GeoSphereSearch: Context-Aware Geographic Web Search

[Extended Abstract]

Jens Graupmann
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
Saarbrücken, Germany
graupman@mpi-inf.mpg.de

Ralf Schenkel
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
Saarbrücken, Germany
schenkel@mpi-inf.mpg.de

ABSTRACT

This paper presents the GeoSphereSearch engine for context-aware geographic queries on the Web. It facilitates the formulation of geographic queries and the visual presentation of discrete or aggregated query results in an intuitive manner. In contrast to other approaches it does not assign geographic footprints to documents, but considers context-aware geographic information on the fly, allowing a fine-grained query evaluation on the level of document fragments, not complete documents.

1. INTRODUCTION

Exploiting geographic information in IR has been an important research topic for quite some time [7, 9, 11, 12, 14]. The popularity of geographic search engines has even increased recently with the advent of commercial Web-based map services and virtual globe applications like Google Earth[1] that enable everyone to use and explore geographic information.

To answer geographic queries on unstructured collections like the Web, existing geographic search engines typically maintain, for each page, a "geographic footprint" that consists of geographic information extracted from the content of the page or meta data like the URL [3, 5, 10, 15]. A *geoquery* is then a query that combines constraints on the content of a Web page with at least one geographic constraint, possibly including range conditions ("between Paris and Nancy") and uncertainty ("near London"). Such a query is usually answered by matching content constraints with content of pages and geographic constraints with geographic footprints, and the results is a ranked list of documents.

The main drawback of existing solutions for geoqueries is that, even though maintaining different geographic footprints for a Web page, they consider the whole page for content conditions. However, it is often the case that a page talks about several locations in different contexts, so context must be taken into account when answering a geoquery. As an example, assume that someone wants to find out which Nobel prize winners were born in Germany. Most Web pages that are relevant do not only contain the birth place, but also a lot of other locations like places where somebody worked, lived, and died. To find good matches, an engine should consider only locations within a context of matches for the content conditions 'Nobel prize' and 'born'. The GeoSphereSearch engines solves this problem by annotating geographic information within the page and aggregating it in the context of a content match on the fly, possibly including external knowledge like a hierarchy of locations.

A more difficult problem are queries where geographic information is not part of the query, but the answer to the query; a simple example are question-answering-style fact queries like "Where did Einstein die?" or list queries like "Where were Nobel prize winners born?". For such *unconstrained geoqueries*, GeoSphereSearch first computes the top-k results for the query (with k set to 100 or 200) and then collects geographic information from the context of results on their page. Optionally, these locations are then grouped to find frequently occurring locations (we call this step *geoclustering*), and the most frequent locations form the answer to the query; see Section 3.3 for details. As another example, suppose that the chair of a conference is looking for the venue of an upcoming database conference, with the constraint to choose a location near to as many university departments doing resarch on IR as possible. The system would first determine Web pages of the IR departments and their corrseponding locations. To visualize the result, GeoSphereSearch applies the freely available virtual globe application Google Earth.In the example, each location where a IR department is located would be visualized on a virtual globe, allowing the chair to pick a region where the visual density is high.

2. SPHERESEARCH

GeoSphereSearch is an extension of the SphereSearch Engine (SSE) [8], a powerful context-aware search engine for heterogeneous semistructured data that integrates information retrieval (IR) and information extraction (IE) techniques. Due to space limitations we only describe briefly some relevant aspects of SphereSearch's query and data model; more details can be found in [8].

SSE applies an expressive graph-based data model that represents the document structure, linkage between documents and annotations produced by information extraction tools. Here, each node in the graph roughly corresponds to an HTML tag in the document or an annotation, and edges correspond to nesting of tags or links between documents.

