ANDREI-CĂTĂLIN TABARCEA

Location-Based Web Search and Mobile Applications

Publications of the University of Eastern Finland

Dissertations in Forestry and Natural Sciences

Number 185

Academic Dissertation

To be presented by permission of the Faculty of Science and Forestry for public examination in the Auditorium M101 in Metria Building at the University of Eastern Finland, Joensuu, on September 30, 2015, at 12 o'clock noon.

School of Computing

Grano Oy

Joensuu, 2015

Editors: Dir. Pertti Pasanen,

Prof. Pekka Kilpeläinen, Prof. Kai Peiponen, Prof. Matti Vornanen

Distribution:

Eastern Finland University Library / Sales of publications

P.O.Box 107, FI-80101 Joensuu, Finland

tel. +358-50-3058396

http://www.uef.fi/kirjasto

ISBN: 978-952-61-1868-0 (printed)

ISBN: 978-952-61-1869-7 (pdf)

ISSNL: 1798-5668

ISSN: 1798-5668

ISSN: 1798-5676 (pdf)

Author's address: University of Eastern Finland
School of Computing
P.O.Box 111
80101 Joensuu
FINLAND
email: tabarcea@cs.uef.fi

Supervisors: Professor Pasi Fränti, Ph.D.
University of Eastern Finland
School of Computing
P.O.Box 111
80101 Joensuu
FINLAND
email: franti@cs.uef.fi

Reviewers: Helena Ahonen-Myka, Ph.D.
Tibidiscis Oy
Finnoonniitynkuja 4
02270 ESPOO
FINLAND
email: hahonen@cs.helsinki.fi

Dirk Ahlers, Ph.D
Norwegian University of Science and Technology
Department of Computer and Information Science
Sem Sælands vei 9
NO-7491 Trondheim
NORWAY
email: dirk.ahlers@idi.ntnu.no

Opponent: Professor Tapio Salakoski, Ph.D.
University of Turku
Department of Information Technology
ICT-talo, 6th floor, Joukahaisenkatu 3-5 B
20520 TURKU
FINLAND
email: tapio.salakoski@utu.fi

ABSTRACT:

Location is an important factor in personalizing applications in various fields such as web search, data mining, contextual recommendations or mobile gaming. Nowadays, due to the rapid development and wide availability of positioning techniques and internet connectivity, location is easily available and significantly improves the applications that utilize the user's context.

This thesis presents contributions in the field of location-based applications. It proposes applications and advances in location-based web search and data mining, postal address detection and location-based gaming. We verified our methods and algorithms using data collected by our users within the MOPSI project.

The first part of the thesis describes an application that identifies location-based data in web pages. Location data are widely available on the web, but rarely in a standardized format. Most of the time they are available as postal addresses, especially on the web pages that describe commercial services or points of interest. We are detecting addresses by using a gazetteer-based method. Our gazetteers use freely available data sources such as OpenStreetMap. Furthermore, we extract a title and a representative image for each detected location. Our goal was to provide information that is close to the user's location, related to the keywords provided by the user and extracted from the content of websites.

The second part of the thesis describes a location-aware mobile game that promotes physical exercise by applying concepts from the classical game of orienteering and uses a geo-tagged photo collection created by users.

The results of the work documented in this thesis were integrated into services that are available to users through mobile phone application or web pages.

Universal Decimal Classification: 004.738.5, 004.774, 004.775, 004.78, 004.9

Library of Congress Subject Headings: Mobile computing; Mobile apps; Location-based services; Wireless localization; Geographical positions; Global Positioning System; Data mining; Web sites; Internet; Street addresses; Mobile games; Orienteering; Cell phones; Smartphones

Yleinen suomalainen asiasanasto: mobiilisovellukset; mobiilipalvelut; mobiililaitteet; paikannus; tiedonlouhinta; WWW-sivut; Internet; osoitteet; mobiilipelit; suunnistus; matkapuhelimet; älypuhelimet

INSPEC Thesaurus: location-based application; web data mining; mobile game; GPS; orienteering; postal address detection

Acknowledgments

First of all, I would like to extend my sincerest gratitude to Professor Pasi Fränti, head of the Speech and Image Processing Unit from the School of Computing at the University of Eastern Finland, for his support and help with research throughout the years. None of my accomplishments and experience gained within these years would have been possible without his guidance. I would also like to thank all the members of the Speech and Image Processing Unit, the administrative staff or School of Computing and all the people that worked within the MOPSI project for creating a great working atmosphere and for providing help and support with my work whenever I needed it. I have learned a lot from all of you.

I am also grateful to Professor Vasile Manta and the Technical University of Iaşi for supporting me and for providing the opportunity to study and research abroad through the Erasmus program and through the joint doctoral agreement.

Last but not least, I would like to thank my family, friends and Cristina for their moral support and care.

This research has been supported by the East Finland Graduate School in Computer Science and Engineering (ECSE), the Technical University "Gheorghe Asachi" of Iaşi, the University of Eastern Finland, the Nokia foundation, and the MOPSI and MOPIS projects. All their support is gratefully acknowledged.

Joensuu September 3, 2015 *Andrei Tabarcea*

LIST OF ABBREVIATIONS

DOM	Document-Object Model
GIS	Geographic Information Systems
GPS	Global Positioning System
HTML	HyperText Markup Language
LBS	Location-Based System
MOPSI	Mobiilit Paikkatieto-Sovellukset ja Internet (Mobile location-based applications and Internet)
URL	Uniform Resource Locator

LIST OF PUBLICATIONS

This thesis presents a current review of the author's work in the field of location-based applications, and the following selection of the author's publications:

[P1] P. Fränti, J. Chen, A. Tabarcea, "Four aspects of relevance in location-based media: content, time, location and network", Int. Conf. on Web Information Systems and Technologies (WEBIST'11), Noordwijkerhout, Netherlands, 413–417, May 2011.

[P2] P. Fränti, A. Tabarcea, J. Kuittinen, V. Hautamäki, "Location-based search engine for multimedia phones", IEEE Int. Conf. on Multimedia and Expo (ICME'10), Singapore, 558–563, July 2010**.**

[P3] A. Tabarcea, V. Hautamäki, P. Fränti, "Ad-hoc georeferencing of web-pages using street-name prefix trees", Int. Conf. on Web Information Systems and Technologies (WEBIST'10), Valencia, Spain, vol.1, 237–244, April 2010.

[P4] A. Tabarcea, N. Gali, P. Fränti, "Location-aware information extraction from the web" (manuscript), 2015.

[P5] N. Gali, A. Tabarcea, P. Fränti, "Extracting representative image from web page". Int. Conf. on Web Information Systems and Technologies (WEBIST'15), Lisbon, Portugal, May 2015.

[P6] A. Tabarcea, K. Waga, Z. Wan and P. Fränti, "O-Mopsi: Mobile Orienteering Game Using Geotagged Photos", Int. Conf. on Web Information Systems and Technologies (WEBIST'13), Aachen, Germany, 8–10 May 2013.

The original publications are included at the end of this thesis by permission of their copyright holders. Throughout the overview, these papers will be referred to as [**P1] –[P6]**.

AUTHOR'S CONTRIBUTION

The contributions of the authors of these papers to this dissertation can be summarized as follows: In [**P1**] the authors define four aspects of relevance in sharing location-based media: location, time, content and social network and study how they appear in media sharing platforms. Prof. Pasi Fränti wrote the paper, Jinhua Chen developed the web interfaces and the author implemented the mobile software, contributed to text writing and performed all experiments for this paper.

[**P2]** describes a location-aware search engine for web and mobile environment. It sketches the overall scheme of the MOPSI search engine prototype, defines all the needed core elements and tests a prototype for Finland. The idea was proposed by Prof. Pasi Fränti. The first draft of the search engine was developed jointly by the author and Juha Kuittinen, but the author was responsible for the version used in this paper, performed all mobile-side programming and experiments and also is the main contributor in writing Sections 3 and 4.

[**P3]** describes and tests the address detection algorithm in [**P2]**, which is based on individually detecting address elements and aggregating them as address candidates that are validated using gazetteers. **[P4]** improves [**P2]** and [**P3]** by replacing plain text extraction with processing of the DOM representation and by improving the methods for extracting the title and representative image for each search result.

For [**P3]** and **[P4]**, the author was the main contributor for the development of ideas and technical solutions, and was the sole person to implement all the related mobile applications. He performed all experiments and was responsible for writing the paper. Other authors mostly provided supported mostly by means of advice and text revisions.

[**P5]** studies how to select a representative image to represent an entire web page. The authors propose a rule-based method to categorize images based on their purpose in the web page. This solution is needed as part of the summarization of the web page

found by MOPSI search. The paper is a result of team work where Najlah Gali and the author jointly contributed to the idea development. The author contributed to the implementation of the proposed method, the experimentation, and paper writing.

