Pers Ubiquit Comput (2014) 18:1047–1060
DOI 10.1007/s00779-013-0718-3

ORIGINAL ARTICLE

Toward location-aware Web: extraction method, applications and evaluation

Basil Hess · Fabio Magagna · Juliana Sutanto

Received: 30 November 2012 / Accepted: 3 July 2013 / Published online: 8 October 2013

Abstract Location-based services (LBS) belong to one of the most popular types of services today. However, a recurring issue is that most of the content in LBS has to be created from scratch and needs to be explicitly tagged to locations, which makes existing Web content not directly usable for LBS. In this paper, we aim at making Web sites location-aware and feed this information to LBS. Our approach toward location-aware Web is threefold: First, we present a location extraction method: SALT. It receives Web sites as input and equips them with location tags. Compared to other approaches, SALT is capable of extracting locations with a precision up to the street level. Performance evaluations further show high applicability for practice. Second, we present three applications for SALT: Webnear.me, Local Browsing and Local Facebook. Webnear.me offers location-aware Web surfing through a mobile Web site and a smartphone app. Local Browsing adds the feature to browse by nearby tags, extracted from Web sites delivered by SALT. Local Facebook extends location tagging to social networks, allowing to run SALT on one's own and one's friends' timeline. Finally, we evaluate SALT for technology acceptance of Webnear.me through a formative user study. Through real user data, collected during a 3 months pilot field deployment of Webnear.me, we assess whether SALT is a proper instance of "location of a Web site".

Keywords LBS · Location extraction · Location-aware Web · Location tagging · Location-based recommendation

1 Introduction

Mobile phones and mobile Internet are rapidly gaining importance. There are currently more people having a mobile phone than a computer, which shows the increasing importance of mobile applications and services. Compared to desktop applications, mobile applications have the potential to provide features which cannot, or only hardly be offered on a desktop computer, for instance, delivering information depending on the user's current location. Location-based services (LBS) are expected to attain a user base of 1.4 billion by 2014 [2].

LBS eventually deliver certain content to the user. Let us consider two aspects of content: The first aspect is the way the content "gets linked" to a location; whether this is done in an unsupervised way or whether users have to assign a location to the content manually. We provide a classification of existing LBS in these two aspects in Table 1. The second aspect is the type of content the LBS links to: While LBS usually create content from scratch, we believe that one of the most suitable types of content is the existing World Wide Web (WWW). Web sites are likely to constitute the largest source of information in the Internet; in 2008, Google announced that their number of indexed unique URLs exceeded one trillion.[1] Using WWW

This paper is an extended and revised version of [16].

B. Hess (✉) · F. Magagna · J. Sutanto
ETH Zürich, Weinbergstrasse 56/58, 8092 Zürich, Switzerland
e-mail: bhess@ethz.ch

F. Magagna
e-mail: fmagagna@ethz.ch

J. Sutanto
e-mail: jsutanto@ethz.ch

[1] http://googleblog.blogspot.ch/2008/07/we-knew-web-was-big.html.

Table 1 Summary of the comparison between Webnear.me and Yahoo Placemaker

		Location tagging	
		Manual	Unsupervised
Linked content	Not Web site	Classical LBS (e.g., AroundMe), Flickr, Plazes, Yelp	Flickr (via EXIF)
	Web site	Google places, Facebook pages, Yellow Pages	Geosearch engines, (Our approach)

as a data source for LBS, however, requires Web pages to be location-aware. In other words, we have to assign a location, based on textual features of a Web page. This requires to process the text and to find the location to which the text relates. It is worthwhile to note that not all Web pages have a location whereas some have several. For example, the location of a restaurant would be the location of the physical address on its Web site. Currently most applications require the Web page owner or the shop owner to manually enter his address to the system. In another example, a blog-Web page that contains the sentence "I have been to Paris" should be tagged with Paris as the location. Whether "Paris Hilton" refers to a Hotel in Paris or to a personage seems less trivial. If it in fact refers to a hotel, the question remains if in Paris/France or in Paris/Texas/USA. These questions have to be addressed in a disambiguation process.

Considering the vast size of the WWW, and to avoid that Web page owners have to manually tag their location, a promising option is to tag Web sites in an unsupervised manner. A resulting location layer above the classical WWW would permit numerous novel applications. As it is for most users easier to enter a URL than an IP address, it will be easier to provide a location than memorizing the corresponding URLs. And while traditional search engines take as input a written search query, a search engine utilizing the location layer will request a location as input. A list of URLs will eventually be the output of both search engine types.

In this paper, we aim at addressing numerous research challenges en route toward the "location-aware Web". Questions that we address are as follows: How to extract locations from the Web in an unsupervised manner? Can this be done with sufficient precision? Is this computationally efficient? What are potential applications and are they accepted by users? Do the applications provide an appropriate definition of "location of Web content"?

We approach these questions by first presenting an unsupervised location-tagging engine called SALT, which extracts locations using textual analysis. After automatic extraction, SALT is still capable of improving the results using user feedback. We will also show that SALT delivers improved precision compared to other open systems. With performance evaluations, we demonstrate the feasibility to run SALT efficiently on standard hardware.

To demonstrate the potential of SALT, we implement three novel applications: Webnear.me is an application of SALT for location-based search engines. It is implemented as a mobile Web browser, augmented with two search functions: "near.me" and "near.this". The former allows to search Web sites by the user's current location and the latter to search by the location assigned to another Web site. The second application, Local Browsing, requests another input type: Instead of providing a location, the user can browse by nearby keywords. Local Browsing delivers a list of URLs depending on the locations of the selected keywords. The third application, Local Facebook, addresses a type of Internet content that is today faster growing than Web sites: socially generated content. Local Facebook makes use of SALT to index the wall of one's own and one's friends' wall on Facebook. This enables to cluster posts by locations they belong to.

SALT eventually creates a database that provides a mapping from geographic coordinates to Web sites, a service not provided by the current WWW infrastructure. Related research presented in Sect. 2 usually lacks to include user aspects in their study. But it is ultimately up to the user to decide if he accepts a location-aware Web, if he considers it as useful and easy to use. We address this question through a formative user study that included participants to complete tasks with Webnear.me and other search engines. Afterward, they were surveyed on perceived usefulness and ease of use of Webnear.me for several search tasks. Related to user acceptance is the requirement of locations to be correctly assigned to Web sites. The underlying question is "What is the location of a Web site?". SALT implies that the location can be derived from textual information (e.g., addresses, city names). Similar approaches make this assumption without further verification. We attempt to approach this assumption during three- month pilot field deployment of Webnear.me and Local Browsing. There we observe user patterns, especially the distance of their locations to the location extracted by SALT.

The following sections are structured as follows: In Sect. 2, we give an overview of related work. In Sect. 3, we introduce our location-tagging engine SALT and compare its performance with Yahoo Placemaker, the only publicly available tagging engine. Further we evaluate computation time and memory requirements of SALT. In Sect. 4, we present the applications Webnear.me, Local Browsing and

Local Facebook, followed by a formative user acceptance study in Sect. 5. In Sect. 6, we present the results from a three-month field deployment of Webnear.me. We conclude the paper in Sect. 7 with a discussion and potential for further work.

2 Related work

Researchers define LBS as "services accessible with mobile devices through the mobile network and utilizing the ability to make use of the location of the terminals" [26]. LBS typically consist of five components: mobile device, service and/or application, positioning, communication, and the content [8]. In our paper, we focus on applications and the content for a location-aware Web, whereas we presume the availability of mobile devices (i.e., smartphones) with positioning (e.g., GPS, WLAN) and communication (e.g., UMTS, LTE) abilities. Example LBS that use Web sites as content are the Yellow Pages, Google Places and Facebook Places. They have in common that Web pages have to be manually added to the system. There are also approaches whereby the Web page owners can include the location directly into the HTML-source: Opera Standards for geocoding [1] describe a possibility to extract geographic information from a Web page by reading predefined HTML tags added by the author of the Web page.