[1] `http://earth.google.com/`

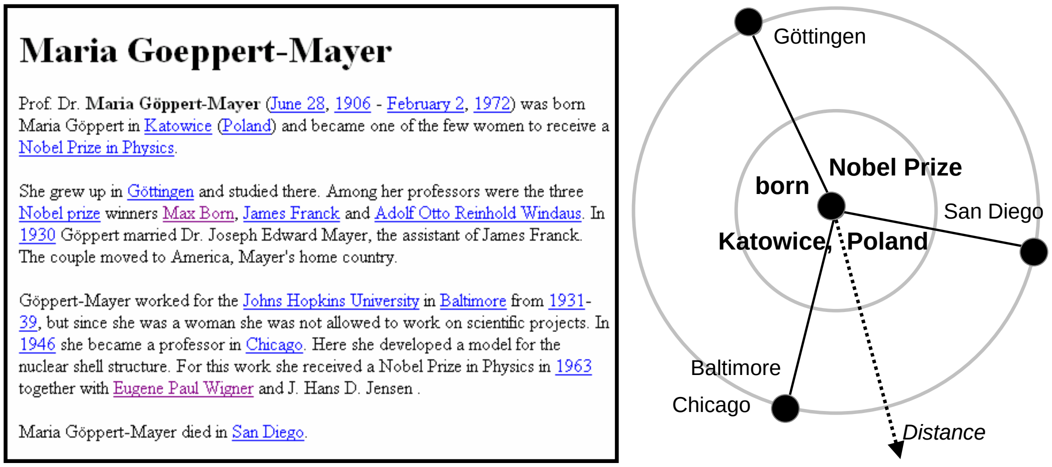


Figure 1: Sphere based scoring

Unlike a classical bag-of-words-model, this preserves document structure that can be exploited for ranking later. IE and other techniques are seamlessly integrated as the results of annotation steps (e.g., Named Entity Recognition) are simply added to the internal graph-based representation as additional labelled nodes. Such annotations comprise person names, dates, money amounts, and locations, and come together with a confidence value that expresses the expected correctness of the annotation. The current implementation uses the GATE system [6] to produce annotations, but it can be easily replaced or extended by other tools.

Queries in SSE consist of a set of keywords and so-called concept-value conditions that exploit the annotations. As an example, the query "hotel, location=Seattle" asks for hotels in seattle; to be more precise, it asks for occurrences of the term 'hotel' on a Web page where the location 'Seattle' occurs in a context. Using the ∼ operator, vagueness may be added to conditions like in "location=∼Seattle", requesting a hotel in or near Seattle.

Such queries are evaluated in a two-step process. First, elementary node scores for single nodes are computed: For keyword conditions, the standard BM25 scoring model is applied. For each concept-value condition, nodes whose label matches the concept in the condition (like 'location') are assigned an elementary node score for that condition that is a combination of the annotation confidence and, for similarity queries, its similarity with the value condition. The node score $ns(n)$ of a node n is then the sum of all its elementary node scores.

The second step of the evaluation adds context awareness by considering the surroundings of a node n, i.e., other nodes in a *D-sphere* $S_D(n)$ around n. Here, $S_D(n)$ comprises all nodes within a fixed distance D of n. The sphere score $s_D(n)$ of n is then an aggregation of the node scores of all nodes in its D-sphere, weighted by their distance to n; formally:

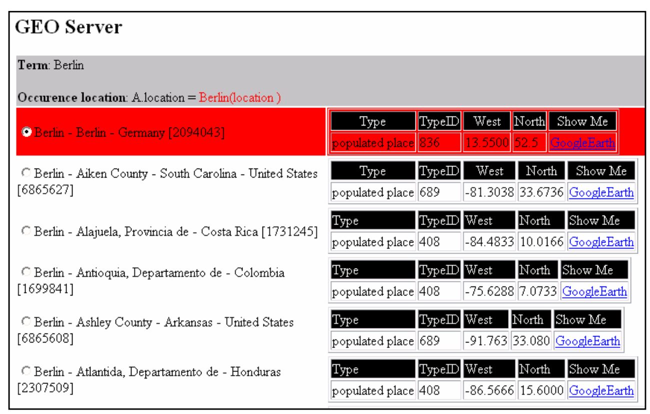
$$s_D(n) = \sum_{v \in S_D(n)} ns(v) * \alpha^{\delta(v,n)}$$

Here, $\delta(u, v)$ denotes the distance of nodes u and v, and the configurable damping factor α ($0 \leq \alpha \leq 1$) determines the contribution of nodes at greater distances.

3. GEOSPHERESEARCH

3.1 Geographic Server

In order to annotate geographic information the ANNIE component of the GATE system [6] is used. For evaluation of queries with geographic constraints GeoSphereSearch integrates a Geographic Server component based on the data obtained from the Alexandria digital library project [1]. This geographical database comprises about 4,000,000 places and geographical features with corresponding coordinates and hierarchical information like regions and countries of given locations. Locations that are identified in a page are not only tagged as location, but also annotated with the corresponding coordinates, disambiguating ambiguous location names using the hierarchical information (similarly to [13]).



Type	TypeID	West	North	Show Me
populated place	836	13.5500	52.5	GoogleEarth
populated place	689	-81.3038	33.6736	GoogleEarth
populated place	408	-84.4833	10.0166	GoogleEarth
populated place	408	-75.6288	7.0733	GoogleEarth
populated place	689	-91.763	33.080	GoogleEarth
populated place	408	-86.5666	15.6000	GoogleEarth

Figure 2: Feedback request

3.2 Geoqueries

A query containing a condition with the concept *location* triggers the usage of the geographic module. First, the location is looked up in the geographical database to determine wether the location's name is unique or ambigous. If more than one location with the specified name exists, it has to be disambiguated; GeoSphereSearch shows all possible mappings to the user and prompts for feedback. Figure 2 shows such a geographical feedback request with the most probable choice preselected.