Finally, **[P6]** described a location-based mobile orienteering game that aims to promote physical exercise and learn new technologies. The game is based on user-generated data from our MOPSI system and was presented during a yearly international festival in which middle- and high-school students learn about science, technology and the environment. Prof. Pasi Fränti contributed with the idea, Karol Waga and Zhentian Wan made the web implementation, and the author created the mobile solutions. The author also wrote the paper and did all experimentations; all authors contributed to organizing the O-Mopsi workshop in SciFest, where the test material was collected.

Contents

1 Introduction

Exploiting the users' geographical location has become more and more popular during recent years, mainly because of the increasing availability of GPS enabled mobile devices such as smartphones or personal navigators and the constant decrease in the prices of such devices. Additionally, extra positioning methods such as cellular network positioning and Wi-Fi positioning facilitate the access to the users' location. Therefore, during the last years there has been increasing interest in the research of location-based services, both in academic and commercial projects.

A location-based service is an application which integrates the user's geographical location with the general notion of service with the purpose to provide information about a certain place or geographical location [ScVo04]. Usually, location-based services are accessible through mobile devices connected to a mobile network and they use the location information provided by the mobile device. There are many categories of location-based services, such as: navigation, search and providing information, monitoring, advertising, management, games, socializing etc. A location-based application is an application that uses such services.

Location-based services are part of the larger field of context-aware services, which are services that adapt their way of functioning according to one or more parameters which reflect the context of targets or users [Küpp05].

Location-based data are very common on web-pages, especially when their content describes commercial services, landmarks or public institutions. However, the location data are rarely embedded as geographical coordinates that can be retrieved automatically, but are more commonly presented in a

human-readable way that can be retrieved using location-based web search.

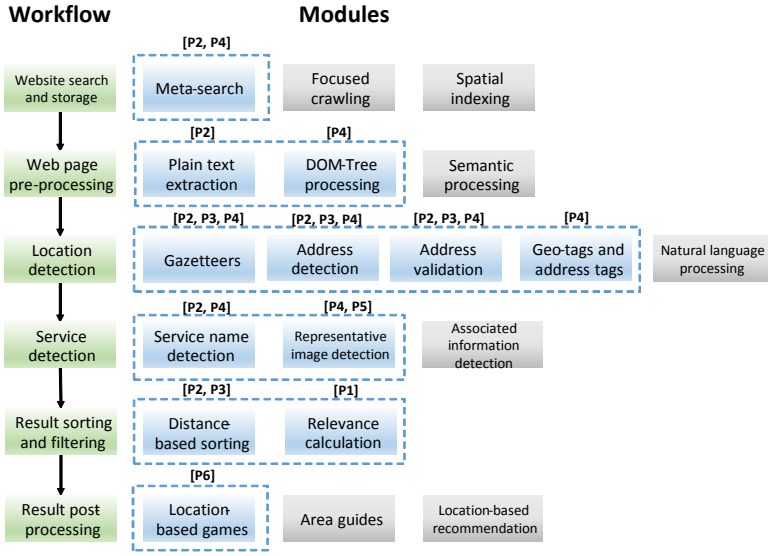


Figure 1-1 Typical workflow and modules for a location-based search solution

As shown in ***Figure 1-1*** (left), a typical workflow of a location-based search solution requires the following steps: website search and storage, web page pre-processing, location detection, service detection, result sorting and filtering, and, optionally, result post-processing. These steps are the research topics covered in this thesis. ***Figure 1-1*** (right) shows possible approaches for each of the proposed steps, out of which we highlighted the modules that are covered in this thesis and in the papers that support it.

As positioning is ubiquitous in our electronic devices and more and more location data are produced every day, there is an increasing need to exploit the location data on the web and to provide results that are relevant in a specific location and are presented to the user in a clear and informative way. In this dissertation, we present a solution that is based on location data and has two types of applications: location-based web search and mobile applications. Our location-based web search solution proposes a method to identify location data from websites by detecting postal addresses. Our mobile applications are: a

location-based search solution for multimedia phones [P2], an application for collecting location-based data [P1] and a mobile game based on the concept of orienteering [P6]. Our mobile game, O-Mopsi, shows an example of how to use location data after they have been identified using location-based search or after they have been collected by mobile applications. Our solution and applications have been integrated into a location-based platform called MOPSI.

1.1 MOPSI PROJECT

The work in this thesis has been carried out within the MOPSI project[1], which is a research project for location-based services that is developed by the Speech and Image Processing Group from the School of Computing at the University of Eastern Finland. MOPSI offers multiple uses of location-aware applications, being a test-bed for various research topics that involve location-aware data. It contains tools for collecting, processing and displaying location-based data, such as photos or trajectories, along with social media integration.

The main topics addressed in MOPSI are: collecting location-based data, mining location data from web pages, processing, storing and compressing GPS trajectories, detecting transportation mode from GPS trajectories, recommending points of interest, using location information in social networks, and detecting users' actions by using their location and building location-based games with the help of user-generated collections.

MOPSI provides tools to collect GPS trajectories and our collection includes more than seven million GPS points, which are assigned to more than 8.000 trajectories. We designed a system for fast retrieval and displaying of the data [WTMF13] that is based on GPS trajectory polygonal approximation [ChXF12]. GPS trajectories are also compressed for optimizing storage. Furthermore, transport mode information can be

[1] http://cs.uef.fi/mopsi

retrieved from automatically analyzing GPS trajectories. We are using a second order Markov model to segment the trajectories and to detect stops, bicycle, running, or car transportation modes [WTCF12]. Furthermore, we have developed a system that calculates the similarity of GPS trajectories using a low complexity spatial measure [MTSF14].

The relevance of location-based media can be assessed by considering several aspects such as time, location, content or social network [P1], which are used to create a context for each user. Using our applications, users can collect geo-tagged photos; our collection includes more than 35.000 photos. A personalized recommendation system can recommend relevant data based on user location and on user context [WaTF12]. Such data can be geo-tagged photos, services confirmed by administrators or GPS trajectories.

Users can share their location in real-time by using mobile phone location-aware applications. This allows for the detection of various location-based actions such as meetings, visiting or passing-by points of interests [Mari13].

MOPSI also includes location-based games, such as O-Mopsi [P6], [Wan14], which describes how to create an orienteering game using the data from a user-generated photo collection and how to develop a web interface and a mobile application.

MOPSI provides tools for collecting location-based data with mobile devices. It is available on most of the major mobile operating systems (Android, iOS, Windows Phone, Symbian). On the server-side we process and display the data collected by users and also provide social features and integration with social media, with functionalities such as chatting, friend tracking and sharing data to Facebook.

1.2 FOUR ASPECTS OF RELEVANCE IN LOCATION-BASED MEDIA

Location-based services are becoming widely used due to the fast development of positioning systems in multimedia phones. Location provides additional information that can be expressed as a point of interest, route or geographic area. Location itself can be considered as information, but it is often attached to other data and shared via location-based services or photo sharing sites.

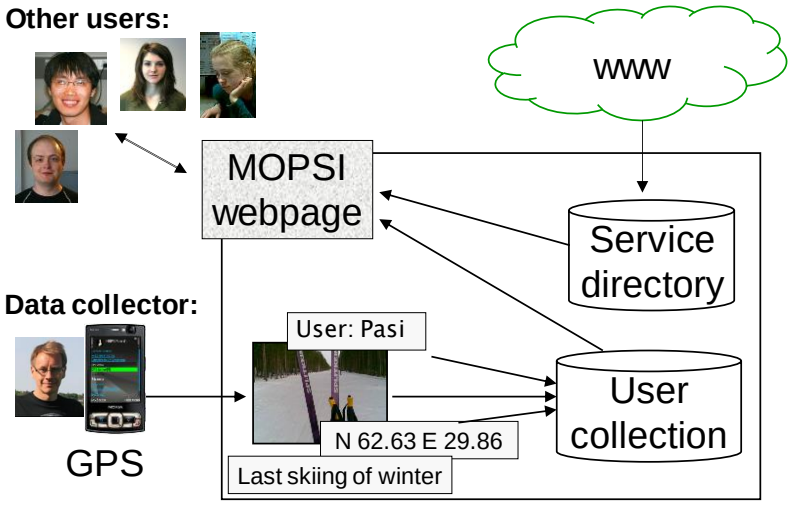


Figure 1-2 Diagram of the MOPSI data collection and services

We study mobile location-based media sharing via the internet in a case study based on the MOPSI service, which is a prototype service for sharing location-based media. The overall structure of the system, outlined in ***Figure 1-2***, consists of two main parts: user collection and service directory.

