LocateThisPage: Drive-by Location-Aware Browsing

Chandan KUMAR [a] Dirk AHLERS [b] and Susanne BOLL [c]
[a] *OFFIS – Institute for Information Technology, Oldenburg, Germany, chandan.kumar@offis.de*
[b] *UNITEC – Universidad Tecnológica Centroamericana, Tegucigalpa, Honduras, ahlers@dhere.de*
[c] *University of Oldenburg, Oldenburg, Germany, susanne.boll@uni-oldenburg.de*

Abstract. Many Web pages contain spatial information and have references to a physical location. A user browsing the Web might like to identify the real world address referencing the contents of a visited page, or to locate the page on a map. In this paper we propose our location extraction system as the browser plugin *LocateThisPage* that retrieves and visualizes location references on visited Web pages and permits easy interaction with the location. The proposed application makes use of the spatial information present in the HTML content and metadata of Web pages, through the adaption and integration of previous work in geospatial search engines and geoparsers. While location-based search engines allow a user to active search for geospatial information, we additionally aim to support a complementary use case of an unobtrusive notification about locations currently browsed by the user. This integrates location awareness into the browsing process itself so that users are informed about the availability of location information and can make direct use of it in a geospatially enhanced Web.

Keywords. Geographic Information Retrieval, Location-Based Search, Web Browsing, Entity Extraction, Geoparsing, Geocoding

1. Introduction

Web search engines index spatial information and provide access to it, such as local search providers, yellow pages, or local Web2.0 services. They allow searching for information from a specific area. On the other hand, users may already have found a resource and now want to know its geospatial characteristics, i.e., a user does not want to search for some service at a location, but already found an interesting Web page and wants to know where the described entity is located. In this paper we provide a solution of this user need with a location extraction application directly in the browser. We propose a browser plugin that retrieves and visualizes German addresses on Web sites and permits easy interaction with the location references of a visited page.

The most prominent location-based services today are local search services such as Google Maps, Yahoo! Maps or Bing Maps for stationary or nomadic users. These service typically provide a map-based view on Web content such as companies, restaurants or events. Currently, efforts are underway to make location a main concept within the Web

infrastructure itself [1,2]. Furthermore, user's current location can be utilized for innovative Web applications, such as the Yahoo! Fireeagle location broker[1] or Mozilla Labs' Geode[2].

Several applications already exist that aim to ease information handling within a browser. Taking a document-centric view, these identify, extract, annotate, display, or provide information on a viewed Web page. One example that offers direct added value is the Skype toolbar[3] which not only identifies phone numbers, but gives the opportunity to directly call them in a sort of action-by-selection. Firefox extensions such as GeoURL[4] and DublinCore Viewer[5] both show relevant metadata, with the former also offering to pass location metadata to a mapping service. GeoLocateFox[6] additionally shows a map preview. In other contexts, Mac OS X has introduced data detectors[7] for the mail application that detects dates, times, phone numbers, or addresses and processes them with the respective applications. In a discontinued project, Microsoft introduced so-called Smart Tags into Word and Internet Explorer that could highlight dates or person names and provide actions on them. A subsequent project, the later Internet Explorer Accelerators[8], allow selection-based search, meaning that text has to be selected and then certain actions are made available based on the selection. The PiggyBank system [3] extracts structured information from Web pages by a defined set of screenscrapers and allows aggregated views over the data, effectively generating semantic wrappers. The W3C Tabulator project[9] provides a semantic data browser for RDF data on the Web and offers views for geospatial information.

However, since semantically structured data is not yet widely available on the Web, we aim for a solution that can also semantically extract the location from arbitrary unstructured documents. We are motivated by the fact that information at the high granularity of address-level data is available on a massive amount of common Web pages. It is therefore a primary focus of our work in geospatially aware search engines [4,5]. Based on our existing work on geoparsers [6], we present *LocateThisPage*, a Web-document-centric location search implemented as a Firefox plugin. With this plugin, we approach the issue of geospatial information from the perspective of a user equipped with a common browser surfing the Web and interacting with a Web page. The plugin reveals the relevant location information from a Web site and makes it intuitively accessible and useful[7].