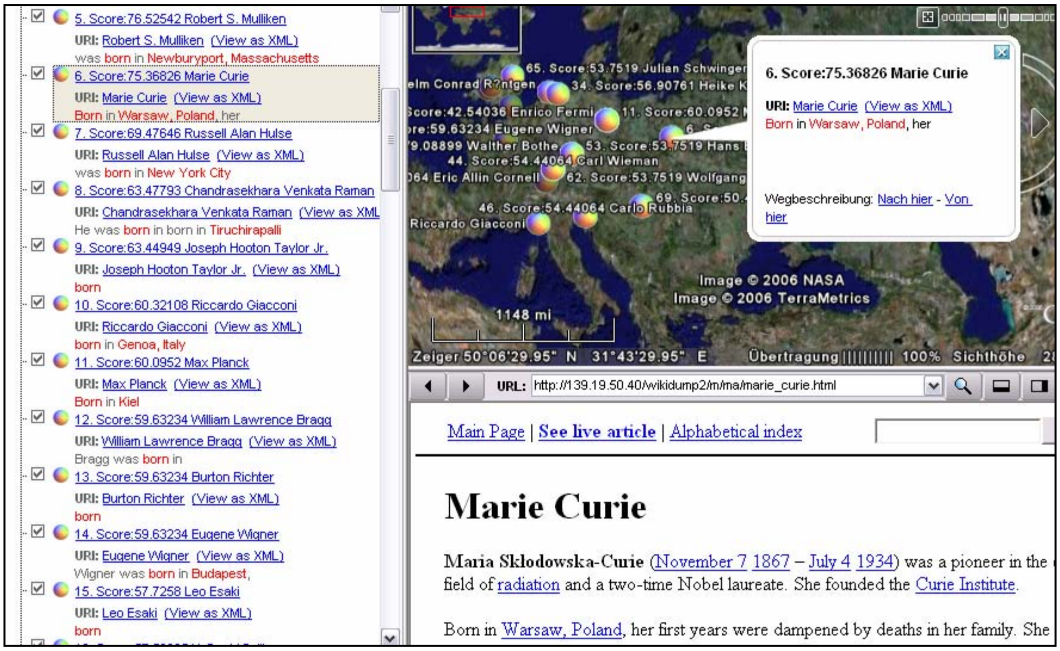


Figure 3: SphereSearch result in Google Earth

Besides exact-match conditions, GeoSphereSearch supports two special kinds of geographical conditions: *similarity* and *range* conditions. To evaluate a query like "hotel=∼Seattle“, GeoSphereSearch first determines the coordinates of Seattle and expands the query with nearby locations in a default radius using the Geographic Server. If the default radius is not appropriate for the user's needs, it can be adjusted via user feedback ("too far away“), the query is then re-evaluated with an adjusted radius.

Range conditions are a special kind of region queries that restrict possible locations to an aera between two fixed end points. As an example, consider that somebody searches for romanic-style buildings along the route from London to Oxford. The corresponding GeoSphereSearch query would be "romanic, building, location=Oxford-London“. After disambiguating the locations, a corridor between Oxford and London is computed, the locations that it contains are extracted from the geographical database and (conceptually) added to the original query.

3.3 Unconstrained Geoqueries and Geoclustering

Queries that have geographic information as answer are formulated with a location condition that contains a wildcard symbol '?'. As an example, consider again the query from the Introduction that searches for birth places of Nobel prize winners. In GeoSphereSearch, that query would be formulated as "nobel prize, born, location=?“.

To evaluate this query, GeoSphereSearch first determines, according to its ranking model, the best k nodes (where k typically is 100 or 200) with highest sphere scores for the terms 'nobel prize' and 'born'; we call these nodes *target* nodes. In a second step the geographic wild card condition is evalutated; it is not used for result ranking, but instructs the engine to aggregate location-related information (i.e., location annotations) near target nodes. The location information is aggregated with additional weights reflecting confidence scores and proximity to target nodes within the graph-based document representation, similar to the sphere-based scoring model sketched in Section 2.

Figure 1 shows a document containing a match for the mentioned query and spheres around the first paragraph that is the target node as it contains the terms 'Nobel prize' and 'born'. Based on the scoring model Katowice is the highest ranked location with respect to that target node. A different query asking for the places of death of Nobel prize winners would rank San Diego highest in this document.