For automatically building a location layer over the WWW, the requirement is a system that extracts locations from Web sites. Some literature suggests to assign a location to a Web site by locating the technical infrastructure of a Web site [13]; the geographic location is then assigned via (1) Server IP address lookup or (2) DNS registry. Although this enables in some cases a rough estimate, it is not certain that this location correlates with the context at all. A user-centric approach is presented in [10], where Web content is tagged depending on the locations where users accessed it. We follow an approach called geoparsing, which constitutes an active area of research [17–19]. Geoparsing introduces the concept of extracting location features from unstructured text or special annotations [1]. The literature divides geoparsing in three parts: (1) Parsing a textual document and identifying geographic references, (2) resolving the ambiguity and (3) finding the relevant scope. A common way to identify all geographic references (geotags) is to cross-check the input with a gazetteer [27]. A gazetteer is a database that stores information about places and the connections between them. In the geotagging process, all words are looked up in the gazetteer. If the gazetteer contains the word, it gets marked as geotag. There are also ideas to use Wikipedia as Gazetteer [12, 21]. Semantic networks [14] can be helpful to find implicit geographic references like "Eiffel Tower", which refers to Paris. Ambiguities are usually split in cases where geographic names are confused with non-geographic ones (geo/non-geoambiguities) and in cases where two geographic names are confused (geo/geoambiguities) [6]. The position in a text, the occurrence and the population of a found geotag can be used to influence the disambiguation process [5]. Advanced algorithms use tools from natural language processing (POS, NER) to find locations and resolve ambiguities [15, 24]. The text is analyzed and divided into phrases, and the nouns are more accurately determined with the aid of the syntax. [6, 7, 20] discuss an algorithm that enables to assign a location even though it is never literally mentioned in the context (e.g., if there are some occurrences of "Texas" and some of "Washington DC", the scope of a page is "USA"). Our approach achieves this during the scope scoring process.

Another stream of research considers the question how user feedback or, in our case, user locations can be utilized to refine (local) search. Location of users during mobile search is investigated in [23]. The result is that 40 % of the searches concern a close location. Other frequent searches are in transit and concern the destination, or the search depends on time. Such user behavior patterns are often used for optimizing information retrieval (e.g., [3, 22, 25]).

Besides the ongoing research, there are a variety of commercial products which offer text extraction, such as MetaCarta, Digital Reasoning (GeoLocator) and SRA (NetOwl). These applications are able to extract places along with other entities (persons, time, organizations, time or money). Moreover, publicly available systems such as Yahoo! Placemaker offer a geoparsing Web service that finds places in unstructured content like feeds, Web pages, news and status updates. Placemaker identifies places and disambiguates them to a location. However, compared to our system, Placemaker does not deliver a precision up to street level. Placemaker has for example been used in [7] for location-based advertising or for a Google Chrome extension that analyzes a Web site regarding its geographic information. Geodoc is a semi-unsupervised system as it requires the user to tag names manually and then automatically extracts geographic information.

We challenge the state-of-the-art by showing the feasibility to efficiently and accurately geoparse Web pages to street-level precision using a worldwide gazetteer database (with up to 16 million places). Other research is either conceptual in nature [18, 19], shows a proof-of-concept for recognizing 150,000 places [18] or a single country only ([17], Germany). Further, we strive to extend the pure-algorithmic aspects usually considered in literature. Especially, we include mobile and desktop applications evaluated with real users in order give a holistic view of the location-aware Web, covering the extraction method, applications and user evaluations.

3 Geoparsing with SALT

In this section, we present our approach to unsupervised geoparsing: SALT. In Sect. 3.1 we describe the process to extract geographic locations from Web pages. In Sect. 3.2 we proceed with analyzing the complexity of SALT, and evaluate SALT regarding its performance (Sect. 3.3) and precision (Sect. 3.4).

3.1 SALT process

Our geoparsing method SALT consists of three main parts: parsing, geo-resolution and disambiguation. These parts are followed by a final location assignment. A user feedback functions to correct possibly wrongly tagged content and influence the final assignment. In the following, we describe all parts:

3.1.1 Parsing

In the first part, we parse the Web page and extract all words. The parser works as follows: We grab the source of a page and extract the content information using regular expressions (removing HTML-, JavaScript- and XML-tags). Alternatively, a common domain object model (DOM) parser can be used to extract the contents of the respective HTML tags. The output from our parser is a table containing all words from the title, content and metadata of a Web site. Summarized, we obtain a list of words: $W = w_1,\ldots,w_n$.

3.1.2 Geo-resolution

The second part comprises resolving all geographic names by looking up the parsed words in our gazetteer. We use two existing open-source gazetteer databases: Geonames and openStreetMap. Based on them, we build our own gazetteer by combining and bringing them in one common form. This contains names (one or multiple words) and parent information (containing country, continent or region), along with their geographic location. The next task is to recognize names (multiple, subsequent words). In a naive approach, one creates a table containing all names, split into separate words. While iterating through the words in sequential order, one looks for names starting with the current word and checks the rest of the name in the next step. We implement a more efficient approach using an efficient string matching algorithm (Aho Corasick) to look up all geographic entries simultaneously. After extracting all names from the content, we start filtering. Since many very common stop words are also associated with a location (like "life", a place in Tennessee, USA), we ignore all single-word names that are stop words. With this filtering we can remove many useless results without significantly influencing the outcome. Summarized, geo-resolution resolves W to a list of geographic names with associate geographic coordinates $N = n_1,\ldots,n_m$, $n_i = (name, latitude, longitude)$.

3.1.3 Disambiguation

The aim of the disambiguation part is to resolve the ambiguities in the list of geographic names N. Two common ambiguities are geo/non-geo and geo/geo. A geo/non-geoambiguity occurs when a common word is also the name of a location: e.g. "Paris Hilton" could refer to a hotel in Paris or to a person. A geo/geoambiguity occurs when two or more locations share the same name, for example "London" points to the well-known capital of the UK, but also to many other smaller cities all over the world.

We perform disambiguation sequentially: Firstly, geo/non-geoambiguous words receive a low value $p \in [0,1]$. Secondly, geo/geoambiguities are resolved with a process we name *scope scoring*, taking the values p as input for each word.

- Regarding geo/non-geoambiguities, we notice that many common words and expressions are stored in the gazetteers (e.g. "The City" is an alternative name for London, UK, but is not unique, since it could refer to any city). Therefore, we maintain a database with such expressions and flag all stop words, words from a common dictionary, and common expressions. Another issue arises from the fact that some names simply cannot be uniquely linked to a location. Examples for those words are "residence", "district", "center" and "city". Those words may be found in gazetteers. They usually have a geographic meaning, but without further context they are ambiguous. To address this problem we add the possibility for a user to provide feedback. The user can tag words or expressions. If a sufficient number of users tag a word, we consider it as stop word and remove it from the gazetteer. To resolve some of the remaining ambiguities we also consider the number of words a name consists of. The more subsequent words are resolved as a name with geographic meaning, the more likely it is that the name is valid and the less ambiguities occur. As result of the process so far, we assign initial values p to all words. The p values are afterward utilized for geo/geodisambiguation. As rules of thumb, we define the following values: $p = 0$ for stop words, values $p = 0.7$ for single-word names and $p = 0.8$ for all names with more than one word. From the p value, we subtract 0.2 if the word is contained in a common dictionary.

