-geo ambiguity* in [1]. Not only do some towns share the same main name, a town's main name might also appear in another town's descriptive terms. We make two assumptions about the usage of town names that allow us to define a strategy for resolving these ambiguous cases.

The first assumption is that the author of a document mentioning a town name intends to talk about *a single town of this name*, not about several towns of that name. That is, someone mentioning *Frankfurt* intends to talk about either one of the two towns in Germany of that name. This assumption is called *single source of discourse* in [1]. Even if this assumption fails, it only introduces a negligible error to a geographic search engine. Thus, in the rare case where a document discusses why neither town named Frankfurt currently has a strong soccer team, it might be acceptable to only assign this page to one of the two towns, say Frankfurt am Main.

The second assumption is that the author most likely meant the largest town with that name. There are for examples two towns of the name *Göttingen*, a larger city and a tiny village, situated about 150 km apart. Intuitively the probability for the larger town to be meant is much higher than for the smaller one. The page will be therefore be assigned to the city of *Göttingen*, not the village, unless there are other strong indications. Just as above, it can be argued that the failure of this hypotheses only introduces a marginal error, especially when the difference in town size is huge. If the document intended to refer the city, the matching is correct and a geographic search engines will return correct results in both towns. If it intended to refer to the village, then the matching is incorrect but only introduces a marginal error.

Our strategy consists of the following steps. First, a metric is used to evaluate matches between town names and terms. Second, we write out the town with the best match, and then delete its terms from the term set. Finally, we start over to find additional matches on the reduced term set. There are several measures for the quality of a match between a town name and a set of terms. The actual implementation details of the algorithm are omitted, since it is tailored to Germany's administrative geography and to the databases available to us. The general strategy however is broad enough to be cast into different algorithms for various countries and data sets.

4.5.1 Measuring Geo Matching

The degree to which a town name can be matched with a set of retrieved terms can be measured in several different ways. None of these perform well on their own, but in combination they provide for a good estimate of how well a town can be matched. For our application, we do not need to compute an absolute value for the degree of matching, but really only an ordering that allows us to pick the better of several towns that can be matched from the same set of terms.

One simple measure would be the *number of matched terms*, i.e., the number of terms in the town name that are contained in the set of terms from the web page. A similar measure is the *fraction of matched terms*, i.e., the fraction of terms in the town name that were found in the page.

For any of the above, one can find examples where they work really well and ones where they fail. Some other types of techniques are stronger. It makes things a lot easier when a *numeric marker* such as a zip code is found. When applicable, this by itself should resolve most ambiguous situations. Another more reliable technique takes place on a geographic level, by looking for *nearby towns*. If we find both `Frankfurt` and `Offenbach`, then we can be pretty certain that the page intends to talk about *Frankfurt am Main*.[4] In our application we employed a simplified version of the last approach that looks for terms that refer to *nearby villages* that are associated with the same town. The big advantage is this measure can be looked up from a table without ever having to compute an actual distance.

4.5.2 The Matching Strategy

Since the implementation of our matching algorithm, called *BB-First*, is very specific to Germany we will not show it in full detail, but rather sketch the underlying strategy. The algorithm is called *BB-first* because it extracts the *best* of the *big* towns *first*. It starts with the set of all strong terms found in the document, called `found-strong`, and the set of all German towns, and proceeds as sketched in Table 4.1.

Since we measure the size of towns only by sorting them into *villages* and *cities*, we ran the algorithm only with these two categories. The algorithm can be directly traced to the underlying assumptions. It clearly prefers large towns over small ones. It also

[3] *Registrierkasse* is German for *cash register*

[4] The city of Offenbach is a direct neighbor of Frankfurt am Main, and about 700km from the other Frankfurt, Frankfurt an der Oder.

1. Group towns into several categories according to their size.
2. Start with the category of the largest towns.
3. Determine the subset of all towns from this category that contain at least one term in found-strong.
4. Rank them according to a mix of the measures described in Section 4.5.1.
5. Add the best matched town to the result.
6. Remove all terms found in this town name from the set found-strong.
7. Start this algorithm over at Step 3, as long as there are new results.
8. If there are no new results, repeat the algorithm for the next category down.

