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. 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 assigns 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 architecture and 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, 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 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 local businesses, locally relevant news items, or tourism information about a particular region. When taking up yoga, local yoga schools are often of much higher interest 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, geographic search engines provide a very useful tool. They allow a user to express in a single query what might take multiple queries with a standard search engines. Thus, a user of a standard search engine looking for a yoga school in or close to Brooklyn may end up trying queries such as `(yoga AND new york)` or `(yoga AND brooklyn)`, but even this might yield inferior results as there are many ways to refer to a particular area 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* on mobile devices. Third, geographic search supports locally targeted web advertising, thus attracting advertisement budgets of small businesses. Other opportunities arise from mining geographic properties of the web, e.g., for market research.

Given these opportunities, it comes as no surprise that most leading search engine companies have made significant efforts to deploy some geographic web search. Our approach differs, both in the way geographic information is extracted from the web and in the way it is integrated into query processing. In particular, commercial engines focus on matching pages with data from business directories, supporting search for local businesses and organizations. While this is an important part of geographic search, we focus on 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 non-commercial web sites, such as local news and cultural events, or the history of a certain area. In order to facilitate such queries, we extract geographic markers, such as addresses or phone numbers, from web pages, independent of their listing in any 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 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, focusing on Germany and crawling the de domain. 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 test and further develop ideas outlined in earlier work [21, 11, 19], by building a complete prototype.

Our contributions are: First, we provide the first in-depth description of an actual implementation of large-scale geographic web search. Our prototype, to be made available soon, is based on a crawl of over 30 million pages in the de domain, with plans to expand further. Second, we combine several known and new techniques for deriving geographic data from the web, using features such as town names, zip codes, phone numbers, link and site structure, and external sources such as whois. We represent the resulting geographic footprint of a page in a simple highly compressible format that is used during link and site analysis and query processing. Third, we provide the first discussion of efficient query execution in large geographic search engines. Due to space constraints, we have to omit many details. An expanded version of this paper is available as a technical report [20].

2. RELATED WORK

In this section, we describe related work on *geo coding*, existing geographic search engines, and the Semantic Web. Since we treat content, not hardware, we have omitted work on determining physical

[1]Department of Computer Science, University of Science and Technology, Kowloon, Hong Kong. alexmar@cs.ust.hk
[2]CIS Department, Polytechnic University, Brooklyn, NY. {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

Eighth International Workshop on the Web and Databases (WebDB 2005), June 16–17, 2005, Baltimore, Maryland.

locations of servers. Most web pages today are hosted in server farms hundreds of miles away from either author or geographic regions they relate to.

Geo Coding: A good discussion of geographic hints commonly found in web pages is provided by McCurley [21], who introduces the notion of *geo coding*. He describes various geographic indicators found in pages, such as zip codes or town names, but does not discuss in detail how to extract these, or how to resolve the geo/geo or nongeo/geo ambiguity.

Work in [4, 11] introduced the idea of a page's *geographic scope*, i.e., the area it addresses in terms of readership. Initially, they assign a position to every web site based on its `whois` entry. Then a fixed hierarchy of administrative regions is superimposed, and some link analysis is performed. If enough links from a region point to a web site and these links are evenly distributed, the region is included in the site's geographic scope. The approach was applied to the United States using states, counties, and cities for the geographic hierarchy. Our approach in Section 4.6 is basically a generalization and refinement of the work in [4, 11], but differs in several ways. In general, [4, 11] is fairly coarse-grained as it focuses on sites instead of single pages and on relatively large geographic regions. It relies on the existence of a sufficient number of incoming links, and thus does not work well for pages and sites with moderate in-degree. The evaluation in [11] is limited to sites in the `edu` domain, where `whois` provides a good estimate of a site's location, and does not address more noisy data from page content.

The approach closest to ours is [1], using a hierarchical gazetteer with 40,000 names of cities all over the globe. It performs geo coding by looking for names of cities with more than 500,000 people, though decreasing the minimum size to 5,000 is reported to have a positive effect. The gazetteer's hierarchy is used for disambiguation when there are several towns of the same name, but the size of towns is not considered in this case. Similar to our geographic footprint, [1] focuses on a document's *geographic focus* rather than the more specialized *geographic scope* of [11]. In contrast to our system, the geographic focus of a page is not represented 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 this by grouping.