- Geo/geoambiguities are resolved by determining the main geographic scope of the document. A scope can be a common city, state, or country. If, for example, the input contains some mentions of "Dallas", "Houston", and "Paris", the scope would be Texas, USA. The possibility that "Paris" was a reference to Paris, France is comparably low. For determining the scope, we define a process that we call *scope scoring* (see the example in Fig. 1): All geotags receive an initial value p, determined by the geo/non-geodisambiguation step. Their initial score is 0. We continue by creating a tree consisting of the geotags' parents. Parents are represented as ontology in the gazetteer. An example parental path ranges from address to city, zip code, county, state, country and continent. The parents of Paris are "Texas" and "USA" in one scope, and "Île-de-France" and "France" in the other. As simplified example let us assume all initial p values are 0.7. This value is then propagated up in the tree with a damping factor $d \in [0,1]$. As rule of thumb we take $d = p$. For Paris, the value $p \cdot d = 0.49$ is added to "Texas", and the value $p \cdot d \cdot d = 0.343$ is added to "USA". This procedure is repeated for Paris/Île-de-France/France, as well as for all other geotags. We continue by computing the average values of all paths in the tree. In our example the path Paris/Texas/USA has a higher average than Paris/Île-de-France/France, and we are subsequently able to discard the ambiguous Paris, France. The scope of Paris is in this case Paris/Texas/USA.

In summary, the entire disambiguation process selects a subset of the geographic names, associated with the scope and a score: $N_{\text{dis}} \subseteq N, n_i = (\text{name}, \text{latitude}, \text{longitude}, \text{scope}, \text{score})$.

3.1.4 Final location assignment

At this stage, we have the following available information: geographic names, their geographic coordinates, their scope and a scope score. For the final assignment, we rank all names by their scope score and number of occurrences. The top ranked name (names in case of equal top scores) is the suggested output by SALT, associated with the URL of the Web page. In the example from Fig. 1, these are Paris, Dallas and Houston (in Texas, USA). We further foresee a feedback function that enables users to select the an alternative name from the ranking as the more appropriate one. The feedback can be added to the scores. We note that the feedback process might be prone to users deliberately giving "wrong" feedback. Therefore, we discard multiple feedbacks from the same user.

3.2 Complexity

To estimate the complexity of SALT, we further discuss each part of the method in detail. The first part essentially comprises parsing the DOM of a HTML page. Modern Web browsers prove be able to perform this "on-the-fly".

The challenge of the geo-resolution are frequent lookups in large gazetteers (with up to 16 million entries). A naive lookup method would query each input word, plus potential pre- and suffix words. Even if the gazetteer is indexed (e.g., B-Tree index) and allows $O(log(n))$ search, the complete geo-resolution will have more-than-linear complexity. As a more efficient alternative, we implemented the string matching algorithm by Aho and Corasick (AC) [4]. AC stores the gazetteer names as patterns in a graph. Given a Web page with l words, all patterns can be matched simultaneously with complexity $O(l + m)$, where m is the number of matched patterns. Note that the graph structure of AC should reside in main memory to avoid long access time during graph traversing.

The disambiguation part is dominated by lookups of the candidate names in common dictionaries (with about 2 million entries). Given the relatively small number of candidate names, we consider $O(log(n))$ lookups in the database to be sufficient. Finally, scope scoring requires

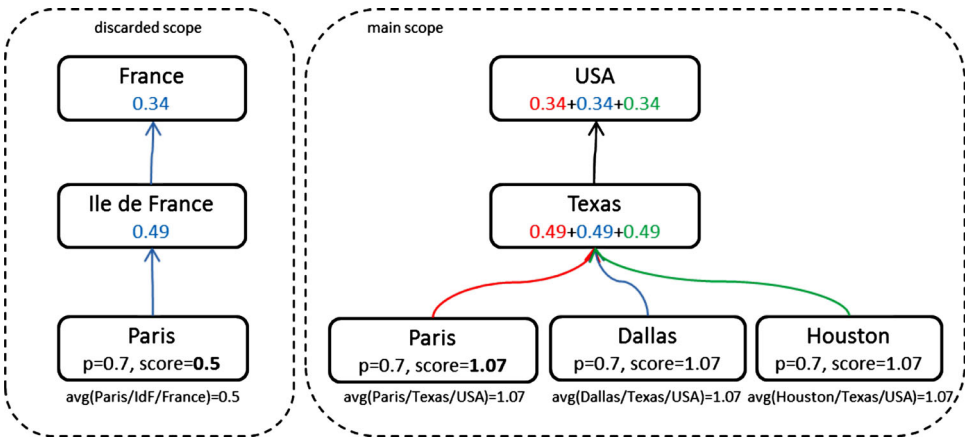

Fig. 1 Scope scoring process

construction of graphs. Usually, these graphs have a depth of about 4 (number of geographic parents). After stop word filtering, given the small number of candidates, scoring is dominated by dictionary lookups with $O(log(n))$ search complexity. Once the Web pages are indexed by SALT, queries will usually request the k nearest pages to a given location. The underlying K-nearest-neighbor problem can be solved in $O(k + log(n))$ time. Modern DBMS (e.g. PostGIS) support this query type. LBS rely on this query type and have shown that it can be efficiently performed in practice.

3.3 Performance evaluation

Considering the large number of Web pages to process and the large number of entries in gazetteers, execution time and memory usage are crucial factors for practical applicability. We evaluated execution time of SALT split by the three main steps: parsing, geo-resolution and disambiguation. The evaluations were done on standard hardware with a quad-core 3.4 GHz Intel i7-3770 CPU and 32 GB of main memory. The first evaluations measured execution time while varying the number of words of the input Web page. As input we selected a pool of pages (news articles and yellow page entries) and randomly combined them to the word count desired for evaluations. Figure 2 depicts the results for both the naive and the Aho Corasick-based method for geo-resolution. With the naive method, the total execution is dominated by the geo-resolution and shows a more-than-linear growth. The inferior performance with naive geo-resolution can be explained with expensive database lookups in the gazetteer for any word, plus pre- and suffixes.

The situation improves significantly when using Aho Corasick for matching the input with gazetteer words. The total execution time is now dominated by the linearly growing disambiguation process. An already rather complex Web page with 5,000 words is processed in under one second and a page with 15,000 words is processed in under three seconds. Most of this execution time is spent for disambiguation during I/O of database lookups. Execution time could be further diminished by using an in-memory database for this purpose.

The linear-complexity Aho Corasick method for geo-resolution comes with a trade-off in memory usage: The gazetteer database of Geonames contains 16 million entries. A condensed gazetteer (Europe and US only) contains approximately 5 million entries. Aho Corasick requires them to be indexed in a graph. The graph should reside in main memory to enable efficient traversing (for string matching). We depict the influence of gazetteer size to both memory usage and execution time of Aho Corasick-based geo-resolution in Fig. 3.