Table 4.1: Basic steps of the BB-First algorithm

assumes a single sense of discourse, since every strong term can cause at most one town to be matched, before it is removed from found-strong. The extracted towns receive a *certainty value*, estimated with the same measures we used to determine how well towns were matched with the set of terms.

The results of this algorithm, i.e., the matched towns, are then finally mapped into our quad-tree based footprint structure with integer amplitudes. Note that cities are not mapped to a single tile but to a larger area of a few kilometers squared. Each tile in the grid receives as amplitude the sum over the certainties of towns that map to this tile. Applying this procedure to every document results in an initial geo coding of our web crawl, that can be processed further during the next step. In this initial coding, each page that contained a geographic marker has associated with it a non-empty footprint structure that models its geographic footprint. In our set of 31 million pages, about 17 million had non-empty footprints based on page content, represented in an average of 137 bytes after compression. About 5.7 million pages had (separate) non-empty footprints based on extraction of markers from their URLs, represented in an average of 38 bytes since there are fewer extracted markers on average.

4.6 Geo Propagation

After applying the above techniques, and excluding whois entries, slightly more than half of all web documents have a non-empty geographic footprint (in the form of a quad-tree) associated with them. This is close to the best one can expect from geo extraction, since not every document contains a geographic reference. One problem we noticed was that many of the pages that did have a footprint were not particularly valuable in terms of their actual content. For example, it seems that many sites return geographic information such as contact addresses in separate pages from the actual content that a user might be looking for. This situation can be overcome by *geo propagation*, a technique that extends the basic radius-one, radius-two (co-citation), and intra-site hypotheses from Web information retrieval to the geographic realm.

According to the radius-one hypothesis, two web pages that are linked are more likely to be similar than any random pair of web pages [11]. This assumption can be directly extended to geographic footprints. If one page has a geographic footprint, then a page it is linked to is more likely to be relevant to the same or a similar region than a random page. The radius-two hypothesis about pages that are co-cited can an be extended similarly.

The intra-site hypothesis, often overlooked or used only implicitly in Web IR, is highly useful in geo coding. It states that two pages in the same site are more likely to be similar than any given random pair of pages. For documents from the same sub-domain, host, or directory within a site, even stronger versions of this hypothesis can be stated. This assumptions can clearly be extended to geographic properties. For Germany, it is particularly useful since there exists a law that any de site must have a page with the full contact address of the owner no more than two clicks from the start page! Therefore, at least one page in any given site at least in principle should provide rich geographic information which is supposed to apply to the entire site.

Geo propagation uses the above geographic hypotheses to propagate geographic footprints from one page to another. The idea is that if two pages are related in any of the above manners, they should inherit some dampened version of each others geographic footprint. We modeled the "inheritance" by simply adding the entire footprint of one page to the other, tile by tile, with some dampening factor $0 < \alpha < 1$. The exact value of α depends on the relationship between two pages. If two pages are in the same directory for example, α will be larger than if they are only within the same site.

Note that this process is not intended to converge, and has to be handled with care. If geo propagation is performed too often, then every single document could end up with a footprint that basically covers the entire country with non-zero amplitudes. In practice, propagation over one or two steps seems to give most of the benefit, and proper dampening factors plus lossy compression (simplification) makes sure that footprints do not get out of hand. The result of the geo propagation step is increased coverage – in terms of percentage of pages modeled - and better quality of the geographic footprints.

4.6.1 Geo Propagation in our Prototype

Based on these general ideas, we implemented several forms of geo propagation. Starting out with about 17 million footprints, we separately performed forward and backward propagation across links as well as between co-cited pages. Thus, if page A has a footprint m_A and links to a page B with a footprint m_B, then we transmit m_A to B and compute a new footprint of the form $m_B + \alpha m_A$ for B. This is implemented using two ingredients: (1) our optimized implementation of footprint operations based on quad-tree structures described in Subsection 4.1, and (2) an I/O-efficient implementation for footprint propagation along links that resembles a single round of an I/O-efficient Pagerank algorithm described in [6]. In particular, footprints are sorted on disk by destination page and then aggregated into the footprint of the destination page.