Geographic Search Engines: Several geographic search engines are already available online. Some are academic prototypes based either on small specialized collections or a meta search approach. In particular, [16] performs automatic geo coding of documents based on the approach in [11]. Most other 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 engines by commercial players. Some, such as the extension to *Northern Light* by *Divine* [12], have already disappeared again. Others such as [14] rely on geographic meta data in the pages, or query internet directories such as the Open Directory Project. Of all these, the Swiss *search.ch* engine [22], which has been around for several years, is closest to our approach. It 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, geographic search has recently received a lot of attention from the major search companies. Both *Google* and *Yahoo* have introduced localized search options [15, 24]. Their approaches appear to be quite different from ours, and seem to rely on an intermediate business directory. Users first retrieve entries for businesses that satisfy certain keywords and are close by, and can then extend the search to actually retrieve pages about these businesses and the area they are located in. The exact algorithms are not publicized. There are two main differences to our approach, the intermittent business directories and the location modeling to point coordinates, i.e., the street address of the businesses, to be displayed on detailed street maps. There seems to be no mechanism to model geographic footprints of pages that cover larger areas, such as a county or state.

Geographic Semantic Web: It seems natural to extend the Semantic Web to a *Geographic Semantic Web*, such as proposed in [13], where each web page contains some meta data, defining its geographic footprint. Several models are already available [3, 9]. Other models from the GIS community, such as GML from the Open GIS Consortium [7], can be adapted. However, there are two major problems, inherent to the Semantic Web, that make this approach infeasible for general web search (though it may be useful in other scenarios). First, there is a *chicken and egg* problem. Authors will only include meta information if search engines use them, while engines will wait for a sufficient amount of meta information to become available before building any services on it. Second, Web authors are not to be trusted, as they frequently provide misleading information to manipulate search engines. For this reason current engines pay little attention to existing meta tags.

3. UNDERLYING DATA

We now briefly describe the data, as used in our prototype. Using the *PolyBot* web crawler [23], we acquired about 31 million distinct web pages from the `de` domain in April 2004. 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 registered domains within a relatively small area. It is also reasonably self-contained in terms of link structure. Thus, the domain provides a nice test bed, meaningful but not outside the reach of academic research. The `de` domain was estimated in 2000 at 3.67% of the entire web [2]. This translates to about 150 million pages to achieve the same coverage as 4 billion pages (the size of *Google* as of November 2004) on the entire web; this is within reach of our current setup. Second, availability of geographic data is a big issue. The `whois` entries for `de` domains are complete and well structured, allowing us to extract information without 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 an understanding of the language, administrative geography, and conventions for referring to geographic entities.

We focused on two geographic data sets for Germany. The first maps each of 5,000 telephone area codes to *one* city and also to the coordinates of the centroid of the region that the code covers. The second maps zip codes to 82,000 towns, and these towns to their positions. 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 for display to the user. Names were often misspelled or abbreviated in various nonstandard ways, requiring painstaking manual cleaning.

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 multiple locations, for example when a company web page refers to several different outlets. We call this collection of locations the page's *geographic footprint*. For every location in the footprint, an integer value is assigned that expresses the *certainty* with which we believe the web page actually provides information relevant to the location.

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 an initial geo coding of the pages. In these first two steps, we make use of databases of known geographic entities such as cities or zip codes. 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. Before we proceed with the description of our geo coding process, we introduce our representation of a document's geographic footprints.

4.1 Geographic Footprints of Web Pages

In every GIS, a basic design decision has to be made between a vector data model and a raster data model, mapping data onto a discrete grid. A web page may contain several geographic hints, some referring to point positions, others (cities or zip codes) refer to polygonal areas. Thus, our data model has to handle both types. We decided to use a raster data model, representing geographic footprints in a bitmap-like data structure. In comparison to a vector model, we lose some precision by pressing the information into the grid. With a sufficiently fine grid however, the degree of imprecision is small, especially when compared to other uncertainties in the data and 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 amplitude with each tile, expressing the certainty that the document is relevant to the tile.

This representation has two advantages. First, it allows us to efficiently implement some basic aggregation operations on footprints. If a page contains several geographic features, the footprint for the page is defined by sum over individual features; after a possible normalization. These operations are very useful during geo propagation and query processing. Second, since for most documents only few tiles 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; important for efficient query processing.

We implemented a small and highly optimized library for operations such as footprint creation, aggregation, simplification (smoothing), and intersection (for query processing) based on quad-trees. 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 geographic markers extracted from pages, various external sources can also be used for geo coding, in particular business, web, and `whois` directories.

Business directories (yellow pages) map businesses and associated web sites to addresses, and thus to geographic positions. Some geographic search engines such as those of *Google* and *Yahoo* [15, 24] appear to make heavy use of business directories. The main problem with business directories is also their biggest strength. They require registration fees, and thus usually list mainly commercial companies, ignoring many personal or non-profit web sites. The fees however also often result in higher data quality.