2. LocateThisPage System Design

The plugin identifies and annotates location references on the currently viewed Web page and allows for easy and intuitive interaction with the spatial data. The semantic location retrieval process comprises methods of parsing, identification, entity extraction, and

[1] http://fireeagle.yahoo.net/
[2] http://labs.mozilla.com/2008/10/introducing-geode/
[3] http://www.skype.com/help/guides/ffextension/
[4] https://addons.mozilla.org/en-US/firefox/addon/geo/
[5] https://addons.mozilla.org/en-US/firefox/addon/dublin-core-viewer/
[6] https://addons.mozilla.org/en-US/firefox/addon/geolocatefox/
[7] http://docs.info.apple.com/article.html?path=Mac/10.7/en/mh35744.html
[8] http://msdn.microsoft.com/en-us/library/cc289775(VS.85).aspx
[9] http://www.w3.org/2005/ajar/tab

Figure 1. Exemplary page with location references

location disambiguation to identify actual street addresses within a page. If the current page does not contain a location, the process can be extended to related pages by subpage parsing. Once the location is identified, the next phase of annotation, visualization, and interaction makes it accessible within the plugin by spatial links and menus that provide location-related service. We have chosen the Mozilla Firefox browser as the preferred system for the client-side extraction as from our point of view, it has a very good ecosystem and a well-documented plugin development process. A Firefox extension is created by combining well-known technologies[10]. The description of the user interface and the integration into the browser is handled by XUL, the XML-based user interface description language of the Mozilla project; the functionality is implemented in JavaScript.

2.1. Client-side extraction of location information from a Web page

The objective is to extract precise spatial information such as addresses or coordinates that provides a physical location that is related to the Web page itself. The system uses two sources of location data within a Web page. The central resource for our geoparser is location references that are found within the page's content. This is not as simple as it may seem. Even though the location information may be present on a page and visible to a user, the actual text with the location information is rather unstructured and can be scattered in the page's HTML content. Additionally, the page's metadata is examined for structured annotation; however this is rather rare on the Web. So, both types of location data are "hidden" in a way and need to be identified and visualized to become actionable data. The challenge in porting our server-side address geoparser algorithms over to a client-side system lies in the complete redesign of the extraction and verification stages. In our previous work [6], we have developed a geoparser that extensively uses external knowledge in the form of an address-level gazetteer. This allows it to combine extraction and verification steps, which is especially helpful for German addresses, which are otherwise more difficult to validate as their components follow less strict rules. With the changed environment, we had to rethink our parsing strategies and adapt them without reducing the identification performance too much. With only generic parsers or screen-scrapers applicable, a main issue is robustness of the extraction. The client-side plugin should work as a stand-alone system and be able to work without communication to our server. As we obviously cannot equip it with a full address database for Germany, it has to perform the address identification without this external knowledge completely on the client side.

[10] https://developer.mozilla.org/en/Building_an_Extension

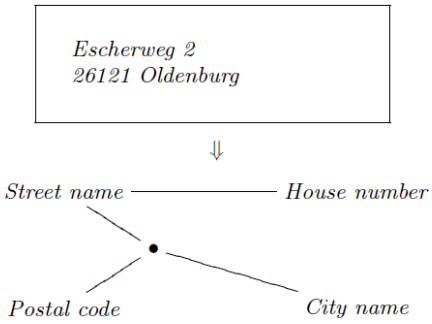

Figure 2. Exemplary German address and decomposition into individual components

A typical full German address is shown in Figure 2 with the decomposition into the four typical primary components indicative for an address. The components have a large variety of possible combinations. We seek these components and identify their possible combinations to build up a location reference up to a full address. The reliable identification of these addresses depends upon the recognition of the individual components and their co-occurrence. The basic pattern consists of a sequence of terms representing street name, a numerical term for a house number, a 5-digit term for the postal code, and a city name; these can occur in different sequences. Textual annotations between the terms can be used as supporting terms. Various heuristics for individual components as well as their relation to each other that form a full address are used.

While on the server side, identification and verification is an interdependent single process, for the client side, we follow a multi-step approach. The identification and extraction of location references within the textual content and the metadata is done by rule-based information extraction. The identification of location references in a page's metadata is done rather simple by checking the presence of the respective fields. The icbm and geo fields are single values of the meta header of a page and can be easily extracted. The coordinate and address microformats[11] require only slightly more parsing, as the microformat classes form an easily identifiable hierarchy within the DOM tree.

The address identification requires more efforts, as parts of an address can be scattered throughout the page source. In a first step, general regular expressions first identify promising sections of the Web page with patterns potentially resembling an address as described above. In a second step, more specialized regular expressions as well as programmatic checks try to identify actual address components. A scoring system allows grading the possibility of terms to be an address part and also to grade the possibility of a group of terms to be an actual address. Subsequently, cross-validation of address part candidates as well as plausibility checks are performed to try to find and arrange them into the best match.