The main limitation of this kind of ad-hoc information sharing is unawareness of the material of others, especially if the users are not directly linked with each other. The data may be available in the service, but the problem is how to find the *relevant* data from a service with a large number of users. We argue that relevance can be defined by the following aspects:

1. Content of the data
2. Location

3. Time
4. Author and his/her network

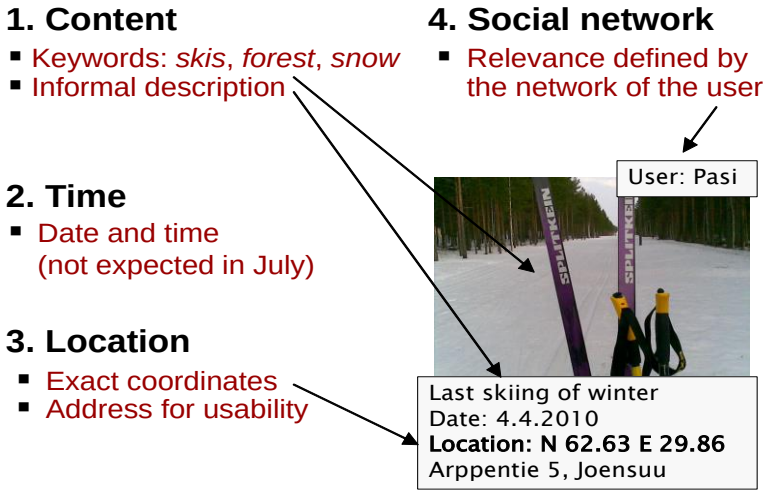


Figure 1-3 Four aspects of relevance in practice

These four aspects are demonstrated in ***Figure 1-3*** by a concrete example, where a person wanted to capture the following scenario. From the photo and its description we can see skis, forest and snow, which relate to wintertime activity. The data also reveals when and where the picture was taken. In 4th April 2010, there were skiing tracks available, which was not self-evident even for citizens of Joensuu. Knowing the proper location was essential. The last piece of information is the identity of the user himself. Strangers may not benefit much of this information, but those who know him and share the same hobby are more likely to find this useful.

1.2.1 Content

Traditionally the relevance is defined by *content* either by user-given keywords or using a predefined format in a database system. This requires a well-designed static database where the service provider models the user behavior beforehand and provides information in the form of a service directory.

On the Internet well-defined attributes are not used, but relevant content can still be found from free text using search engines if the content matches the keywords provided by the user. Tagging of the photos can also be done afterwards, but usually a free-form textual explanation is simpler. It also serves the purpose of social media.

In our application, instead of using manual tagging, we support free-form text description. We implemented queries based on time, location and content for browsing our data on the web. We also implemented a simple recommendation framework based on user location and rating of the photos [WaTF12].

Further analysis of the relevance of content-based image retrieval could be done based on features such as color, texture and shape. Automatic image categorization aims at converting visual content into a set of keywords to describe the content. In [CSLJ09] and [YKSJ09] both visual content and user tagging are jointly applied to recommend the group where a photo should fit best.

1.2.2 Location

Exploiting the l*ocation* of the user has become popular due to the wide availability of GPS positioning in multimedia phones. In case of lacking GPS coverage, positioning can also be provided by the cellular or WiFi network of a mobile phone, or even by using the IP address for a rough estimation of location. Once the location is known, it gives significant additional relevance that can be utilized in several different ways. In our system, location is the key element and it provides additional relevance in the following ways:

1. Browsing data collection on a map
2. Showing the location of other users
3. Tracking the movements of the user
4. Filtering relevant search results for the service directory

Figure 1-4 demonstrates the map view where photos have been clustered and then shown using Google Maps API.

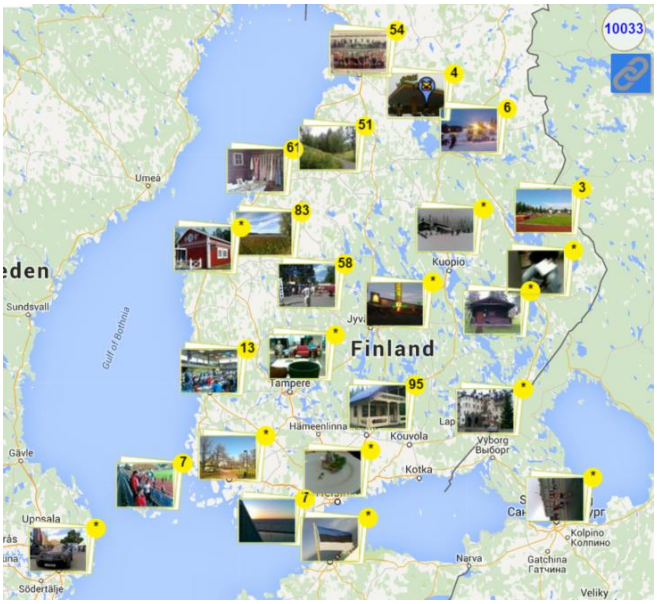


Figure 1-4 Map view of the data collection

Location of users has been visualized in ***Figure 1-5***, using a so-called *smart swap* algorithm [ChZF10] that provides accurate clustering in real-time. For representing the clusters, approaches using icons, grids, Voronoi diagrams, and coloring by the density have been considered in [Delo10]. We use a color bubble attached with the text representing the most recent users in the cluster. The browsing is supported by a zooming operation to get inside bigger clusters.

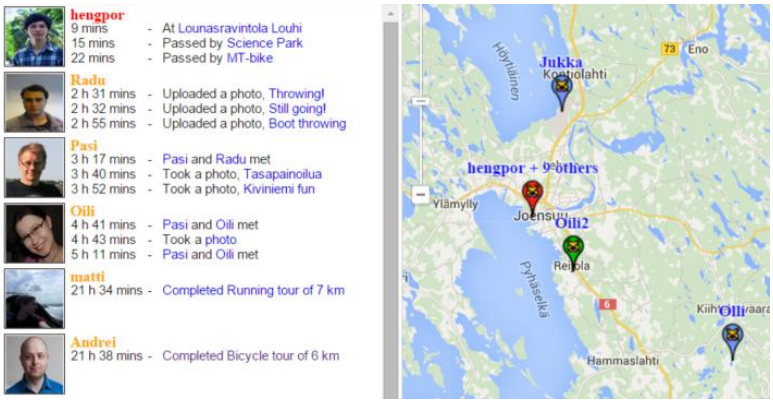


Figure 1-5 Map view of user locations

The collection can also be used as a part of service directory in MOPSI either in mobile phones or on the web, see **Figure 1-6**. Given the location, the user enters a query by using keywords, but instead of providing relevant search results by content alone, results nearby are given if they exist in a local database (green), or found in the user collection (yellow).

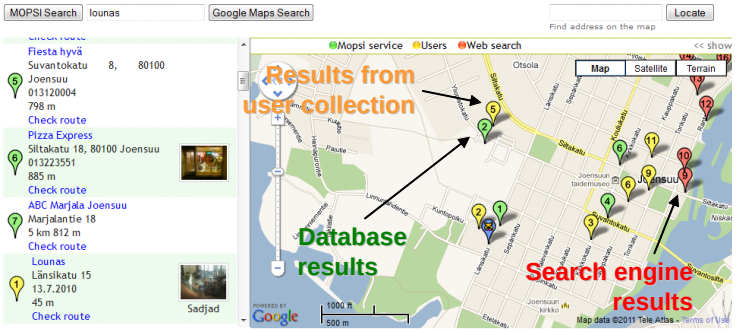


Figure 1-6 Web page interface to the service directory

Additional information (red) is provided by *location-based search* [P2], which is a combination of traditional location-based service and search engine. Following the idea in [HuFi10], our system allows users to transfer search engine results (red) into the service directory (green) by adding proper keywords similarly, and by using photos from the user collection (yellow).

1.2.3 Time

Time can be added to the relevance of the data in several ways. Firstly, the information may be relevant only within a specific time period. A concert or a sports event taking place at a date and time is essential information for the participants. In photo collections, it is also relevant to know when photos were made. In our collection, we utilize this by providing a time line view of the data as shown in ***Figure 1-7***. A similar layout was considered in [SeBD09], with the addition that also links to Wikipedia are supported, to provide more information in addition to the photos.