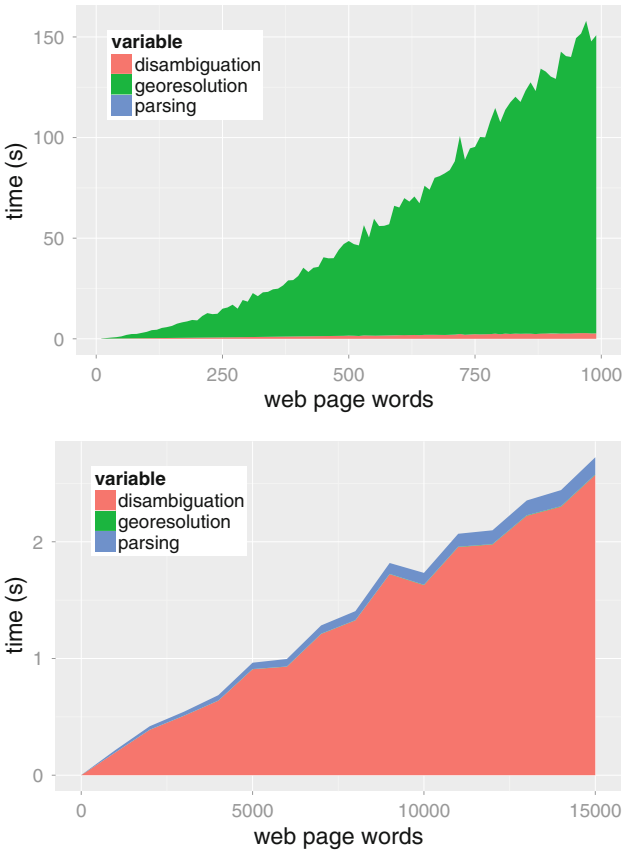

Fig. 2 Execution time with naive (*above*) and Aho Corasick-based (*below*) geo-resolution, varying Web page words. Using a condensed gazetteer of 5 million entries (*above*) and a worldwide gazetteer with 16 million entries (*below*)

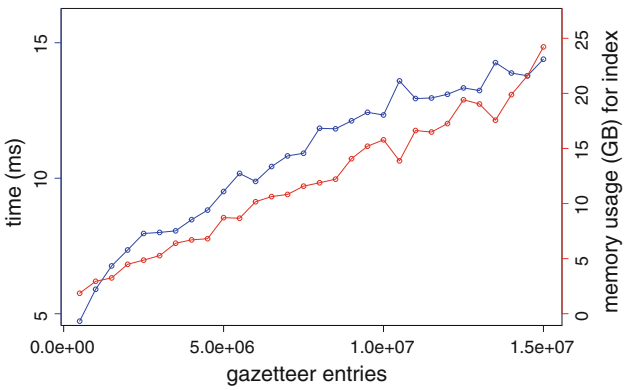

Fig. 3 Execution time and index memory usage of Aho Corasick-based geo-resolution, varying gazetteer entries. Execution time is measured using a 50,000 word input

Notice that even the full gazetteer fits in about 30 GB of (main) memory, a reasonable size for standard modern servers. Even with a page input size of 50,000 words, geo-resolution is performed in a few milliseconds. We further note that for larger gazetteers, the index could be easily

distributed among multiple servers using a map-reduce based approach for lookup.

3.4 Precision: SALT compared to Yahoo Placemaker

We compared SALT with the only publicly available location tagger: the Yahoo Placemaker. One of the differences between SALT and Placemaker is that SALT provides street-level accuracy while Yahoo Placemaker only at the city level. In our experiment, we compared the different recognition rates as follows: We counted how many times the system could not recognize a location, how many times it is right and how many times it is wrong. In this comparison, we used the SALT system without the feedback feature.

The experiment setting was as follows. We crawled the Swiss Yellow Pages for entries with a domain, and then randomly selected 50 items from the Zurich area. Then two researchers manually assessed the items independently and assigned locations if available. Afterward we randomly chose 50 entries. Table 2 shows that both systems have similar wrong recognition rates. However, SALT more often recognizes a location and generally achieves a higher recognition rate.

4 Location-aware Web applications

To demonstrate the applicability of SALT to real-world applications, we developed three applications making use of online content and location extraction with SALT. These applications address two types of content: *Web sites* and *social media*. Web sites are hereby representing content that has been created since the launch of the WWW, bearing an enormous potential for re-use in LBS. Content from social media is recently massively increasing. Although social media platforms like Facebook offer features for explicitly tagging user entries with location, reliable algorithmic tagging methods are currently missing. For instance, functions that filter posts of friends by a location are not available.

We first describe a proposal for integrating the location layer created by SALT in the existing Web infrastructure. Then we present our applications: *Webnear.me*, *Local Browsing* and *Local Facebook*.

Table 2 Summary of the comparison between Webnear.me and Yahoo Placemaker

	Not rec.	Wrongly rec.	Correctly rec.	Precision
SALT	6	14	30	Street level
Yahoo placemaker	13	11	25	City level

4.1 Integration to the Web infrastructure

A comprehensive location-aware Web ultimately requires an integration to the Web infrastructure. In essence, SALT enables a mapping from Web pages (i.e., their URLs) to locations. Another service which maps domain names (part of the URL) is domain name system (DNS), resolving domain names to IP addresses based on a distributed hierarchical directory. DNS could be extended to also accommodate geographic coordinates besides IP addresses. An alternative to this is to establish a service parallel to DNS. As in DNS, transport could be based on TCP or the more lightweight UDP. A less intrusive integration is to provide an open Web service (e.g., RESTful), from which the locations can be obtained. Applications supporting this Web service could relatively easily integrate location-awareness for displayed Web pages. For indexing Web pages with SALT, we suggest to use search bots in a similar manner as they are used by search engines. In this sense, given the availability of a sufficient number of SALT servers, in the future, Web sites could even be indexed on-the-fly when they are requested. Using the clients for extraction seems rather unrealistic given the high memory demands of SALT.

For a proof-of-concept implementation, we choose the RESTful Web service option. The service is depicted in Fig. 4 and accommodates the following main components: (1) event handler (2) location-webpage storage and (3) SALT.

The event handler manages all requests from the different interfaces. Consider a request for the location of a given Web page: In this case the “location lookup” unit (part of the event handler) invokes the location-Web page

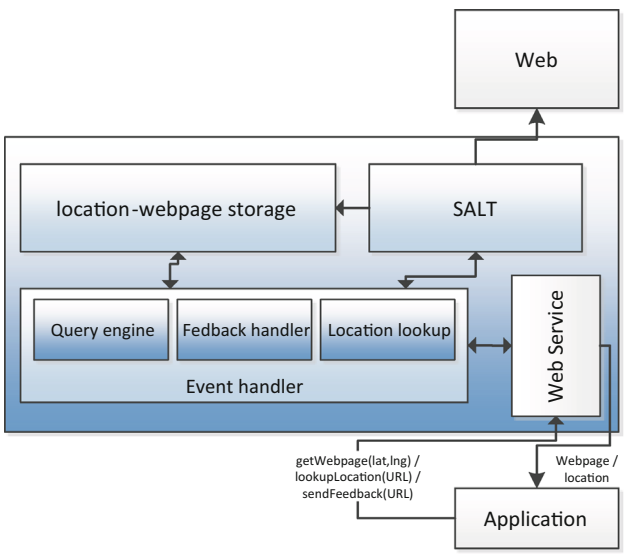

Fig. 4 Implemented architecture for the location-aware Web

storage which checks whether the requested Web page is already in the database. In this case, the Web page location is returned. Otherwise, the "location lookup" handler adds the URL to a queue, processed by SALT. SALT eventually stores the result in the location-Web page storage.

The "feedback handler" component allows applications to return feedback to the received mapping (i.e., "right", "wrong", or an alternative location). The overall feedback influences the ranking of possible locations in the location-Web page storage.

The third part of the event handler is the "query engine". This unit manages application requests to query Web pages near a specific location. The respective queries are sent to the location-Web page storage, implemented as a geo-spatial database (MongoDB in our case). Below we present the three applications in detail.

4.2 Location-aware Web browser: Webnear.Me

The first application we present is *Webnear.me* [11]. It is developed as a platform-independent HTML5 Web site, a Windows Phone 7 app, an Android app and a desktop version in the form of a Mozilla Firefox plugin. The aim of Webnear.me is to allow users navigating through the Web not only via Web links and text input but also via locations. The features are (1) displaying Web sites near the current user-location (Webnear.**me**), and (2) displaying Web pages near the currently visited Web page X (Webnear.**this**).