In many situations, the address on the content of a Web page exhibits no continuous character. For descriptive or formatting reasons, non-address terms appear in-between address terms. This is reflected in the regular expressions and the subsequent steps. The plugin recursively scans the DOM tree of the HTML page. It searches through the sub-

[11] http://microformats.org/wiki/adr

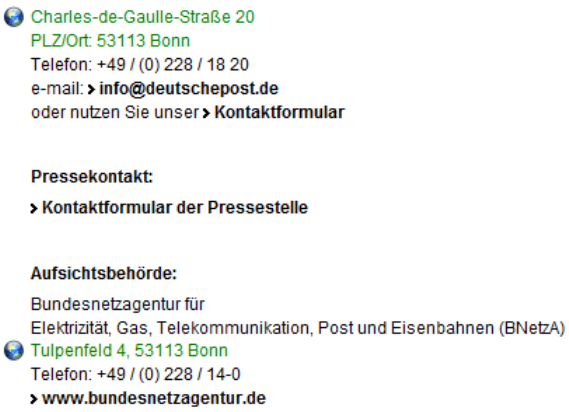

Figure 3. A Web page with two location references: Highlighted address

trees by generating a plaintext version to apply the initial set of regular expressions to identify candidates, but also maintains the position within the DOM structure. While the text version is easier to handle for general patterns, the structure gives additional hints towards word boundaries, grouping, or supporting terms. This is necessary to apply the visualization as seen in Figure 3, but also improves the identification process. It allows for an improved performance, taking the visual layout of the page into account, especially in nested table structures. Thus the system is able to find addresses written with the individual components spread over either rows or columns in complex tables. It parses tables in both possible directions to find the best match according top plausibility checks. To mark candidate parts and candidate addresses as well as identified addresses, it injects specific out-of-band tags into the DOM tree that can later be styled. This allows it to also handle intersections with other found address parts or other addresses on a page.

The client-only identification process for addresses works without connection to a database or gazetteer. Therefore, it will identify address patterns on a page, but cannot perform actual validation. However, its performance is still very strong and only falls a little bit behind the gazetteer-based geoparser for general, unstructured content of Web pages. Informal tests show more than 90% of correctly identified addresses, a fact that is surely helped by the care that is often taken in writing out addresses. However, the system can also identify addresses inside a paragraph, so a separate block for the address only supports the extraction, but is not mandated.

2.2. Exploration of location-relevant links of a Web page

In trying to find the location related to the Web page content, users should not be restricted to only the current page. In some cases, the answer to the question “where is this service located?” is not available on the current page. Similar to how a user would start to click through the site structure to find contact information, the plugin performs a similar action. This saves the user some work and can actually annotate and highlight those links on a page that lead to potentially relevant address information. This then allows to enhance those links with preview information [8].

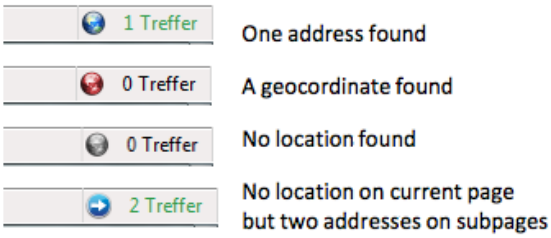

Figure 4. Status bar icon showing different states of the system

If the geoparser is unable to find any relevant address on the given Web page, the user may chose the option of a subpage parsing run. We of course do not want the system to retrieve all linked pages, therefore, some selection and prioritization is done. This is a form of link prefetching[12] that also pre-extracts page information. Due to the various criticisms of prefetching, we do not enable it be default, but only execute it if the user explicitly requests it. It therefore has to observe rules for prefetching links, especially not accessing POST content and aiming to reduce potentially state-changing GET requests. One rule ensures that only URLs without query part are accessed to reduce the chance of accessing non-safe actions on a server implemented with non-safe methods (cf. RFC 2616). Additionally, the plugin tries to only access textual file types.

The links that are actually accessed by the exploration should mainly be such promising links that have a high possibility to contain address information. This means that the system has to compute a relevance ranking on unseen pages [9]. For this information discovery purpose a mini-crawler is built based on some lightweight features of previously proposed geospatially focused crawling techniques [5] to retrieve linked pages that might carry the relevant location reference. It uses heuristics of previously extracted promising keywords (“contacts", “imprint", etc.) to quickly discover pages containing location-relevant information. The purpose of the function is to retrieve supporting addresses for the current page. These will mostly only be found on linked pages the current domain, therefore it implements a same-domain policy, which limits the amount of links under consideration. Furthermore, only a limited number of outgoing links are examined, after they have been prioritized according to their perceived relevance. This function provides the user the functionality to uncover location information within a given website without having to manually check all outgoing links.