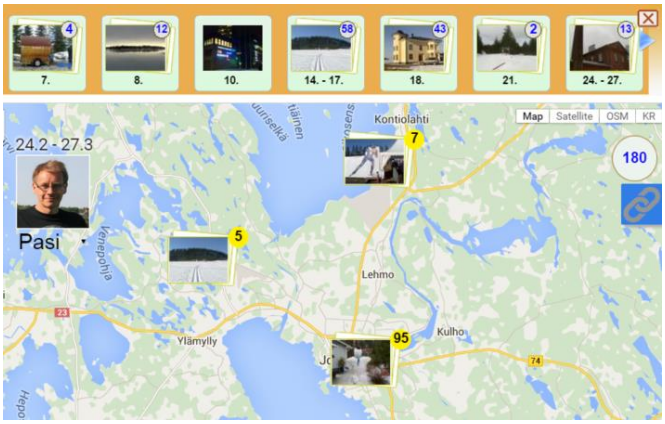


Figure 1-7 Time-line view of the data collection

Secondly, the time and location themselves can be the essential data from an exercise session. For example, the skiing track shown in ***Figure 1-8*** records the length, duration and average speed. This is typical record keeping in the training of a cross-country skier. Although there are specialized GPS sport trackers, the use of our service and mobile phone allows for automatic and real-time sending of the data to the server for user convenience. Moreover, photos can also be taken from the same session by the same device, and presented later jointly with the trajectory of the user as proposed in [PCRC08].

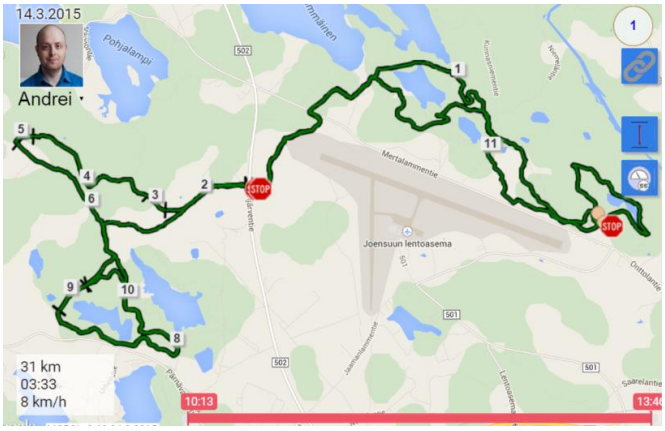


Figure 1-8 Joint time and location for tracking sport activity

In our collection, tracking user's routes is one of the main functions. The web interface also provides navigation from the current location to the location of the search result using Google Maps API based on road maps. An interesting idea for future consideration would be to use the route collection of all users to offer better navigation for pedestrians and hikers instead of the road network that is more suitable for cars [KaKa09].

The third possibility to utilize the time information is to consider the age of the data. The newer the information the more likely it is still valid, as the life expectancy of cafeterias, for example in typical metropolitan areas are often measured in months rather than in years. Moreover, information such as weather condition is needed right here and right now. In ***Figure 1-3***, the skiing conditions are recorded for 4th April, but are hardly relevant for users in July.

1.2.4 Experiments

Next we will provide an overview of the data collected by our users until 25th October 2010. The collection includes many test photos, and the number of users is small, which may somewhat skew the results. Nevertheless, some trends and observations can be seen.

In total, there were 3.589 photos of which most are city views (839), followed by pictures of nature (801) and other people (279). There are also a few pictures of events (90), documents (40) and animals (59). In addition, there are photos that are counted as test photos or failures.

Another point of view is what kind of descriptions has been typed in by the users. Due to the experimental stage, a large percentage of the photos (27%) lack any description. The lack of descriptions is also caused by the difficulty to type by mobile phone, but descriptions can be added later from the web interface.

Among the photos that have some kind of description, a significant share (35%) are only random, some test word (*Symbian_test*), or very generic object descriptions (*Mug*, *Wires*, *Mouse*) indicating test use. In total, 65% of all photos have a

meaningful description. The most documented descriptions are travel photos of places (685), nature (579), general objects (263), architecture (212) people (210), and a few general descriptions of events and animals.

People are often described by their names, or by their roles (*runner*, *floorball player*). Only few are related to place (*Untung / STMIK*), age (*Young Andrei*) or relationship to the person (*my son Amir*).

Events are significantly more often found in the user description than could be concluded by content analysis alone. In our case, events include mostly work-related meetings described by their acronyms (*ecse*, *abi*, *mopsi meeting*, *ubiikki*) but also running competition (*Åland half marathon*) and actions attached with feelings (*quality time in skiing elevator*).

Another difference between content and user descriptions are in travel photos. The location is not easy to recognize from the content but it could be concluded from the positioning data. For example, *Clarke Quay*, *Geger beach*, *Suceava*, *Tahkovuori* and *Aholansaari* are locations whereas the following descriptions include additional details: *Petronas Towers* (building complex), *Heureka* (science center), *Singapore flier* (Ferris wheel) and *Olavin linna* (*castle*). The extreme case is *Musta Pekka mutkan takana* (Black Pete behind the curve) where Black Pete is the name of a particular slope in Tahko skiing resort.

Table 1-1 compares the textual description used in our system with two other photo-sharing sites, Picasa[2] and Flickr[3]. In our system, location is provided automatically without any user interaction, whilst in Picasa or Flickr, location is either manually annotated by users or taken from the photos' meta information, especially if they are taken using mobile phones.

[2] http://picasaweb.google.com

[3] http://www.flickr.com

Table 1-1 Distribution of keywords (tags) used in Picasa and Flickr, in comparison to the user descriptions of the MOPSI collection.

Description	Picasa	Flickr	MOPSI	
			All	Real
Places	---	28%	21%	32%
Events and action	31%	17%	5%	7%
People	6%	7%	6%	10%
Objects	---	5%	8%	12%
Architecture and nature	25%	21%	23%	37%
Animals	---	3%	2%	2%
Other	20%	16%	---	0%
Garbage	19%	2%	35%	---

In Picasa, users provide the location by dragging the photo on Google Map. Keywords and location are thus provided explicitly as two different entities, and consequently, users tend not to type any location related keywords. Flickr has a somewhat more complicated interface based on *Yahoo! Maps*. Only a predefined set of keywords are allowed, which explains the quality of the tags (only 2% garbage).

Despite the automatic positioning in our system, it does not reflect on the distribution of the type of descriptions written. Unlike in Picasa, users still tend to describe the location anyway for travel pictures, probably because the position is not confirmed in the device, but it's retrieved in the background. Overall, the distribution of topics is rather similar to that of Flickr. There are slightly more people and objects described, but these could be just artifacts from the system being in the testing stage.

For the purpose of photo collecting, two mobile applications were developed (Java and Symbian C++). Samples are shown in ***Figure 1-9***. A large number of failures were caused by the Java version, which lacks several important features. Firstly, there is an unavoidable delay from the *click* sound and when the photo is actually taken. People tend to move the camera right after they hear the sound and before the actual picture will be taken. Secondly, auto-focus supported by Symbian helps very much

with picture quality, but it was not available in Java. Other typical failures originated from low quality cameras that do not work well in low illumination. A few faulty pictures were caused by irrecoverable transmission errors.

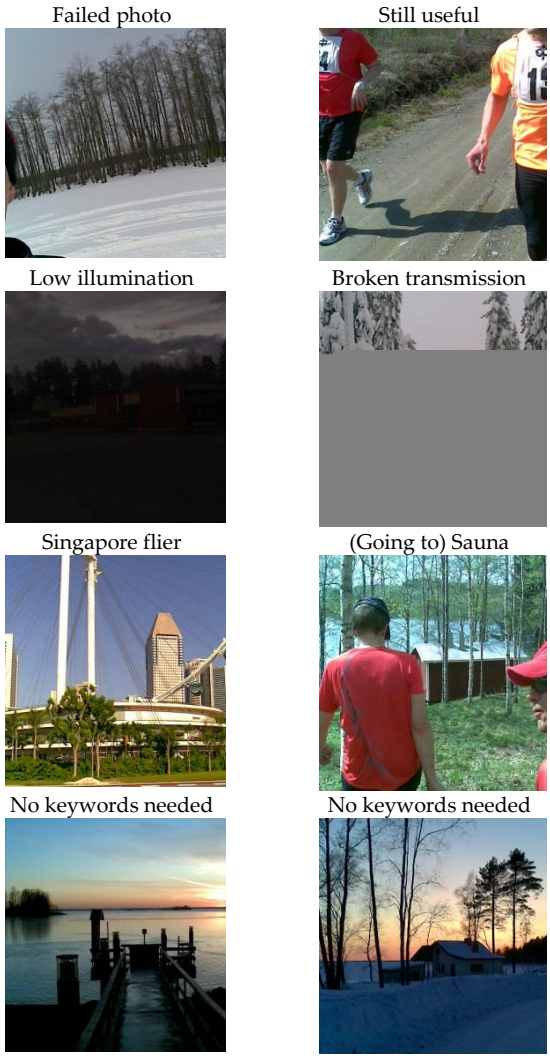


Figure 1-9 The photos in the first row are examples of software problems (click sound), the second row of low illumination and broken transmission problems. The rest are successful photos

1.2.5 Conclusions

We have presented a tool used for collecting user data (mainly photos and routes), serving as a test bench of new ideas, and a prototype service directory. We have discussed how different aspects of relevance appear in location-based data and tested the named aspects in different modes such as search, map views, or timelines. Our tool can be used later for mobile location-based games and as an educational tool for teaching principles of GIS. Although our tool is used for collecting data from users, the same discussions and conclusions on the aspects of relevance can also be applied on location-data that are generated through other means, for example resulting from automated web search or data mining.