Visible to the user is the Firefox extension in the desktop case and the HTML5-page in the mobile case (Fig. 5). The extension is in form of an additional toolbar and consists of four buttons and a text field. The aim of the text field is to display the location of the current Web page visited by the user. The user can interact with the extension by giving feedback to the estimated location through pressing the "right", "wrong" or "let me correct" button. If the user presses "let me correct", the extension opens a Web page with a list of possible locations, from which the user can choose the right one. Moreover, the user has the possibility to discover Web pages which are geographically close to the current one. By pressing the "discover" button, the browser opens a new window which displays Web pages close the current one.

4.3 Keyword-based browsing: Local Browsing

The second application for the location-aware Web and SALT is Local Browsing. It is developed as a native Windows Phone 7 application. Local Browsing adds an additional level between the mapping from locations to Web sites: keywords. Compared to Webnear.me, which allows to display the full list of Web sites corresponding to a location, Local Browsing adds the feature to filter the Web sites by keywords.

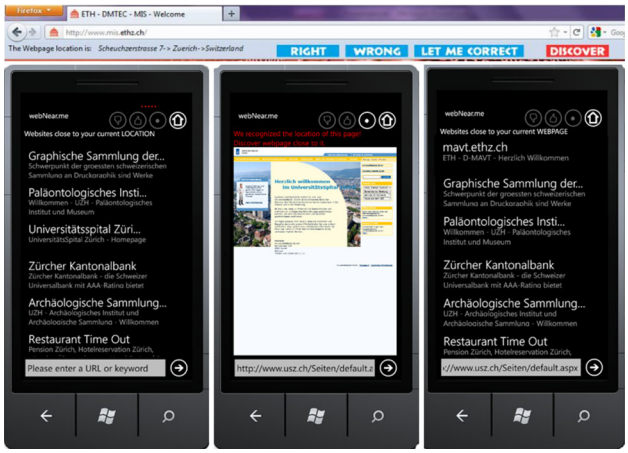

Fig. 5 Webnear.me Firefox extension (*above*) and the Windows Phone mobile client (*below*)

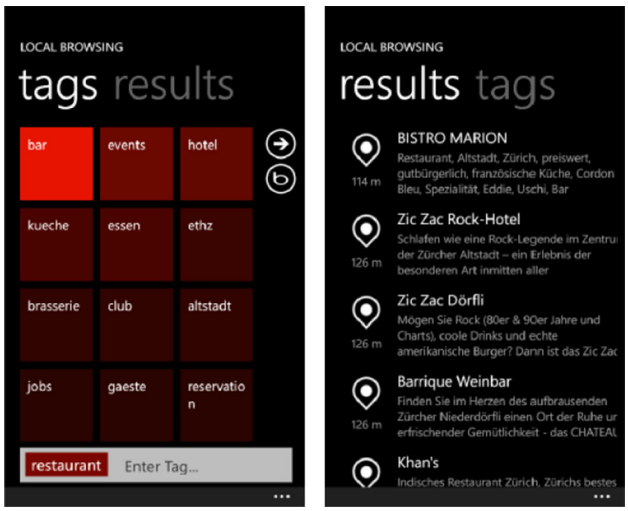

Fig. 6 Local Browsing mobile app for windows phone with tags (*left*) and results (*right*)

The basic functionality of the app is depicted in Fig. 6. When the user starts the app, he is shown a list of keywords that correspond to nearby Web sites. The keywords are ordered by their proximity to the user location and their number of hits. An example of such ordering would list the tags "bar", "events" and "hotel" first. In addition, the user can enter his own keywords. When one keyword is selected, the app displays all other tags that appear together with the keyword, allowing to further refine the Local Browsing. Once the user selected all keywords, a list of Web sites is displayed, containing those Web sites that (1) are nearby the user location and that (2) contain all selected keywords.

Local Browsing shares the same server infrastructure with Webnear.me, described in the previous section. While SALT extracts location information from Web sites, we

also extract often occurring keywords extracted from metadata and from textual content of the Web sites. These keywords are mapped to the location extracted by SALT, allowing to display a list of nearby keywords.

To sum up, Local Browsing allows for more selective discovery of nearby content, while Webnear.me is more focused on augmenting Web browsers with location-awareness.

4.4 Location-based social networks: Local Facebook

The third application shifts the focus from applying SALT to Web sites toward social networks. Social networks like Facebook and Twitter are today among the most successful Internet platforms, creating massive socially generated data, which might partially have location-based relevance. Although Facebook offers the feature to explicitly and manually tag posts with location identifiers, there is no unsupervised method in place, and own and friends' posts cannot be filtered by location. SALT is conceptually applicable for all textual content, therefore also to social networks like Facebook.

Local Facebook makes use of the official Facebook API to gather posts from one's own and one's friends' wall. On each post, we run SALT to extract location information, if available. For Local Facebook, we slightly modify the server infrastructure: instead of creating a database that maps locations to Web site URLs, we map locations to Facebook posts. The stored records include an unique identifier for the post and the timestamp of its creation. This allows to gather the original posts and to order them by a timeline as known from Facebook.

Local Facebook is an implementation example that shows the flexibility of SALT to be used beyond Web sites.

5 Formative user study

In this section, we present a formative user study that has been conducted to verify the acceptance of the SALT algorithm, applied to the Webnear.me application. This study consists of twelve participants, completing different tasks with Webnear.me. Afterward, they were requested to fill out a survey.

5.1 Experiment procedure

Four women and eight men participated in our experiment. These twelve participants used our system to fulfill different tasks. Participants were asked to imagine several situations in which they used their mobile phone in the last week. The participants were asked to do the task again as follows: (1) with the system they used originally (Web browser or search engine) and (2) with Webnear.me. We recorded the log of operations in each system and surveyed the participants after they completed all tasks. The goal of the questionnaire was to find out about the acceptance of the new technology used in Webnear.me. Our questions focused on three different scopes: (1) the feature which displays Web sites close the current position of the user (Webnear.me), (2) the feature which displays Web sites close to another one (Webnear.this), and (3) the application and the location-aware Web in general. Figure 7 shows the questions given to the participants. The participants had to answer with a score (1–7) for each question. The questions are based on the technology acceptance model (TAM) [9]. The TAM states that the perceived ease of use and the perceived usefulness determine together the user's acceptance of a new technology. In addition to TAM, we also collected free opinions from the participants.

Scope: Showing webpage near to the user current position

Q1: WebNear.me is easy to use

Q1	1(disagree)	2	3	4	5	6	7(agree)
	0%	0%	0%	8%	8%	67%	17%

Q2: WebNear.me helps me in my daily mobile web behavior

Q2	1(disagree)	2	3	4	5	6	7(agree)
	0%	0%	0%	8%	59%	25%	8%

Scope: Showing webpage near to a specific webpage (WebNear.this)

Q3: WebNear.this is easy to use

Q3	1(disagree)	2	3	4	5	6	7(agree)
	0%	0%	33%	50%	17%	0%	0%

Q4: WebNear.this helps me in my daily mobile web behavior

Q4	1(disagree)	2	3	4	5	6	7(agree)
	0%	0%	0%	0%	8%	75%	17%

Scope: General

Q5: The application "WebNear.me" add value to the mobile web

Q5	1(disagree)	2	3	4	5	6	7(agree)
	0%	0%	0%	8%	0%	67%	25%

Fig. 7 Questionnaire results

5.2 Formative user study results

The results show that all participants noticed an added value of the location-aware Web and the two features of Webnear.me (Web sites close to the user and close to a specific site). The participants perceive a higher usefulness for the Webnear.this (avg: 6.08) than for the Webnear.me feature (avg: 5.33). However, the participants perceived the Webnear.this feature as slightly harder to use and understand. In particular, the participants told that it was not so easy to understand what the feature exactly does. After explaining the aim and showing for what the feature is good for, we received a much better feedback. Six participants told us that they would like the feature especially for discovering Web pages while they are waiting. For the Webnear.me feature the participants told us that they especially see an added value when they want to have fast access to the homepage which is close to them. As example one participants told that the application is great when he is looking for a telephone number of somebody in the current building. Finally, eight participants told that Webnear.me is more entertaining than a search engine or classical Web browsing.