2.3. Visualization and interaction with spatial information

The system presents found information in two ways, first, by indicating the overall status by a global icon, and second, by highlighting addresses within the page. The state of the matching for a page is visible in the status bar of the browser by a small icon representation and textual information about the number of locations on the current page. The icon is a colored globe; its color shows the different states: gray (no matches), red (only geocoordinate match), blue (only address match) or green (geocoordinate and address match). The geocoordinate match signifies the location references found in the metadata

[12] https://developer.mozilla.org/en/Link_prefetching_FAQ

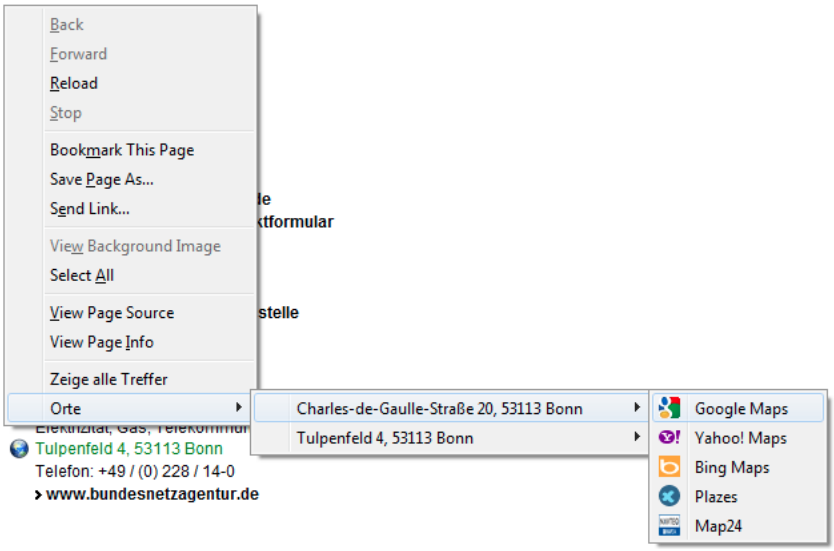

Figure 5. Context menu items to access the location and related services

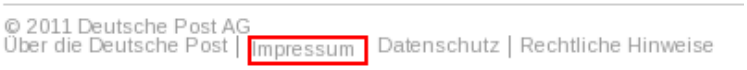

Figure 6. Highlighted link to an address match on a related page

information of a page, not in the textual content. When an address is found on a subpage instead of the current page, an arrow is displayed instead of the earth icon, with the same color schemes. Some of these states are shown in Figure 4. The visual annotation (highlighting) of found addresses directly on a page also provides an easy and intuitive access to the spatial information (Figure 3). It is also accessibly by an entry in the context menu when used over the found address. This allows individual access to each matched address.

The icon in the status bar panel provides access to the action menu of the system. The menu displays several items corresponding to the state of the plugin. For all identified locations, we maintain a list of possible actions such as service calls that point to a service or service interactions that can enhance the retrieved information, this allows the user to quickly choose between a set of tools suitable for the task, similar to the selection of mapping services for Wikipedia[13] but for all Web pages carrying respective information. It also allows accessing the options of the system, initiating the subpage crawling, and the service manager to add or modify services. Figure 5 shows the menu items to access the found locations in the current page. The location tools are mapping views, routing services, verification or comparison, search services, or social networks. If a subpage parsing was performed, an entry is shown which enables the user to visit the subpage containing addresses.