The whole result of the query can then be either clustered or unclustered. For the unclustered result, each target node is annotated with the highest ranked location in its surrounding, yielding a combined view of the resulting location together with the content in the page. For many queries, however, a clustered version of the result that aggregates all occurrences of the same location could be better. To cluster the results, the scores of all occurrences of the same location are summed up, yielding a ranked list of locations. For the example with birth places of nobel prize winners, this list would include cities like Ulm, Warsaw, and Paris.

3.4 Visualization

A key ingredient of a successful geographic search engine is an intuitive visual presentation of query results [2, 4]. GeoSphereSearch allows to visualize both clustered and unclustered results with Google Earth. This free-of-charge virtual globe program can be seen as a Geographic Information System itself. It displays satellite images wrapped around a virtual global and allows to place annotations on the globe and search for locations and annotations. Additionally, it features an integrated Web browser view and provides an import data format to connect it to other applications.

Figure 3 shows the Google Earth user interface presenting the unclustered result of the query introduced above. This kind of visual presentation permits the following intuitive ways of exploring the query result:

1. *Result List*: A match in the result list can be selected (Panel A). The globe rotates and zooms to the location connected to that match. Other matches in the surrounding of this match can be seen. Additionally, the corresponding result document is shown in the web browser at the bottom of the Google Earth interface.

2. *Geographic Exploration*: The virtual globe can be turned and zoomed using the mouse. All matches are marked with a sphere symbol. By exploring the result set in this manner the geographic distribution of matches, e.g. regional clusters, can be judged and identified. Selecting a match on the globe highlights the corresponding entry in the result list and shows the corresponding document in the browser pane.

4. CONCLUSION

We demonstrated how geographic information can be intuitively integrated in context-aware Web search. GeoSphere-Search geographic extensions facilitate context-aware query evaluation of geographical condition. Furthermore its visual representation allows to explore a query result on a virtual globe.

5. REFERENCES

[1] Gazetteer Development at the Alexandria Digital Library Project. http://www.alexandria.ucsb.edu/gazetteer/.

[2] R. Albertoni, A. Bertone, and M. D. Martino. Information search: The challenge of integrating information visualization and semantic web. In *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA 2005), Copenhagen, Denmark, August 2005.*

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

[4] M. B. Carmo, S. Freitas, A. P. Afonso, and A. P. Cláudio. Filtering mechanisms for the visualization of geo-referenced information. In *Proceedings of the 2005 Workshop On Geographic Information Retrieval (GIR 2005), Bremen, Germany, November 2005.*

[5] P. Clough. Extracting metadata for spatially-aware information retrieval on the internet. In *Proceedings of the 2005 Workshop On Geographic Information Retrieval (GIR 2005), Bremen, Germany, November 2005.*

[6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.*

[7] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of web resources. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000), Cairo, Egypt, September 2000.*

[8] J. Graupmann, R. Schenkel, and G. Weikum. The spheresearch engine for unified ranked retrieval of heterogeneous xml and web documents. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005), Trondheim, Norway, 2005.*

[9] C. B. Jones, A. I. Abdelmoty, D. Finch, G. Fu, and S. Vaid. The spirit spatial search engine: Architecture, ontologies and spatial indexing. In *Proceedings of the 3rd International Conference Geographic Information Science (GIScience 2004), Adelphi, MD, USA, October 2004.*

[10] B. Martins, M. S. Chaves, and M. J. Silva. Assigning geographical scopes to web pages. In *Proceedings of the 28th European Conference on IR Research (ECIR 2006), London, UK, April 2006.*

[11] B. Martins, M. J. Silva, and M. S. Chaves. Challenges and resources for evaluating geographical ir. In *Proceedings of the 2005 Workshop On Geographic Information Retrieval (GIR 2005), Bremen, Germany, November 2005.*

[12] K. S. McCurley. Geospatial mapping and navigation of the web. In *Proceedings of the 10th international Conference on World Wide Web (WWW 2001), Hong Kong, China, May 2001.*

[13] E. Rauch, M. Bukatin, and K. Baker. A confidence-based framework for disambiguating geographic terms. In *Proceedings of the Workshop on the Analysis of Geographic References, Edmonton, Alberta, May 2003.*

[14] T. Tezuka, T. Kurashima, and K. Tanaka. Toward tighter integration of web search with a geographic information system. In *Proceedings of the 15th international Conference on World Wide Web (WWW 2006), Edinburgh, Scotland, May 2006.*

[15] L. Wang, C. Wang, X. Xie, J. Forman, Y. Lu, W.-Y. Ma, and Y. Li. Detecting dominant locations from search queries. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005), Salvador, Brazil, August 2005.*