6 Field evaluation

In a pilot field deployment, we are especially interested in evaluating user access to online content tagged with SALT. We aim at answering the question whether our applications provide an appropriate definition of "location of Web content". In the case of Webnear.me and Local Browsing, the question refines to "What is the location of a Web site?". In case of Local Facebook, the question refines to "What is the location of socially generated content?". We remind that SALT bears the underlying hypothesis that these locations can be derived from textual information. One potential issue is geographic ambiguity and how SALT is able to deal with it.

A second issue is less connected with algorithmic inaccuracies but more with recommender issues: Applications like Webnear.me that list nearby content in a list view should list the entries by relevance to the user. In the simplest case, one could order the results by proximity of the user location (p_{user}) to the location extracted by SALT (p_{SALT}). However, this leaves out other parameters like the number of hits to an item, or what other users near this item were accessing. Research [10] has proposed to consider the following user access patterns: *local*, *multi-local*, *transit* and *random*. We take them into account and propose the following evaluation components:

- Location of item i, extracted by SALT: p_i^{SALT}
- Location of user: p_{user}
- Number of hits for item i: h_i
- Set of locations where item i was accessed in the past: $P_i^{\text{ACCESS.}}$

Metrics Following the above components, we further define the metrics that can be measured during a field deployment:

- Aggregate distance between p_i^{SALT} and P_i^{ACCESS} (average, minimum, maximum and distribution).
- Distribution and "locality" (pattern) of accesses to individual Web sites
- Distribution and "locality" (pattern) of accesses from individual users.

6.1 Field deployment procedure

The applications Webnear.me and Local Browsing were given to 25 participants for a pilot field deployment during three-month. Since all study participants are located in or near Zurich, Switzerland, we emphasized tagging of Web sites in Zurich. In total, there are 4,500 entries in the database indexed by SALT. Out of these entries, 200 are Web sites tagged in Pittsburgh, PA, USA. This was the location of the Ubicomp 2012 conference, where Webnear.me was an official demonstrator [11]. A specialty of the version demonstrated and used at Ubicomp was the availability of both a list and a map view for browsing nearby Web sites. During the conference, visitors could download and use the application on their own Android smartphones, or use one of our demonstration smartphones (Android and Windows Phone).

All participants agreed to be subject to an experiment, since all actions were logged during the test period. Logged parameters are: (anonymized) user id, timestamp, performed action in the app and user location. Over a three-month test period, a total of 486 usable log entries were collected. About 1,000 entries were discarded for the evaluation, since they were either accesses to non-location-based Web sites, or the entries did not contain valid user locations.

6.2 Field deployment results

During the deployment period, we collected about 1,500 user actions (out of which 486 are complete and fully usable entries). Figure 8 shows density maps of tagged Web sites and user accesses in the city center of Zurich.

The first observation is that, indeed, Web sites are more evenly distributed than user accesses. Considering access times, we suppose that most accesses originate from work, study and home location of the participants. Other spots were a public transport stations and shopping areas.

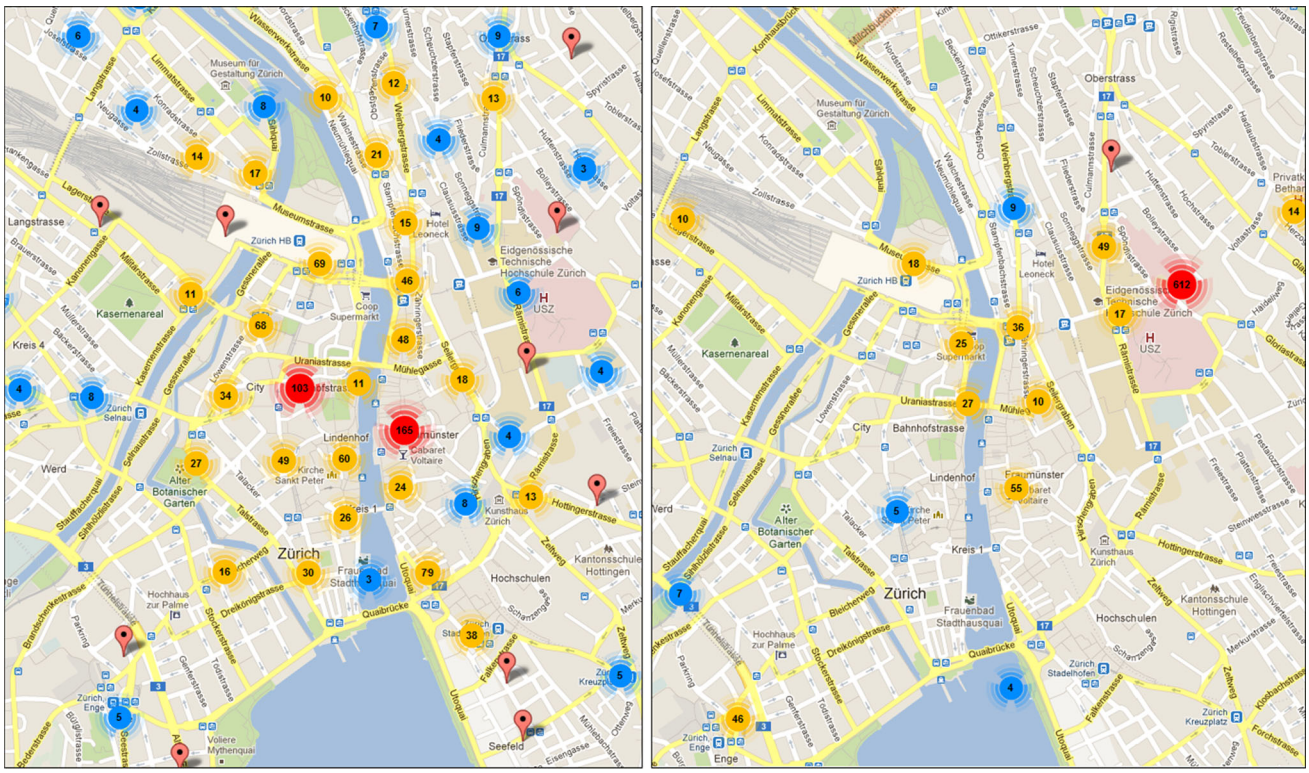

Fig. 8 Locations of indexed Web sites (*left*) and location of users that access the Web sites (*right*)

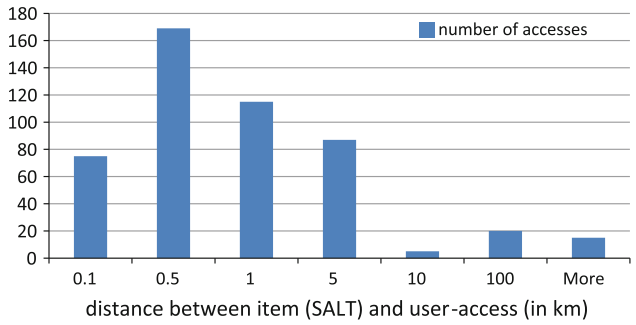

Fig. 9 Histogram showing distances between Web site locations (p^{SALT}) and user access locations (p^{ACCESS})

Distance between Web sites and user access Fig. 9 depicts a histogram of distances between Web sites tagged with SALT and the locations of users accessing them.The number of accesses generally decreases with increasing distance since the items are ordered by distance. However, we observe that Web sites further away (500m+) even showed higher popularity (despite availability of nearer ones). This can be explained with that users are often already familiar with Web sites at their current location (at work, at home). An application like Webnear.me is more useful to discover new Web sites, which can be further away from the current location. The data collected in Pittsburgh, where Web sites were selected from a map, eliminate a bias induced by ordering of the list. We observe that although nearby Web sites still have the highest access density, Web sites further away are in proportion more often accessed. Compared to the study participants in Zurich, many conference attendees in Pittsburgh were not city locals. We would have therefore expected that they would show higher access rates for content in very close proximity. This indeed happened in the case of the list view, however, not in the case of the map view.