2 *Location-Based Web Search*

Location is an important factor to personalize web search. This is because the content of a website has an area of interest that influences its relevance [BRWY11].

The volume of geospatial data is increasing as more and more devices have access to the Internet and positioning technology [PaMP03]. A large part of this multimedia data are nowadays generated with devices that automatically annotate them with location information, but free-form content such as websites do not implicitly contain any geographical information. Mobile search engines allow users to find information anytime, anywhere and the search performance is influenced by the type of mobile device and the user's context, which is influenced by aspects such as location, profile, previous activity, time of year or social network [LiRG10].

Location-based services such as Fonecta[4], Google Maps[5] and Nokia Ovi Services[6] emerged very fast in our everyday lives via mobile phones and other consumer electronics. Their main limitation, however, is that they are fully or partially based on databases where the entries must be explicitly geo-referenced beforehand when added. *Search engines*, on the other hand, are efficient in finding information from the Internet without any prior knowledge or explicit search structure. Their limitation is that the location of the user is not yet well utilized in the current solutions. This is because the information on web pages is rarely attached to the location for which it would be relevant.

[4] http://www.fonecta.fi

[5] http:/maps.google.com

[6] http://www.ovi.com/services/

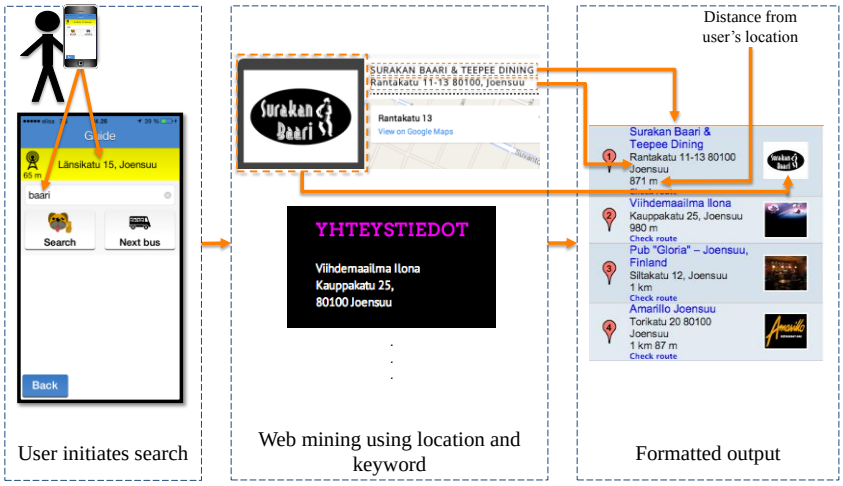


Figure 2-1 Web mining using location and keyword

Location-based search aims at finding a business or place of interest around a specific geographical location. This is supported by search engines that support geographical preferences [MCSL05]; the relevance of a search result depends on the distance between the user-specified location and the location of the service [YoTM01]. Location-based search changes the search from web-oriented to service-oriented, which makes it a more challenging task due to two reasons. First, it is not enough just to find a relevant web page for the user as in traditional web search. Instead, we also need to detect the location that the web page is relevant to. Second, we need to extract information from the web page. This is either a simple summary of the content (such as title and image), but in case of service directories, we need to extract the part belonging to that particular service. It requires both the identification of geographical data and automatic information extraction from web pages. A simplified workflow of a location-based application that also applies to our system is shown in ***Figure 2-1***.

Locations are embedded explicitly as geographical coordinates (usually latitude and longitude), both in source code as HyperText Markup Language (HTML) meta tags named *geo-*

tags[7] and in the content of the web pages as plain text. Locations are embedded implicitly as *geographical references* that are found in the text content of the web pages in many ways, such as: postal addresses, place names, descriptions in natural language and driving directions. Identifying geographical references and associating a web page with one or multiple locations is a process called *geo-referencing*. A particular case of geo-referencing is *geo-tagging*, which is the operation of assigning geographical coordinate metadata to multimedia such as photos, videos and websites.

Very few web pages are using explicit localization. A world-wide study does not exist, but according to [Väns04] less than 0.1% of Finnish websites were using geo-tags in 2004. Furthermore, less than 1% of the websites related to the German city of Oldenburg were using explicit localization in 2008 [AhBo08a] and 7% of the service websites from Finland collected in MOPSI until May 2015 [P4]. Therefore, the main method of geo-referencing web pages is to detect the implicit locations from their content. According to [Mccu01], including postal addresses is the most common method of implicit localization, especially for the pages that describe commercial services.

In this section, we propose an alternative solution based on web search and ad-hoc geo-referencing. We aim at combining the benefits of web search and traditional location-based services exploiting the location. We define a location-based search engine as being a web search engine in which the geographical location is an additional relevance criterion. The general idea behind our solution is the first implementation of the idea originally outlined in [HaFM02] and improves most of its technical aspects. Here we describe the technical solutions for implementing the system on multimedia mobile phones and provide experimental comparison of its search capability, in comparison to existing location-based service solutions such as Google Maps and Yellow Pages.

[7] http://www.w3.org/2003/01/geo/

2.1 LOCATION-BASED SEARCH APPLICATIONS

The first methods of assigning locations to web resources, such as [BCGG99] and [WaAm03], rely on identifying the host locations, which are the location of the owner or the administrator of the website. These methods assign a single location for each website by querying its *Whois*[8] records for the address and telephone number of the network administrator. This method is expanded in [Mccu01] by additionally using hyperlinks, meta tags and postal addresses as sources for location information.

Most of the websites are not geo-tagged by default and their content cannot be directly used by location-aware services. Web pages are designed to be browsed by humans and contain geographical references that are complex, informal, diverse, ambiguous and difficult to be processed by a computer [ShBa11]. However, we can adopt an unsupervised process of extracting locations from web pages. According to [HuLR05], several strategies can be used for geographic reference extraction: text matching using gazetteers, rule-based linguistic analysis, text matching based on regular expressions, identification of host locations and reading geographic meta-tags. Our application uses text matching based on gazetteers and regular expressions.

According to [WXWL05], there are three types of locations that can be inferred from web pages: provider location (where the owner of the page is), content location (where the content is pointing to) and serving location (the area for which the web page is relevant). Provider location is detected using a set of heuristic rules such as referred frequency, URL levels and spatial positions of address strings in the web page. Content locations are calculated by extracting all geographical references with the use of probabilities that measure the reliability of each source. The probabilities are based on the measures of power and spread of a geographical reference [DiGS00] and use a location hierarchy *country-state-city* in the form of a geographic tree. Serving location

[8] http://www.whois.net

is found in a similar way, but it additionally uses links between pages and user visits logs. Our application is focused on detecting the content location by extracting geographical references.

Postal addresses are the most common way geographical references are found in web pages [GoWK07]. They are converted into locations by services that provide *geo-coding*, which is the process of finding geographical coordinates (usually latitude and longitude) from other types of location data, such as street addresses or postal codes. Our application detects addresses using free geo-coding services. We use OpenStreetMap services and build a geo-coded database for Finland with publicly available data. According to [FLMN10], because of occasional service unavailability and data accuracy, free geo-coding services such as OpenStreetMap or Geonames [9] are best suited for applications that require almost accurate geo-tagging, whilst systems that deal with public health or vital services require higher quality data.

One of the biggest challenges that arise when identifying locations from a web page is the ambiguity of terms, which can be between locations with the same name (known as *GEO/GEO ambiguity*) or between location names and non-geographical entities (known as *GEO/NON-GEO ambiguity*). Both types of ambiguity can be resolved using heuristic rules, but the algorithm [ZJLY12] additionally attempts to resolve the GEO/GEO ambiguity using an algorithm similar to Google's Page Rank [PBMW99]. In our case, ambiguity is a smaller problem because we detect postal addresses, which have a low level of ambiguity when they contain accurate elements such as postal code.

There has been several works reported in the literature for location-aware search.

A location-based search engine for the Singapore area [Tsai11] is able to search for locations by using filters on area names, building names, landmark types, business names and business

[9] http://www.geonames.org

categories. It uses the location of the user along with search filters to select items from a catalogue of businesses and landmarks.

A personalized mobile search engine enhanced with capturing users' preferences in the form of click-through data is proposed in [LeLL13]. The users' preferences are captured in the form of concepts, which are modeled as ontologies and separated into location concepts and content concepts. The search engine also considers users' GPS location and uses content and location entropies to balance the content and location concepts. The locations from documents are detected using a predefined ontology that uses city, province, region and country names.