Propagation was also performed within sites; in this case I/O efficiency is not a concern since each site can be processed independently and thus entirely in main memory. Some normalization by the size of the site is necessary to avoid very large amplitudes. In the end, we obtained about 28.4 million pages with non-empty footprints, for a page coverage of more than 90%. We also separately stored 490,000 footprints that apply to entire sites. This amounts to about 60% of all sites, which is smaller than expected, due to the large number of parked single-page sites. The site's footprints can be used separately during query processing or added back into pages' footprints in advance.

5. Geographic Search

Geographic search engines allow users to focus a search on a specific geographic area by adding a query footprint to the set of keywords. There are a number of possible interface for specifying the query footprint and displaying the results, and we sketch

Figure 5.1: A simple interface for a geographic search engine.

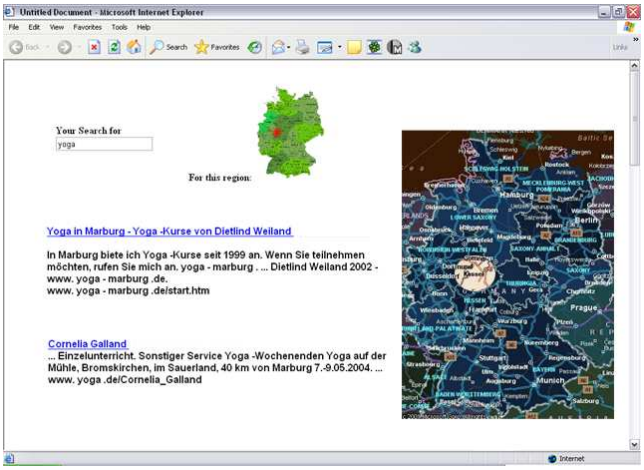

Figure 5.2: The result set with visualization of the geographic footprints.

here only one fairly basic solution. In particular, the area of interest could be specified as a keyword (such as a city name) that is converted by the interface into a suitable query footprint, or a user could use an interactive map for this purpose. In a mobile environment, the current location of the user could be determined from the networking infrastructure and then translated into a footprint. Results could be shown as lists or displayed on an interactive map, and additional geographic browsing operations may be supported. Note that a query footprint should not be seen as a simple filter for keyword-based results, but as a part of the ranking function. We will describe the actual query processing in two passes, first on an abstract level and later in terms of our actual current implementation. We start by briefly describing a simple interface to give the reader a better understanding of the scenario.

5.1 Interface

A simple user interface as shown in Figure 5.1 consists of a text box, a map and several radio buttons. Users may zoom into more detailed maps. The map allows the user to choose the center of the region of interest. The radio buttons allow us to specify the range for the search distance from the desired location. A query footprint is then computed from the center of the map and the distance buttons, and the search engine looks for relevant results whose footprints intersect the query footprint. The returned results, shown in Figure 5.2, contain the title of each page, a text snippet from that page, and the URL. For each result, we also provide a visualization of the geographic footprint. This allows users to evaluate if the results are of the desired type and gives hints for refining the query (and happen to be highly useful during debugging). We note that other alternatives are to extract a query footprint from the query [20], or to assume the current location of the user or a default location chosen by the user.

5.2 A Simple Geographic Search

Without going into too much detail, the differences between geographic search engines and their traditional counterparts can best be outlined on a very abstract level. In a nutshell, a traditional search engine works as follows:

1. The user inputs a set of search terms.
2. The engine uses the inverted index to determine the set of pages that contain all the search terms. This is done by intersecting the sets of document IDs in the inverted lists of the search terms.
3. The engine then uses the numbers, contexts, and positions of the term occurrences in these pages, together with other measures such as link structure, to rank the results. This is typically done concurrently with the second step.

At first glance, the query processing in our geographic search engine works in a very similar way:

1. The user inputs a set of search terms *and* a query region that is converted into a *query footprint*.
2. The engine then uses the keywords *and* the query footprint to compute a result set from the inverted index.
3. The engine then uses the keywords *and* query footprint, plus other measures such as link structure, to rank the results.