Pattern of accesses to individual Web sites For illustrating the impact of the type of Web site on user locations, we selected two Web sites with high hit rates: the Web site of Coop, a major Swiss retail chain with several locations in a city and the fast food chain Burger King with two location in the Zurich inner city. Maps with the Web site locations and the user access locations are depicted in Fig. 10.

We note that Coop has a dense network of branches. In cities like Zurich, the distance between branches is about 1 km. If we consider the access to the Coop Web sites, we observe that most of the user accesses lie in a distance of not more than 500 m. While in Zurich this could be accounted for the dense network of branches, this distance also holds for smaller cities with less dense networks. According to the terminology introduced in [10], this constitutes a multi-local pattern.

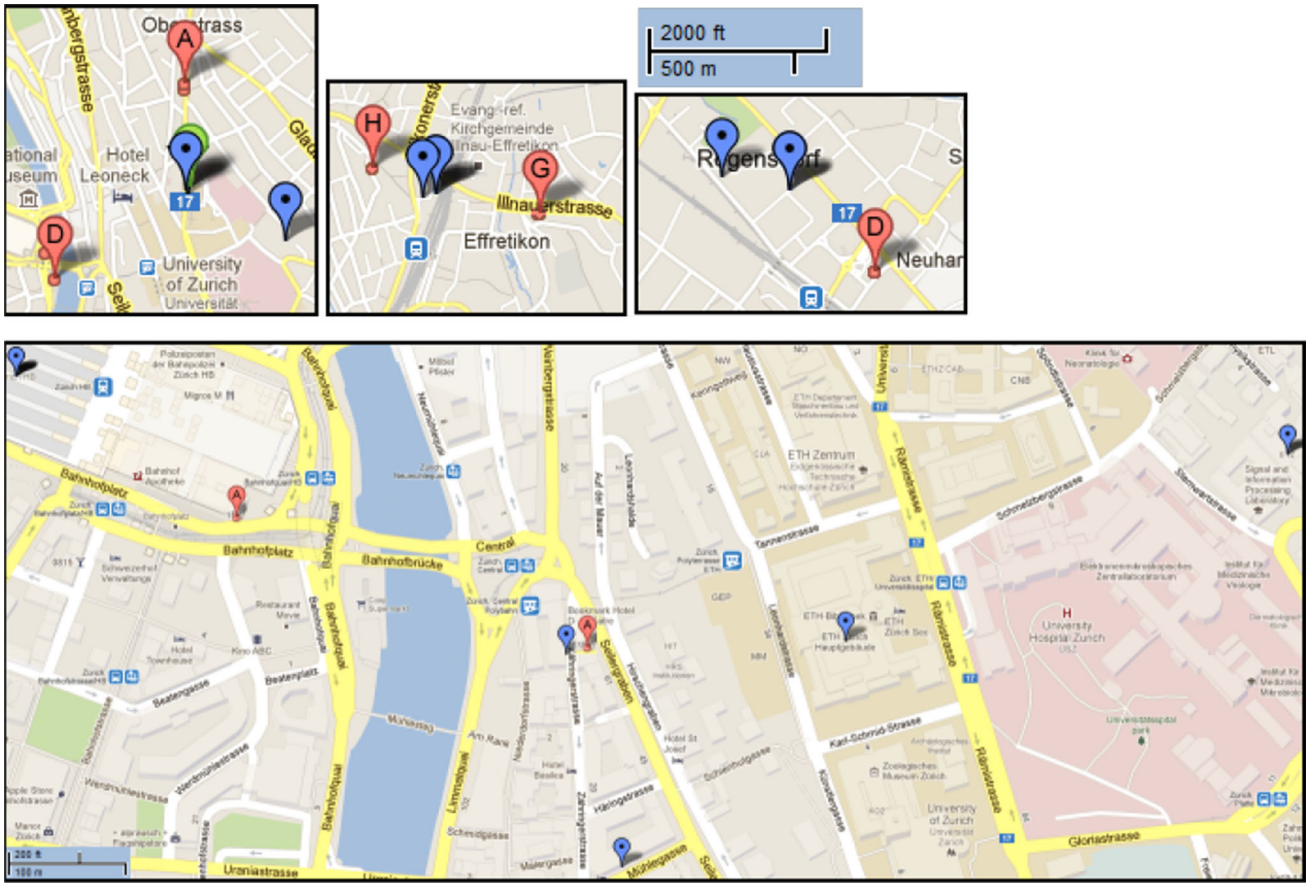

Fig. 10 Web site locations and user accesses: coop (retailer, *above*), Burger King (*below*). *Red pins mark* Web site locations, *blue pins mark* user locations (color figure online)

A different pattern may be observed when looking at the pattern from the fast food chain Burger King. In Zurich center, there are two Burger King branches. The access locations in this case more distributed. Besides home and work locations which represented most part of the accesses to Coop originated, the accesses to Burger King also originated from mobile locations and from train and tram stations. The Burger King Web site was in those cases not among the first listed in the Webnear.me and Local Browsing application. A combination of the location extracted by SALT with other components like the number of hits would in this case make sense to boost the Web site's ranking. While Burger King and Coop are both well-known brands, we observed that Web sites of smaller businesses only received hits from relatively close locations. In this case, an ordering should give more weight on the proximity (SALT) than the number of hits. A possible solution to ensure that small businesses with a low total number of hits but a relative high number of local hits are not ranked too low is the following: instead of weighting proximity to a Web site with its total number of hits, the proximity should be weighted with the number of hits that originate from close proximity.

Pattern of accesses from individual users The users during the three-month test period showed repetitive patterns of their own location, which confirms what is suggested by related research. Another tendency we observed in the later test period was that Web sites further away were accessed more frequently. This can be explained with the purpose of Webnear.me to explore unfamiliar Web sites nearby. Compared to social media, Web sites do not change very often and the number of Web sites is nearly static. If no new Web sites are created, users will likely extend their search for more distant Web sites. This factor should be considered while ranking results obtained by SALT.

7 Conclusion and future work

In this paper, we presented a new approach on location-based services: the location-aware Web. The idea is to use location information available on Web sites to make them usable for LBS. For this purpose, we developed SALT, a location extraction engine. SALT has the advantage that, compared to all other available location extractions systems, it can handle feedbacks and extract locations with a

precision up to the street level. As our experiment showed, SALT has a better recognition rate as compared to a similar approach, Yahoo Placemaker. This holds even without using the feedback feature of SALT. Regarding practical applicability of SALT, our performance evaluations showed the feasibility to run SALT in under one second for already rather complex Web content.