A location extraction method that receives websites as input and equips them with location tags, being able to extract location with a precision up to street level is outlined in [HeMS13]. Words are extracted from the web page and checked against free gazetteers (Geonames and OpenStreetMap) using Aho-Corasick string matching algorithm [AhCo75]. The validation and disambiguation of the locations is done by detecting their context with the use of the other geographical references from the text. The method outperforms a commercial solution (Yahoo! Placemaker), but it correctly detects just 60% of locations. The authors demonstrate the applicability of their method by describing three practical applications derived from their work: location-aware Web surfing through a mobile device, browsing by using nearby tags, and location tagging through social networks.

A system that is capable of handling geographical queries of the triplet of <theme> <spatial relationship> <location> is described in [PCJA07]. It handles spatial relationships such as *inside*, *near*, *north--of*, *south-of* and geo-references, stores and indexes web pages using both pure text and spatial indexes. Using both types of indexes enables a full set of geographical query operators, graphical query formulation and the ranking of results according to conceptual as well as spatial criteria. Geographical ontologies are used for query expansion and for disambiguating the queries and the extracted locations. The

system relies on web crawling, pre-processed indexes and combines textual and geographical relevance. Locations are detected using a gazetteer lookup approach, which is enhanced with context rules and additional name lists used for filtering.

A system that leverages on contextual information, which can be used in the named entity recognition and disambiguation steps is described in [QXFX10]. A set of location evidence is built, updated and used to provide geographical contextual evidence.

A method to identify address data by combining patterns and gazetteers from free sources such as OpenStreetMap is used in [SMRS13] to identify companies from web pages. The web pages are pre-processed by removing HTML tags, extracting text, line splitting, tokenizing and part-of-speech tagging. The single attributes *postal codes*, *city names*, *street names*, *street numbers*, and *company names* are identified on the pre-processed data using regular expressions and heuristic rules. The attributes are then aggregated starting with the company name.

A method for extracting postal addresses and associated information using sequence labeling algorithm is introduced in [ChLi10]. Unlike most of the existing methods, postal addresses are not detected by using gazetteers. Instead, addresses are detected by pre-processing data with a named entity recognition tool, extracting features from text and training models using support vector machines and conditional random fields. Pattern mining is applied to identify the boundaries of address blocks and to extract the associated information for each detected address. The associated information is defined as information that refers to the detected addresses and allows better comprehension.

A knowledge-based web-mining tool that adopts a geospatial ontology, a rule based screening algorithm and inductive learning for automated location retrieval is described in [LGCZ12]. Address detection is customized to discover the locations of emergency service facilities; other detected addresses are discarded.

A method that analyses the DOM tree of a web page is used to extract product data from company websites [DoHu12]. The leaf nodes of the DOM tree are analyzed and used to generate semantic information vectors for the other nodes, which in turn are used to generate a maximum repeating semantic vector pattern. The generated pattern is used to detect product data regions and to build product templates, which are used along with a semantic tree matching technique to identify product information.

A vision-based approach of extracting data records from web pages is proposed in [LiMM10]. Instead of using HTML structure and DOM trees, the method uses the visual block trees generated using the algorithm described in [CYWM03]. The visual block tree is primarily based on the visual features that humans can capture from web pages, using the information from the page layout and attributes such as fonts and background color. The data records are extracted based on the visual block tree and the visual features of its elements, which are rectangular data blocks. These data blocks are filtered, clustered and regrouped to identify data records.

2.2 CONTRIBUTION

In this section, we propose a method for extracting locations from web pages and complement it by extracting a title and a representative image for each search result. Our goal is to integrate processes of geo-referencing, geo-coding and geo-tagging into a unified location-based system that is able to provide relevant information. This information has to be close to the user's location, related to the keywords provided by the user and extracted from the content of websites.

A schematic diagram of our proposed system is shown in ***Figure 2-3***. The proposed method is implemented in the framework of MOPSI that is a research project of location-based services [FKTS10]. Besides location-based search, MOPSI offers

tools for collecting, processing and displaying location-based data, such as photos or trajectories, along with social media integration.

The contribution of this section can be summarized as follows:

We propose a location-based system that uses a meta-search approach similar to [LeLL13]. We use the results of a search engine that is not location-aware and post-process them by extracting the locations from the provided websites. The meta-search approach has the advantage of allowing us to find relevant websites without crawling, indexing and storing websites, but it makes our system dependent on the relevance of other search engines and vulnerable to changes in the external search engines we use.

We extract associated information, as in [ChLi10], to bring better comprehension of the search results, in our case the name of the services or places the locations are referring to and a representative photo. The associated information is extracted by using rule-based heuristics and the DOM tree nodes that contain location information.

We download and parse the HTML source of the provided web pages and we use its DOM tree representation, similarly to [DoHu12] and unlike many methods that are concentrating just on plain text extraction.

Our approach is different than [QXFX10] because we rely on gazetteers in the address detection step, but we are also using contextual information in order to detect the entities that are related to the detected locations.

We find locations by detecting postal addresses. The proposed method in [DoHu12], using DOM, is effective and it can be applied also for detecting locations and associated information, but is limited to the websites that contain a list of items with a clear and repeating pattern. The address elements are identified individually and then aggregated in order to build an address candidate, in a method similar to [SMRS13]. The difference is that we start the aggregation with the street name and we detect the additional information in the next stages. We identify address

elements using the gazetteer approach and regular expressions. Our process is service-oriented, so we need accurate locations, not just areas as in other works, and we consider the location as postal address. Our approach is lightweight as it does not require training, but it is dependent on the quality of the gazetteer data and the addresses on the web pages.

We validate the addresses by using a gazetteer and then find the respective coordinates. In this case, disambiguation is not a problem because we aggregate elements that form a unique location, for example street numbers and postal codes.

We do not associate a single location to a single web page, but we build an ordered search result, which we rank by distance from the user's location. The search results are general and they are not limited to a theme or a type, such as products or companies.

Our search engine does not use pre-collected databases of services and relies on automatically detecting locations and extracting data in real time from web pages. Compared to [LGCZ12], we aim at a broader scope for our application and we do not limit its use to a certain type of service.

Our system is flexible and can detect locations from any set of web pages, not just the results of an external search engine. The implementation is not limited to specific geographical areas, although it is dependent on the accuracy of the gazetteer data we use. For this purpose, we use OpenStreetMap gazetteer data, which is available for most countries.

In respect to existing commercial services such as Google Maps, Bing Local[10], Yahoo Local[11] and Yellow Pages, our goal is the same: provide location-relevant information to the user. However, these applications are mainly based on commercial databases, user input and pre-collected data resulted from web crawling, and only exploit the results of real-time web search to a

[10] http://www.bing.com/local/

[11] http://local.yahoo.com/

limited extent. A location-based search engine is an alternative approach for information retrieval to traditional location-based services based on fixed databases. It aims at utilizing the location of the user, but without being restricted to any fixed location-based service.

2.3 MOPSI PROTOTYPE

Our location-based system [P4] takes as input the user's context, which is location, city and search keyword and outputs an ordered list of search results that contains the following information: rank, title, URL, location, address, representative image and distance from the user's location (see ***Figure 2-1***). The web interface of our system in shown in ***Figure 2-2***.

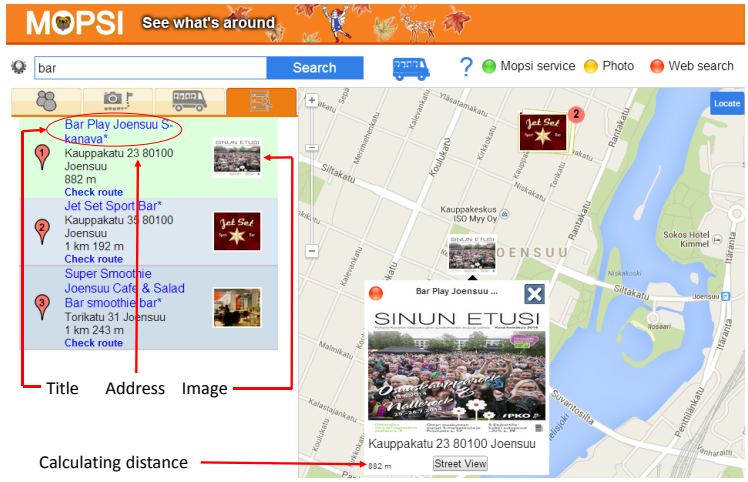


Figure 2-2 The web interface of the MOPSI search engine

The search workflow is detailed in ***Figure 2-3*** and starts by finding websites that are relevant to the location and the search keyword provided by the user. This is done by the *website provider*, which queries a conventional search engine with the <keyword, city> phrase and outputs a list of websites. A part of the *data extraction* module, the *web page parser*, downloads the web pages detected by the website provider and outputs their HTML source along with their DOM tree representation. The *address detector*

module then marks the nodes in the DOM tree that contain addresses and outputs a list of address candidates. The marked nodes are used by the *title and image extraction* module to detect a representative image and a title for each detected address. The address candidates are validated by the *address validator*, which uses our gazetteer based on OpenStreetMap. Finally, we rank the locations by distance and aggregate all the detected attributes as search results, which we display to the user as a ranked list.