Thus, the engine uses both keywords and the query footprint to retrieve candidates results in the second step as well as to rank them in the final step. In our case, the first step is simply a question of interface design. The second step is also fairly similar to traditional search engines, with the difference that now only those pages survive that contain all search terms and have a non-empty intersection. The final ranking however is a little more complicated since it has to merge two unrelated ranking measures, importance and geographic proximity.

5.3 Geographic Ranking

We now describe in detail how we rank pages based on both terms and geographic footprints. The user of a geographic search engine wants top results to fulfill two criteria: they need to be relevant as well as close to the query footprint. A straightforward approach would be to simply use the query footprint as a filter that removes all results that are “outside” the query area, and to then use the standard search engine ranking on those results. At the other end of the spectrum, we could use the search terms as a filter, and rank all documents in the intersection of the inverted list by their distance to the center of the query area.

We decided on a general framework that includes these two cases as well as the spectrum in between, allowing users to select their

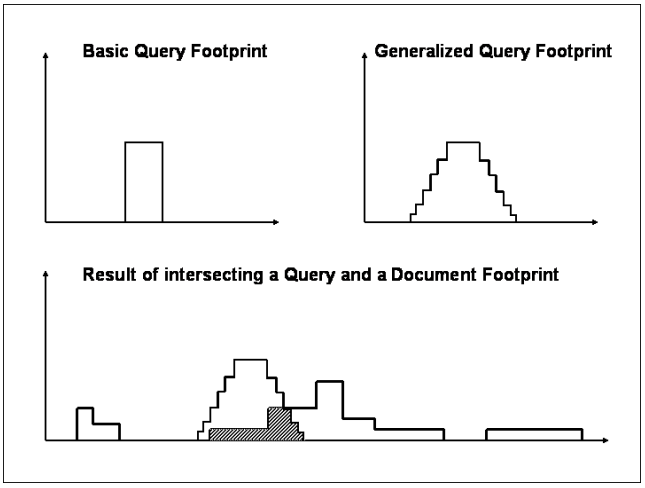

Figure 5.3: A simple illustration of footprints in a single spatial dimension. At the top, we have a query footprint with a distance threshold (left), and a footprint for a query that gives a lower score for documents that are farther away (right). At the bottom, we show an intersection computation between a query footprint and a document footprint. In reality, the geographic score would be determined by first applying a log-based function (somewhat reminiscent of those used in cosine measures) to the amplitudes of the footprints, and then computing the volume of the intersection.

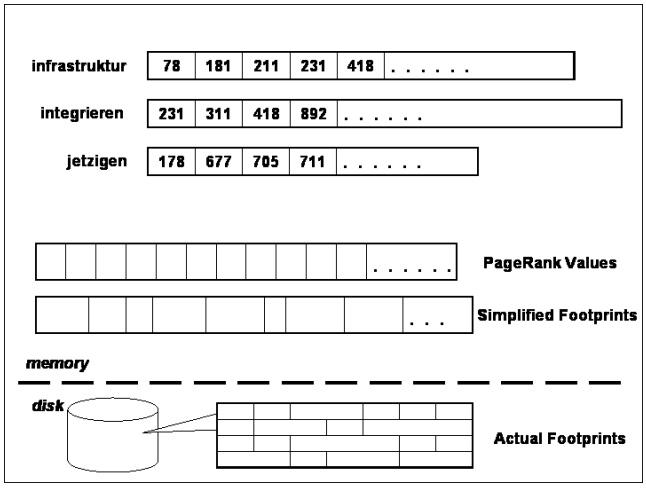

Figure 5.4: Organization of index structures, lookup tables, and geographic footprints in a scalable geographic engine.

own preferences. First, we allow the user to choose different shapes for the query footprint as shown at the top of Figure 5.3. If a user prefers a sharp cutoff at a distance of say 10 km, then the user selects the footprint on the left, while the query footprint on the right models a more gradual approach. During the ranking phase, we then compute a *geographic score* for each page in the intersection of the inverted lists of the query terms, based on the size (volume) of the intersection between query and document footprint; see the bottom of Figure 5.3. If the intersection is empty, the document is discarded.