Web directories such as Yahoo and ODP maintain geographic directories that categorize sites by region. They are difficult to maintain, far from complete, and often outdated. However, they can be useful as an additional data source in geo coding.

As an integral part of the Internet infrastructure and freely accessible, the `whois` directory is also a good source of geographic information. For every domain, it contains the address of the individual that registered it. An earlier study [25] showed a high degree of accuracy for `whois` entries. However, the quality of `whois` entries differs between top-level domains. For the `de` domain, they are highly structured and usually complete, with precise addresses and phone numbers. In contrast, entries for the `uk` domain typically contain less information and are fairly unstructured.

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

4.3 Germany's Administrative Geography

Effective geo coding requires an understanding of a country's *administrative geography* and common usage of geographic terms. Thus, one has to know how geographic names are composed, what the role of states and counties is, and how postal or area codes are used. Since every country is organized differently, the rules presented for Germany in this section will have to be adapted for other countries and languages. In the United States, for example, most addresses contain the state, which can be used to resolve ambiguities between towns with 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 the usage of geographic terms in Germany. States, like counties and districts play little role in daily life, are rarely mentioned and thus ignored. Area 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 (also boroughs), with a one-to-many relationship between the two. Every village is associated with exactly one city, but a 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. First there is an optional single-term *descriptive prefix*, such as *Bad*. Second there is a mandatory *main name*, such as *Frankfurt*, while third, extra *descriptive terms*, such as *bei Köln*, *am Main*, *Sachsen*.[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 written as *Frankfurt M.*, *Frankfurt/Main*, *Frankfurt a.M.*, or just *Frankfurt*.

4.4 Geo Extraction

This step reduces a document to the subset of its terms that have geographic meaning. If there is any uncertainty whether a term is used in a geographic meaning or not (called *geo-nongeo* ambiguity [1]), then this is resolved at this point. We extract only those geographic markers that we know how to map to geographic positions: town names, phone numbers, and zip codes. In addition to page content, we also analyzed URLs. URLs are a very useful source of geographic information, but tricky to analyze since terms are often not well separated (e.g., finding a city name in `cheapnewyorkhotels.com` is not straightforward). We refer to [20] for details.

4.4.1 Town Names

When extracting terms that might refer to towns, we could simply write out all terms that appear as part of some town name. However, this would produce a lot of garbage; many terms from town names are also common German or English words or surnames. To avoid this, we manually divided the set of all terms that appear town names into 3,000 *weak terms* that are common language terms And 55,000 *strong terms* that are almost uniquely used as town names. When parsing web pages, we first try to extract all strong terms. Next, we look for any weak terms that appear together with the extracted strong terms in the same town name. The underlying idea is that we try to find a town's main name first and then parse for weak terms (often found in the descriptive suffixes and prefixes) to resolve any ambiguity.

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

[1] near the city of Cologne, on the river Main, in the state of Saxony

[2] *Main* and *Oder* are names of rivers; however, *Oder* is also the German word for *or*.

To further increase the precision of the extraction, we assigned *killer terms* and *validator terms* to the strong terms. Any appearance of a strong term will be ignored if one of its killer terms also appears within some distance. Also, if a strong term has a validator term assigned to it, then any appearance of the strong term will be ignored unless the validator term appears within some distance. This allows us to handle town names that are also normal German words. We also introduced a list of *general killers* such that any strong term within some distnace of a general killer will be ignored. This list was filled with 3,500 common first names and titles such as *Mr.*, *Mrs.*, or *Dr.* to avoid mistaking surnames for town names. More details are given in [20], discussing phone numbers and zip codes.

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, thus to geographic locations. The problem, is that some terms can point to several town names, called *geo-geo ambiguity* [1]. Some towns share the same main name, a town's main name might even appear in another town's descriptive terms. We make two assumptions about the usage of town names that allow us to define rules to resolve 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* [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 has a strong soccer team", it might be acceptable to only assign this page to one of the two towns.

The second assumption is that the author most likely meant the largest town with that name. There are for examples two towns with the name *Göttingen*, a larger city and a tiny village, situated about 150 km apart. One expects that there are more pages about the larger of the two towns. The page will therefore be assigned to the city, not the village, unless there are other strong indications. As before, it can be argued that the failure of this hypothesis only introduces a marginal error, especially when the difference in size is huge.