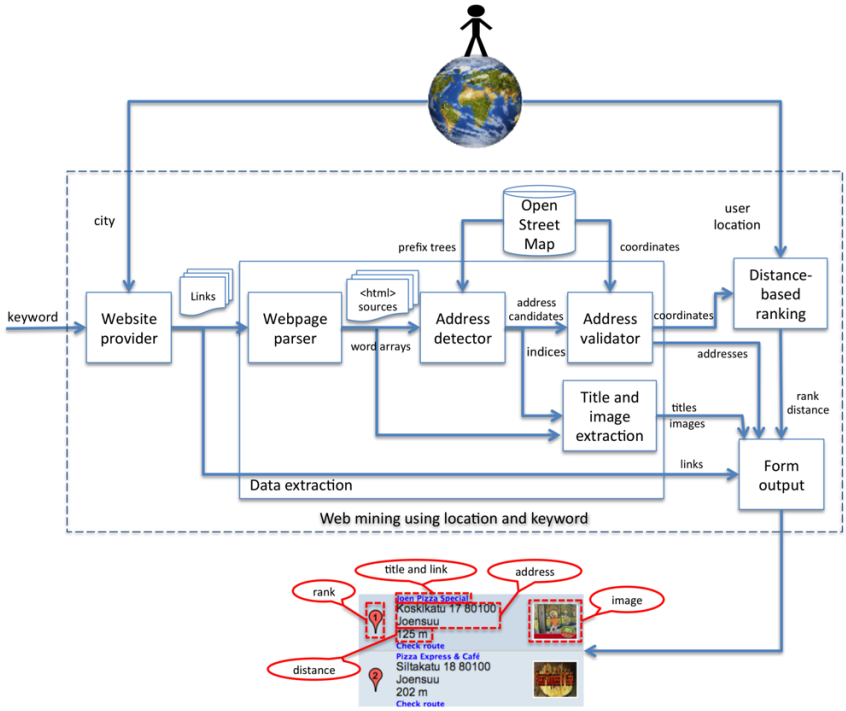


Figure 2-3 Location-based search workflow

2.4 LOCATION-BASED SEARCH MODULES

The *website provider* takes the user's context as input (city and keyword) and provides a set of URLs that are relevant to that context. The URLs are later used in the data extraction processes.

Services such as BOSS API by Yahoo![12] or Custom Search by Google[13] allow third parties to build search products by using the infrastructure of their search engine. The website provider uses those services to perform a conventional text search using the <keyword city> query. The website provider relies on the relevance of the results of the external search engine. It uses the keyword and the city provided by the user and it does not expand the query in any other way. In order to reduce the time of the search, we just use the first 10 results of the query to build the list of websites that is the output of this module.

The *data extraction* module extracts location-based data (title, URL, location, address and representative image) from the collection of web pages detected by the *website provider*. It includes the following sub-modules: *web page parser*, *address detector*, *address validator* and *title and image extractor*.

The *web page parser* uses the Document Object Model representation to generate a tree structure for each web page it downloads. The DOM tree of the web page is used in latter stages for detecting locations and location-related information such as service name and representative image.

The *address detector* searches for postal addresses on the web page using a text matching algorithm based on street-name prefix trees [P3]. It uses the text nodes of the DOM tree of the web page to identify the following address elements: street name, street number, postal code and city. Address candidates are constructed by aggregating address elements that are close to each other in the text of the web page. The prefix trees are constructed on demand for each city using our own gazetteer for Finland, see [TaFM09], and OpenStreetMap for the rest of the world. The address detector outputs a list of address candidates and marks the nodes that hold location information.

The *address validator* uses OpenStreetMap geo-coding services to validate the addresses detected at the previous steps and

[12] http://developer.yahoo.com/boss

[13] https://developers.google.com/custom-search

converts them into geographical coordinates. The coordinates are then used for displaying the results on the map and for computing the distance from the user's location. The postal address candidates that are not validated by the geo-coding services are discarded.

The *title and image extractor* identifies the title and relevant image associated with the detected locations. It uses the DOM tree and searches for text and images in the sub-trees of the nodes that contain location information. The output is a list of geo-referenced entities that contains the information described in ***Figure 2-3***.

Finally, items are sorted using *distance-based ranking* and all the information is assembled by the *form output* as a list of search results.

2.5 WEB PAGE PARSING

HTML documents are considered to be semi-structured data, which are neither raw nor strictly typed [Abit97]. HTML documents do not conform to a formal data model in the way that structured data such as a database do. They are not structured because they do not have a fixed schema and because their elements typically hold information solely for rendering. They are not completely unstructured because the HTML tags and their tree structure can be used to guide data extraction.

An HTML document can have a DOM representation, which is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents[14]. Therefore, an HTML document can be represented as a tree made up of parent-child relationships between the HTML elements. A parent can have one or many child nodes and the <html> tag is the root element of the tree. ***Figure 2-4*** shows a simplified example of an HTML

[14] http://www.w3.org/DOM

page, where we only display the sub-tree that contains the location information. After we detect the text nodes that contain postal addresses, we use the tree structure of the HTML document to detect the title of the service, see section 2.7.2.

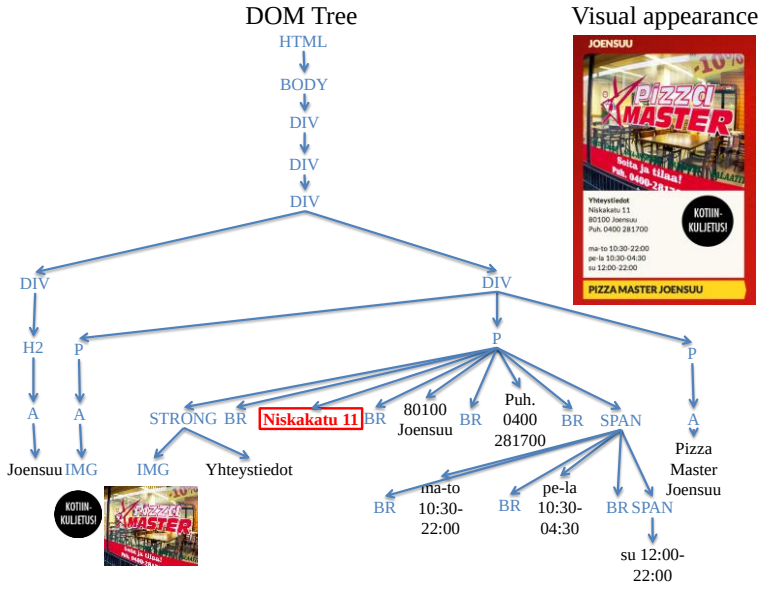


Figure 2-4 Part of a web page that contains location information: visual appearance and DOM tree

2.6 ADDRESS DETECTION USING STREET-NAME PREFIX TREES

One of the most important problems in our location-based application is how to extract and validate the geographical locations from free-form or semi-structured text.

In most languages, the street-name expression of an address is one out of several common terms. In English, it usually ends on *street*, *way*, *drive*, *road*, and in Finnish on a suffix such as *-katu*, *-kuja*, *-tie*. A simple heuristic-based method described in [FKTS10], detects street names using regular expressions with predefined endings (suffixes) and it works reasonably well for Finnish addresses. However, not all street names follow a predefined pattern, and names that have a different suffix or structure would not be detected.

A common way to detect addresses from free form text is to build and train a classifier as in [ViNa05]. However, customizing the classifier to other languages and countries takes considerable work as new ground truth tagged text corpus must be created by hand.

Another way of matching the potential address strings is to use *brute force* by comparing each word in the document to the entries in an address database. However, this can be rather inefficient if the database is large.

Detecting postal addresses and street names is a *Named Entity Recognition* problem and is usually done with the help of gazetteers. Named entity recognition without gazetteers is discussed however in [MiMG99], but it turns out to have bad results in detecting locations and addresses.

Nevertheless, most of the applications that identify postal addresses are using gazetteers. For example, the Web-a-Where system [AHSS04] uses a gazetteer and a three-step process: spotting, disambiguation and focus determination. Our address detection algorithm shares the first two steps (spotting and disambiguation), but it aims to find individual locations and services on web pages, while Web-a-Where assigns a geographical focus to a web page.