Second, the user can choose the relative weight of the term-based and geographic components of the ranking function. Thus, the total score of a document under the ranking function will be a weighted sum of its term-based score, its geographic score, and maybe an additional measure such as Pagerank. We note that in reality, both the intersection computation and the summing up of the various scores requires careful normalizations to avoid adding up apples and oranges. Both the shape of the query and the relative weighting of the scores are provided by the user through a simple system of sliders that model the trade-offs, and that can be interactively adjusted to reorder the current query results.

5.4 Efficient Geographic Query Processing

Given this approach to ranking, we can now describe efficient query processing in more detail. Figure 5.4 shows the example of a query with three search terms. After the query is issued, the inverted lists for the three terms are loaded into main memory (shown here only with their document IDs), and their intersection is computed by traversing the lists from left to right. For any document in the intersection, two lookups are performed. First, we have an in-memory table of highly compressed document footprints, obtained by lossy-compressing the footprint structures down to a size of about 100 to 200 bytes each. We perform a lookup into this table to check if the intersection between the query footprint and the document footprint is nonempty, and if so, we compute an approximation of the geographic score of the document. We then perform a lookup into an in-memory table of Pagerank values to compute a final approximate document score.

After we have traversed the inverted lists and determined say the top-50 results, we can then perform a more precise computation of their geographic scores by fetching their precise document footprints, which might be stored separately on disk or together with the actual pages which need to be fetched anyway in order to extract text snippets for display on the result page. There are a number of other performance optimizations in search engines, such as index compression, caching, and pruning techniques [22], that we have omitted here, but that also apply to this case. By integrating these, we can achieve query throughput comparable to that of a standard non-geographic engine.

We note that if we compress the in-memory footprints down to an average of 100 to 200 bytes, then several million pages could be covered by each node of the search engine cluster, a realistic number for large engines. In our prototype, we use a cluster of 7 Intel-based machines with reasonably large disks and main memories for our 31 million pages, which can sustain rates of a few queries per second.

6. Conclusion

In this paper we described the design and implementation of a crawl-based geographic search engine prototype for the German web domain. We introduced the notion of a document's *geographic footprint*, as the set of locations it might provide information for, and showed how to construct footprints for crawled pages through a *geo coding* process consisting of *geo extraction*, *geo matching*, and *geo propagation* steps. We discussed how this process depends on a country's administrative geography and on conventions for referring to geographic locations. We represent geographic footprints through a compressed quad-tree structure that is manipulated via a few simple operations during geo propagation and query processing. Finally, we described techniques for efficient query processing in a geographic engine, and discussed a basic search interface.

Our prototype engine is currently close to completion, and we plan to make it available soon. One major open issue is the need to perform an appropriate evaluation of the quality of our footprints

and query results, which we plan to do in the near future. Beyond the current prototype, there are many exciting open problems for future research in this area. On the most general level, many aspects of Web search and information retrieval, such as ranking functions, categorization, link analysis, crawling strategies, query processing, and search interfaces, need to be reevaluated and adapted for the purpose of geographic search.

We are particularly interested in the following directions. First, we are working on extracting *derived geographic features*, by which we mean terms such as "Oktoberfest" or "statue of liberty" that are not listed in standard geographic databases but clearly associated with a particular location. This can be done through the use of data mining operations that relate such features to known geographic entities. Second, we are interested in query processing optimizations for geographic search engines. In our approach in this paper, we first use the inverted index to narrow down the set of documents and then look at the geographic footprints. We expect that additional optimizations are possible through the integration of appropriate spatial index structures and through spatial partitioning of the inverted index structures. Other possible improvements could adapt and integrate pruning techniques for top-k queries such as [17, 22]. Third, we plan to study focused crawling strategies [5] that can efficiently fetch pages relevant to a given geographic area (e.g., the United States) that runs across many top-level domains.

Finally, we are very interested in *geographic data mining* of the Web. A simple example of such mining are the derived geographic features mentioned above. There are many other possible applications in connection with market research, competitive intelligence, and national security. An example of commercial geographic web mining is *metacarta* [25]; other natural candidates for integrating geographic techniques would be products such as IBM's *Web Fountain* system [21].

7. Acknowledgments

We thank Thomas Brinkhoff for cooperation on earlier related work [23] leading up to this project, and for continuing feedback and support.