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 of the algorithms is 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 various ways. None of them performs well on its own, but in combination they prove adequate for deciding the best of several possible matches.

One simple measure is 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. If a *numeric marker* such as a zip code is found, then this will usually resolve any ambiguity. Another approach is based on looking for *nearby towns*. If we find both `Frankfurt` and `Offenbach`, we can be pretty certain that the page intends to talk about *Frankfurt am Main*.[3] In our application we employed a simplified version, using Germany's administrative hierarchy as an indicator of distance. This measure can be looked up from a table quickly, 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 1.

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 1: Basic steps of the BB-First algorithm

In our implementation, we measured the size of towns only by sorting them into *villages* and *cities*, thus running 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 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 contains a geographic marker has an associated non-empty 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.

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 associated with them. This is not unexpected, since not every document contains a geographic reference in its text. On the other hand, 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

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

addresses in separate pages from the actual content that a user might be looking for. These issues 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 a random pair of pages [10]. This assumption can be 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 states that two pages in the same site are also more likely to be similar. For documents from the same sub-domain, host, or directory within a site, even stronger statements. This can also 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. Thus, at least one page in any given site by law 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 α, $0 < \alpha < 1$. 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 does not converge, and has to be handled with care. If geo propagation is performed too often, every single document could end up with a footprint spanning the entire country. In practice, one or two steps seem to give most of the benefit, and proper dampening factors plus lossy compression (simplification) prevents footprint sizes from getting out of hand. This results in an increased number of pages with non-zero footprints and an increased number of non-zero tiles therein.

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 Pagerank algorithm [6]. Footprints are sorted on disk by destination page and then aggregated into the footprint of the destination page. Propagation was also performed within sites. Finally, resulting footprints need to be normalized. 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 added into pages' footprints or used separately during query processing.

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 search results, and we discuss here only some basic approaches. In particular, the area of interest could be automatically extracted from a keyword query, by looking for terms that match a city or other geographic term and replacing it by a suitable query footprint. The automatic identification of queries that are geographical in nature is discussed in [17]. Or alternatively, users 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 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 now describe the actual query processing in two passes, first on an abstract level and later in terms of our actual current implementation.

5.1 Basic Geographic Query Execution

Without too much detail, the differences between geographic search and traditional engines are best outlined on a very abstract level. In a nutshell, a standard search engine works as follows: (i) The user inputs a set of search terms. (ii) The engine determines set of pages that contain all the search terms from the inverted index. (iii) It 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: (i) The user inputs search terms *and* a query region that is converted into a *query footprint*. (ii) The engine then uses the keywords *and* the query footprint to determine the set of pages that are in the intersection of the inverted lists and whose footprint has a nonempty intersection with the query footprint. (iii) 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 query footprint, to retrieve candidates results during the second step and third step. In our case, the first step is simply a question of interface design. The second step is also fairly similar to standard engines, at least on a high level, except that now only those pages survive that contain all search terms and have a non-empty intersection. The final ranking is a little different, since it has to merge two unrelated ranking measures, importance and geographic proximity.

5.2 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. One approach would be to simply use the query footprint as a filter, removing all results "outside" the query area, and then using the standard ranking. 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 continuum in between, allowing users to select their own preferences. First, they can choose different shapes for the query footprint as shown at the top of Figure 1. If a user prefers a sharp cutoff at a distance of say 10 km, she selects the footprint on the left, while the query footprint on the right models a more gradual approach. During the ranking phase, we compute a *geographic score* for each page in the intersection of the inverted lists of the query terms, based, e.g., on the volume of the intersection or the vector product between query and document footprint; see the bottom of Figure 1. If the score is zero, the document is discarded. Second, she can choose the relative weight of term-based and geographic components in the ranking. 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. Both, shape and relative weighting of the scores, can be provided by the user through a simple sliders, modeling the trade-offs and allowing interactive re-

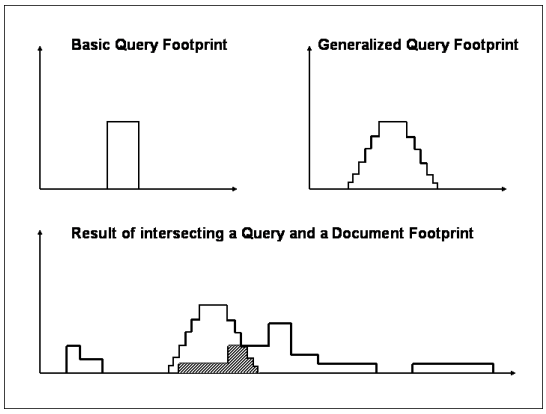

Figure 1: An illustration of footprints in a single spatial dimension. At the top, we have a query footprint with a distance threshold (left), and a footprint for a query that gives a lower score for documents that are farther away (right). At the bottom, we show an intersection between a query footprint and a document footprint.

ordering the results.