Moreover, we presented three applications for SALT. Two of them are use cases for the location-aware Web while the third shows the applicability of SALT for social networks. Webnear.me is a new type of augmented Web browser using the location-aware Web technology. Webnear.me introduces a new dimension of searching the Web, whereby users can not only navigate via links or text inputs but also navigate via a location. The location for navigation can be from the user's current location or from another Web page. This new feature is interesting for people discovering the Internet without precise target. Our user acceptance study confirmed the usefulness of the location-awareness feature. Moreover, users enjoy using the Webnear.this function. According to the study participants, it gives mobile information seeking an entertainment factor. The second application, Local Browsing, replaces the idea of an extended Web browser with added functionality that allows filtering Web sites by keywords. The third application, Local Facebook, shows the flexibility of SALT to be also used for social media content. This adds the functionality to Facebook to filter one's own and one's friends' wall by locations. Besides these three applications, there are also numerous other applications possible. These can, for example, employ new location filtering methods like Local Browsing does. Or they could apply SALT to other types of Internet content with location-based relevance, like local questions or local Twitter tweets.

Using a field deployment, we investigated the question whether SALT delivers appropriate locations for users that use our applications. We observed that the user proximity to the SALT location is indeed an important factor for displaying nearby Web sites. However, there are other factors that influence the importance of a specific Web site for a user's location. Our results give indications that the type of Web site (e.g., restaurant, retailer) and the location of the user (e.g., home, work, waiting at a location, mobile) are important factors that should be considered for delivering relevant local Web sites. This affects not only the location-aware Web but rather all kinds of LBS. Future research could thus focus on finding optimal weightings for information retrieval. This could include components like the user proximity to the SALT location, the total number of item hits and the relative number of item hits at a location. Another future direction is the development of sophisticated filters that allow to conveniently accessing the desired Web sites. Local Browsing proposes one filter method that allows users to filter Web sites by keywords that have a local relevance.

Further future research could improve the system in two ways. First, as we rely on user feedback to identify wrongly assigned locations, we may encounter some users who deliberately give wrong feedback to harm the overall system performance. Future research could improve this method by figuring out how to identify such users and discard their feedbacks. Second, we use directories for local Web sites, and tag Web sites that users visited on-the-fly if they are not found in our database. Future research should aim to crawl the Web sites with search bots similar to those used in search engines. Regarding the pilot field deployment, we highlight that larger scale field evaluations should be done in future research for more generalized statements on user behavior.

Acknowledgments The authors would like to thank Sebastian Wendland, Samuel Zihlmann and Dominik Bucher for the design and implementation of the Local Browsing application, as well as Thomas Bürli and Benjamin de Capitani for implementing a prototype of SALT. Further the authors thank Mihai Grigore for his help in proofreading the manuscript.

References

1. Opera standards for geocoding (2009) Location-based publishing and services. http://dev.opera.com/articles/view/location-based-publishing-and-services/
2. Gartner identifies 10 consumer mobile applications to watch in 2012 (2010). http://www.gartner.com/it/page.jsp?id=1544815
3. Agichtein E, Brill E, Dumais S (2006) Improving web search ranking by incorporating user behavior information. In: proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '06, pp 19–26 ACM, New York, NY, USA. doi:10.1145/1148170.1148177
4. Aho AV, Corasick MJ (1975) Efficient string matching: an aid to bibliographic search. Commun ACM 18(6):333–340
5. Odon de Alencar R, Davis Jr C, Gonçalves M (2010) Geographical classification of documents using evidence from wikipedia. In: proceedings of the 6th Workshop on geographic information retrieval, p 12. ACM
6. Amitay E, Har'El N, Sivan R, Soffer A (2004) Web-a-where: geotagging web content. In: proceedings of the 27th annual international ACM SIGIR conference onresearch and development in information retrieval, pp 273–280. ACM
7. Anastácio I, Martins B, Calado P (2010) Using the geographic scopes of web documents for contextual advertising. In: proceedings of the 6th workshop on geographic information retrieval, p 18. ACM
8. Dao D, Rizos C, Wang J (2002) Location-based services: technical and business issues. Gps Solut 6(3):169–178
9. Davis F (1989) Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q, pp 319–340
10. Hess B, Gasimov A, Sutanto J (2011) A universal approach that makes legacy online content location-based. In: proceedings of the 10th international conference on mobile and ubiquitous multimedia, pp 127–133. ACM
11. Hess B, Magagna F, Sutanto J (2012) Discovering the web by location with webnear.me. In: proceedings of the 2012 ACM

conference on ubiquitous computing, UbiComp '12, pp 538–538. ACM, New York, NY, USA. doi:10.1145/2370216.2370298
12. Hill L (2000) Core elements of digital gazetteers: placenames, categories, and footprints. Res Adv Technol Digit Libr pp 280–290
13. Lakhina A, Byers J, Crovella M, Matta I (2003) On the geographic location of internet resources. Sel Areas Commun IEEE J 21(6):934–948
14. Leveling J, Hartrumpf S, Veiel D (2006) Using semantic networks for geographic information retrieval. In: Peters C et al. (eds) Accessing multilingual information repositories, pp 977–986. Springer, Berlin. http://link.springer.com/chapter/10.1007%2F11878773_109
15. Lieberman M, Samet H, Sankaranayananan J (2010) Geotagging: using proximity, sibling, and prominence clues to understand comma groups. In: proceedings of the 6th workshop on geographic information retrieval, p 6. ACM
16. Magagna F, Hess B, Sutanto J (2012) Building location-aware web with salt and webnear.me. Procedia Comput Sci 10:601–608
17. Markowetz A, Chen Y, Suel T, Long X, Seeger B (2005) Design and implementation of a geographic search engine. In: 8th international workshop on the web and databases (WebDB)
18. McCurley KS (2001) Geospatial mapping and navigation of the web. In: proceedings of the 10th international conference on World Wide Web, pp 221–229. ACM, Hong Kong. doi:10.1145/371920.372056
19. Morimoto Y, Aono M, Houle M, McCurley K (2003) Extracting spatial knowledge from the web. In: applications and the internet, 2003. Proceedings. 2003 Symposium on, pp 326–333. IEEE
20. Qin T, Xiao R, Fang L, Xie X, Zhang L (2010) An efficient location extraction algorithm by leveraging web contextual information. In: proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems, pp 53–60. ACM
21. Quercini G, Samet H, Sankaranarayanan J, Lieberman M (2010) Determining the spatial reader scopes of news sources using local lexicons. In: proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems, pp 43–52. ACM
22. Teevan J, Dumais ST, Horvitz E (2005) Personalizing search via automated analysis of interests and activities. In: proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '05, pp 449–456. ACM, New York, NY, USA. doi:10.1145/1076034.1076111
23. Teevan J, Karlson A, Amini S, Brush AJB, Krumm J (2011) Understanding the importance of location, time, and people in mobile local search behavior. In: proceedings of the 13th international conference on human computer interaction with mobile devices and services, MobileHCI '11, pp 77–80. ACM, New York, NY, USA. doi:10.1145/2037373.2037386
24. Tobin R, Grover C, Byrne K, Reid J, Walsh J (2010) Evaluation of georeferencing. In: proceedings of the 6th workshop on geographic information retrieval, p 7. ACM
25. Tseng V, Lin K (2006) Efficient mining and prediction of user behavior patterns in mobile web systems. Inf Softw technol 48(6):357–369
26. Virrantaus K, Markkula J, Garmash A, Terziyan V, Veijalainen J, Katanosov A, Tirri H (2001) Developing gis-supported location-based services. In: web information systems engineering. Proceedings of the second international conference on, vol 2, pp 66–75. IEEE
27. Zubizarreta Á, de la Fuente P, Cantera J, Arias M, Cabrero J, García G, Llamas C, Vegas J (2009) Extracting geographic context from the web: georeferencing in mymose. In: Boughanem M et al (eds) Advances in information retrieval, Lecture notes in Computer Science, vol 5478. Springer, Berlin, pp 554–561