[13] http://en.wikipedia.org/wiki/Template:GeoTemplate

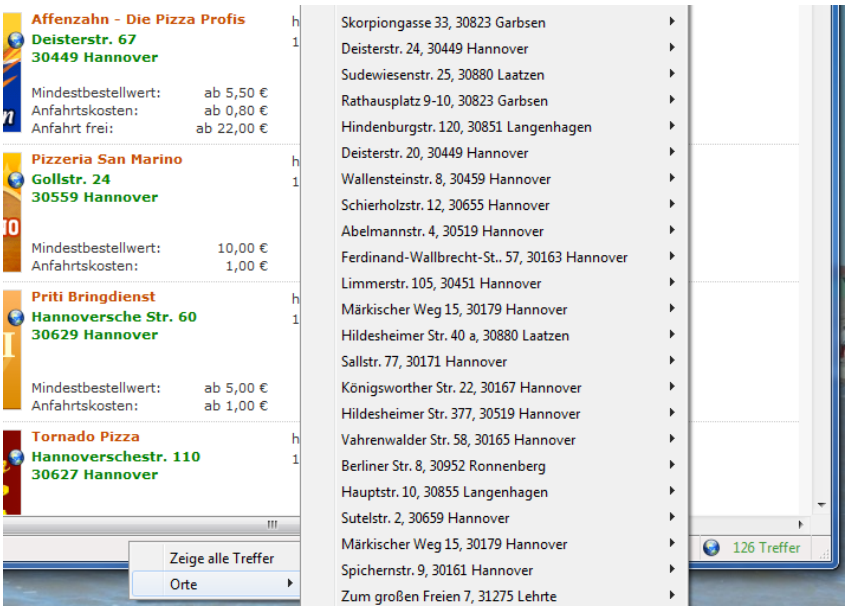

Figure 7. Access of multiple location references from the status bar

The links of subpages which contain relevant addresses are also highlighted in the page itself (Figure 6) and can be used. The plugin provides an initial list of mapping providers. A configuration dialog allows the user to enable or disable the address search as well as enable or disable subpage parsing. Additionally the user can change the JSON structure holding the information about the mapping providers, which allows adding more services.

What has been described to this point allows to map each address individually, but often, the distribution of all locations in a document, the document's footprint [10], is of interest. A very simple geospatial footprint without further analysis steps is the mapping of the point cloud of all locations of the document. While most mapping services only allow mapping one location per query, there are small number of wrapper services that take in more than one location. At this point, we have integrated the service of a third party website[14] using the Google Maps API. This can also be extended to use other services[15], to generate a KML file on the fly to show on a mapping service, or, finally, this might also be more tightly integrated into the rest of our geospatial search infrastructure on our own maps. Since currently only one service is used, the plugin does not show a list of mapping providers, but only displays the direct function ("Zeige alle Treffer", Figure 7). If further analysis is wanted a service could be deployed on our own systems, but the focus at this step was to mainly use public services. This functionality can generate a map as shown in Figure 8 which allows to visualize a long list of matched addresses (Figure 7) as a point cloud.

[14] http://gmaps.kaeding.name
[15] (e.g., http://batchgeo.com/ or http://www.multiplottr.com/)

Figure 8. Mapping of all found addresses on the page of Figure 7

3. Conclusion and Future Work

This paper described the *LocateThisPage* browser plugin to provide document-centric location tools for end user. The system discovers the location references of a Web Site using the crawler to find location relevant pages and rule based extraction to locate addresses within a Web page. While use of such technologies is usually found server-side local search applications, we proposed it as an independent rule-based client side application at the quick disposal of the user. The location annotations are extracted from the content of common Web pages to tie them to useful location-based services. The identification of addresses and metadata location references is performed only on the client without assistance by gazetteers.

The address extraction algorithms are currently centered towards German Web pages. We plan on developing the plugin towards a community tool that, e.g., explores the use of location-based social bookmarking and also aim to develop extractors for other countries. For the future, we might add verification against the server-side systems, which include a full gazetteer. For even tighter integration with the server-side components, we envision learning from user-supplied information about page evaluations and the performance of our technology. Verified information may be added to our search engine database to improve the spatial Web index and conversely, support user decisions, such as finding authoritative sources for certain location references or georeferenced entities.

Acknowledgements

We thank our student Leon Winter for implementing the plugin functionalities.

References

[1] A. Cooper, D. Mulligan, H. Schulzrinne, E. Wilde, Challenges for the Location-Aware Web. Web Science Conference, 2010.

[2] E. Wilde and M. Kofahl. The Locative Web. In LocWeb2008. ACM, 2008.

[3] D. Huynh, S. Mazzocchi, and D. Karger. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. ISWC '05, 2005.

[4] D. Ahlers and S. Boll. Location-based Web Search. In A. Scharl and K. Tochterman, editors, The Geospatial Web. Springer, London, 2007.

[5] D. Ahlers and S. Boll. Adaptive Geospatially Focused Crawling. In Proceedings of the 18th Conference on Information and Knowledge Management, CIKM09, 2009.

[6] D. Ahlers and S. Boll. Retrieving Address-based Locations from the Web. In GIR '08: 5th Workshop on Geographic Information Retrieval, 2008.

[7] S. Boll and D. Ahlers. A Web more Geospatial: Insights into the Location Inside. In D. De Roure and W. Hall, editors, WebEvolve2008, 2008.

[8] H. Weinreich, H. Obendorf, W. Lamersdorf, The Look of the Link - Concepts for the User Interface of Extended Hyperlinks. HYPERTEXT '01, 2001.

[9] S. Chakrabarti, K. Punera, M. Subramanyam, Accelerated Focused Crawling through Online Relevance Feedback. WWW '02, 2002.

[10] A. Markowetz, T. Brinkhoff, B. Seeger, Geographic Information Retrieval. 3rd International Workshop on Web Dynamics, 2004.