8. REFERENCES

[1] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: geotagging web content. In *Proceedings of the 27th SIGIR*, pages 273–280, 2004.

[2] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about web pages via random walks. In *Proc. of 26th Int. Conf. on Very Large Data Bases*, September 2000.

[3] DCMI Usage Board. Dublin Core Qualifiers. Recommendation of the DCMI, Dublin Core Metadata Initiative, Jul 2000.

[4] O. Buyukkokten, J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar. Exploiting Geographical Location Information of Web Pages. In *WebDB*, pages 91–96, 1999.

[5] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proc. of the 8th Int. World Wide Web Conference*, May 1999.

[6] Y. Chen, Q. Gan, and T. Suel. I/O-efficient techniques for computing pagerank. In *Proc. of the 11th International Conf. on Information and Knowledge Management*, pages 549–557, November 2002.

[7] Open GIS Consortium. `www.opengis.org`.

[8] A. Daviel. `www.geotags.com`, Apr 1999.

[9] A. Daviel. Geographic registration of HTML documents, Apr 2001. Internet Draft `http://geotags.com/geo/draft-daviel-html-geo-tag-05.html`.

[10] C. Davis, P. Vixie, T. Goodwin, and I. Dickinson. A means for expressing location information in the domain name system. RFC 1876, Internet Engineering Task Force, Jan 1996.

[11] B. Davison. Topical locality in the web. In *Proc. of the 23rd Annual Int. Conf. on Research and Development in Information Retrieval*, July 2000.

[12] Digital Envoy inc. `www.digitalenvoy.net`.

[13] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of web resources. In *Proc. of the 26th VLDB*, pages 545–556, Sep 2000.

[14] Divine inc. Northern light geosearch. `www.northernlight.com/geosearch.html` last accessed Feb 2003.

[15] M. Egenhofer. Toward the semantic geospatial web. In *Proc. of the 10th ACM GIS*, pages 1–4, Nov 2002.

[16] Eventax GmbH. `www.umkreisfinder.de`.

[17] R. Fagin. Combining fuzzy information: an overview. *SIGMOD Record*, 31(2):109–118, June 2002.

[18] Google inc. Google Local, 2003. `http://labs.google.com/location`.

[19] L. Gravano. Geosearch: A geographically-aware search engine, 2003. `http://geosearch.cs.columbia.edu`.

[20] L. Gravano, V. Hatzivassiloglou, and R. Lichtenstein. Categorizing web queries according to geographical locality. In *Proc. of the 12th CIKM*, pages 325–333, Nov 2003.

[21] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a webfountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 43(1):64–77, 2004.

[22] X. Long and T. Suel. Optimized query execution in large search engines with global page ordering. In *Proc. of the 29th Int. Conf. on Very Large Data Bases*, September 2003.

[23] A. Markowetz, T. Brinkhoff, and B. Seeger. Exploiting the internet as a geospatial database. In *Workshop on the Next Generation Geospatial Information - 2003*, Oct 2003. Similarly presented at the *3rd Int. Workshop on Web Dynamics at WWW13*, 2004.

[24] K. McCurley. Geospatial mapping and navigation of the web. In *Proc. of the WWW10*, pages 221–229, May 2001.

[25] Metacarta, 2002. `www.metacarta.com`.

[26] Open Directory Project. `www.dmoz.org`.

[27] OpenOffice.org. International spelling dictionaries. `http://lingucomponent.openoffice.org/spell_dic.html`.

[28] V. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *Proc. of the 2001 SIGCOMM*, pages 173–185, Aug 2001.

[29] Räber Information Management GmbH. `www.search.ch`.

[30] V. Shkapenyuk and T. Suel. Design and implementation of a high-performance distributed web crawler. In *Proc. of the Int. Conf. on Data Engineering*, February 2002.

[31] Yahoo! inc. `www.yahoo.com`.

[32] Yahoo! inc. Yahoo LOCAL, 2004. `http://local.yahoo.com`.

[33] M. Zook. Old Hierarchies or New Networks of Centrality? The Global Geography of the Internet Content Market. *American Behavioral Scientist*, 44(10):1679–1696, June 2001.