5.3 Efficient Geographic Query Processing

Given this ranking approach, we now describe query processing in more detail. Figure 2 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 memory (shown here only as document IDs), and their intersection is computed. For any document in the intersection, there are two lookups. First, we maintain an in-memory table of conservatively approximated document footprints, obtained by lossy-compressing the footprint structures down to a size of at most 100 to 200 bytes each. We lookup in this table to check if the intersection between the query footprint and the document footprint is nonempty; if so, we compute an approximation of the geographic score of the document. Next, we perform a lookup into an in-memory table of Pagerank values to compute a final approximate document score.

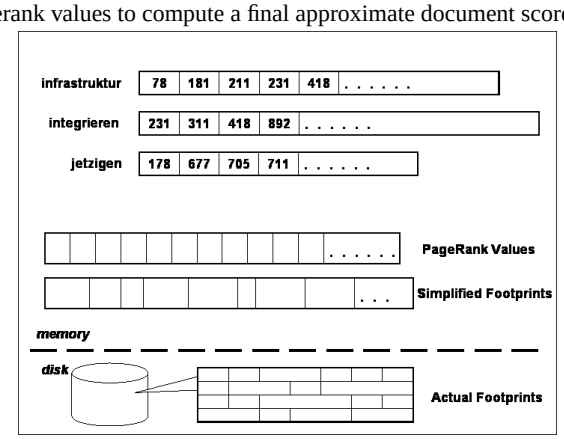

Figure 2: Organization of index structures, lookup tables, and geographic footprints in a scalable geographic engine.

After having traversed the inverted lists and determined say the top-50 results, we can perform a more precise computation of their geographic scores by fetching footprints from disk. There are a number of other performance optimizations in search engines, such as index compression, caching, and pruning techniques [18], that are omitted here. By integrating these, we achieve query throughput comparable to that of a standard non-geographic engine.

When compressed to 100 or 200 bytes, several million pages' footprints can 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 to sustain rates of a few queries per second.

6. CONCLUSION

This paper outlined design and implementation of a crawl-based geographic search engine for the German web domain. We described in detail how to extract geographic footprints from crawled pages through a *geo coding* process consisting of *geo extraction*, *geo matching*, and *geo propagation*, and discussed ranking and query processing in geographic search engines. Our prototype should be available online soon. One open issue for the near future is an appropriate evaluation of the quality of our footprints and query results.

Beyond this, 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 automatically identifying and exploiting terms such as "Oktoberfest", that are not listed in geographic databases but clearly indicate a particular location, through the use of data mining techniques. Second, we are looking at optimized query processing algorithms for geographic search engines. Third, we study focused crawling strategies [5] that can efficiently fetch pages relevant to a given geographic area, running across many top-level domains. Finally, we are interested in *geographic mining* of the web.

Finally, we like to thank Thomas Brinkhoff for cooperation on earlier work [19] leading up to this project, and for continuing feedback and support.

7. 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*, 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 Int. Conf. on Information and Knowledge Management*, 2002.

[7] Open GIS Consortium. `http://www.opengis.org`.

[8] A. Daviel. April 1999. `http://geotags.com`.

[9] A. Daviel. Geographic registration of HTML documents. IETF Draft, July 2003. `geotags.com/geo/draft-daviel-html-geo-tag-06.html`

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

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

[12] Divine inc. Northern light geosearch. Last accessed February 2003.

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

[14] Eventax GmbH. `http://www.umkreisfinder.de`.

[15] Google Inc. Google Local Search, 2003.

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

[17] L. Gravano, V. Hatzivassiloglou, and R. Lichtenstein. Categorizing web queries according to geographical locality. In *Proc. of the 12th CIKM*, 2003.

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

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

[20] A. Markowetz, Y. Chen, T. Suel, X. Long, and B. Seeger. Design and Implementation of a Geographic Web Search Engine. Technical Report TR-CIS-2005-03, CIS Department, Polytechnic University, February 2005.

[21] K. McCurley. Geospatial mapping and navigation of the web. In *Proc. of the 10th World Wide Web Conference*, pages 221–229, May 2001.

[22] Räber Information Management GmbH. `http://www.search.ch`.

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

[24] Yahoo! inc. Yahoo Local Search, 2004. `http://local.yahoo.com`.

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