An ontology-based approach that extracts geographic knowledge is presented in [BLMD07]. The address is divided into three parts: basic address (street and building number), complement (optional, may be neighborhood name) and location identifiers (phone number, postal code, city name) and a spatial index called geo-index are built for each page. Address detection is done using the gazetteer as described in [SDBD05] for geo-parsing and geo-coding. It relies on a set of rules and patterns implemented as regular expressions using four elements: *basic address*, *postal code*, *phone number* and *city/state*. Our gazetteer also uses an ontology based on address elements, but our rules are more flexible because our address candidates do not impose any order in the address elements.

Another ontology-based method is described in [CaWJ05]. Postal addresses are detected using conceptual information retrieval combined with graph matching. Concepts, which are knowledge and address elements, are identified and linked in graphs by using semantic relations. Address detection is done by matching the graphs from a web page with template graphs that represent addresses. Geographical concepts are identified using a concept set that is a gazetteer.

A syntactic approach to postal address detection is described in [CQXW05]. It consists of two steps: vision-based text segmentation and syntactic pattern recognition. In the text segmentation step, the application analyzes html tags and identifies two types of blocks: cue blocks (for the purpose of indications, annotation, and explanation) and body blocks (main text body content). Postal address detection relies on calculating the confidence of the identified blocks, which in turn is based on tokenization of the words as city names, state names, street and organization suffixes. Our approach is different because we rely mainly on street name detection.

A graph-ranking algorithm for assigning a geographic scope to a web-page is proposed in [SMCA06]. Address detection is aided by a geo-ontology knowledge base, which uses a set of rules, relationships and heuristics.

A method of ranking web search results using geographical information is described in [LeLM07]. Address detection is done by using a regular expression to detect typical address elements. Once a pattern is detected, the street name and city name are validated using a gazetteer. Our method is different because by aggregating address elements, we consider more patterns for postal addresses, and because we use faster methods for gazetteer matching.

In [AhBo08b], describes a geo-parser that identifies address level locations. Instead of relying on metadata or other structured annotation, the geo-parser uses a gazetteer database. The database used by the geo-parser contains postal codes, city names and street names. Every city-postal code combination is also used for validation. The address detection assumes that the address

blocks have a certain structure, and that there are certain dependencies between the address elements. We also utilize the idea of identifying a number of address elements and validating the address by geo-coding it.

2.6.1 Proposed Method

We propose a rule-based text-matching solution that detects address-based locations using gazetteer and street-name prefix trees created from the gazetteer. Instead of extracting the raw text from a web page, we are using the DOM tree representation and navigate through its text nodes in order to detect postal addresses. This way we can mark the nodes that contain location information and use them to detect additional information such as title and representative image. For each node of the DOM tree, we extract the text of its sub-tree and use it for address detection. This allows us to find addresses that are spreading through several nodes (for example if one element is bold) or tables.

```
nodes = GetDOMTree(url)
cities = RetrieveCities(nodes)
FOREACH city IN cities
  prefixTree = RetrieveStreetPrefixTree(city)
  FOREACH node IN nodes
    AddressDetection(node, prefixTree)

AddressDetection(node, prefixTree)
  words = ExtractSubtreeText(node)
  FOREACH word in words DO
    IF prefixTree CONTAINS word THEN
      Search for number, postal code, city name near word using regular expressions
      IF number, postal code or city name found THEN
        Aggregate found elements into address candidate
        Get coordinates of address candidate using gazetteer
        IF coordinates found THEN
          Add address candidate to address list
          Mark node as containing address information
```

Figure 2-5 Pseudocode for address detection

Our address detection algorithm (see ***Figure 2-5***) starts by identifying street names. For each detected street name, we analyze its neighboring text and identify other potential postal address elements: street number, postal code and city name. We aggregate the detected elements into address candidates, which are then validated by our gazetteer. An address candidate needs to contain at least the following elements: street name, a street number and a postal code or a city name.

The main problem in address detection is ambiguity: we might find sections of a web page that have the same structure as an address, but do not exist in reality. We therefore validate all hypothesized addresses and discard the false detections by using a digital gazetteer, which is a geospatial dictionary of geographic names [HiFZ99]. As a side-product, the validation process provides the geo-coding, i.e. converts the given address to a pair of coordinates, which we use to plot the results on a map or to calculate the distance to the user's location.

In our approach, no ground truth tagging is needed. We only need gazetteers and simple rules on how the street name appears in relation to other address elements. Efficient use of the gazetteer is possible because we know the user's current location and its interest area consists only of those services that are close to him. Therefore, we build a fast access structure to that partial gazetteer, which is a prefix tree. We construct a prefix from all street names in a given municipality and use just the municipalities that are in the proximity of the user's location.

For testing purposes, we have also implemented a heuristic method of postal address detection without a gazetteer, which is much simpler and exploits structural characteristics of postal addresses, but its applicability is limited, as we demonstrate in our experiments section. The proposed solution is faster and more accurate than the heuristic solution alone and much faster than the brute force.

The accuracy of our address detection approach may vary, as each country has its own standards on postal addresses that can

tolerate such variations as abbreviation, different order of elements and missing elements. For example, in Finland it is common that the address block has a <*street-name, street number, postal code, neighborhood, municipality*> structure (see **Figure 2-6**). In Finnish addresses the street name and street type are concatenated, with street name being the prefix and street type being the suffix. Examples of street names are: *Kauppa-* (market), *Alexanterin-* (Alexander's) or *Kaisla-* (Reed) and types: *-katu* (street), *-tie* (road), *-polku* (way) or *–kuja* (alley). Finnish addresses have a fixed order, but neighborhood and postal code are optional. In Singapore the <*street number, street name, street type, postal code, municipality*> structure is more common, but variations exist, for example, a street name can be written using abbreviations such as Ave instead of Avenue, which is rarer in Finnish addresses, or street number can also be written after the street name.

Kaislakatu 8, 80130, Kanervala, Joensuu, Finland
Torikatu 25, 80100 Joensuu, Finland
Parppeintie 6, 82900 Ilomantsi, Finland
Aleksanterinkatu 25, 15140 Lahti, Finland
East Coast Park Service Road 1, Singapore
290 Orchard Rd., 238859, Singapore
22 Orange Grove Road, 258350, Singapore
6 Bayfront Ave, Singapore

Figure 2-6 Example of Finnish and Singaporean addresses

Although the websites we use to test our methods are from Finnish and Singaporean services, our address detection algorithm is not tailored for a specific country or language, although it only detects addresses that follow a structure similar to Western European addresses. Our method does not use a predefined order of address elements, therefore it tolerates variations in the order of elements, as we search for elements both before and after the detected street names, but it could produce false positives since it does not consider any semantic relationship between elements. The support of abbreviations

depends on the data in the gazetteer, which needs to have a separate entry for each possible abbreviation and is difficult to generate and maintain. Furthermore, the text matching method we use supports only exact matching and it does not detect wrongly spelled street names or municipalities. This should be improved in future work.

2.6.2 Gazetteer Database

2.6.2.1 Usage and Common Operations

Location-based applications that use geo-referencing require a data structure or a database that connects any given address to its exact coordinates. Such database is called a gazetteer, which is a geospatial dictionary of geographic names having the following minimum components: geographic names, geographic locations represented by coordinates and type designation [HiFZ99].

In our location-based application, we use a *gazetteer database* to store addresses in the form of <street, number, municipality> along with their corresponding geographical coordinates (latitude and longitude) for Finland and OpenStreetMap for the rest of the world. The gazetteer is used every time an address is converted into a location or when the postal address that is the closest to a known location is required. Furthermore, both OpenStreetMap and our gazetteer database are used for creating the arrays of prefix trees used in the detection of postal addresses on web pages. To speed up the location-based search, the gazetteer database has to be optimized for easy and fast access.

In our location-based search scenario, the following operations are needed:

1. Find all the municipalities within a bounding box

This query is performed when a search is initiated and determines the user's interest area. This is usually defined as a square bounding box with a fixed length and it can intersect only one or several municipalities.

In the first case (***Figure 2-7***, left) the bounding box intersects only one municipality because the provided location is near the

center. In the second case (***Figure 2-7***, right), the user's location is near the border between the two municipalities and the search engine needs addresses and location points from both municipalities.

This operation is performed once per search and its running time is not critical.

Figure 2-7 The bounding box intersects one municipality (left. Joensuu) or two municipalities (right, orange – Vantaa, blue – Helsinki)

2. Find all the street names from a municipality

In order to detect postal addresses in the selected municipalities, a list of street names has to be provided for each municipality. Instead of using all the street names in our database, we limit the search to the municipalities that are close to the user's location. For fast access, street names are stored in prefix trees.

3. Convert the detected addresses into coordinates

This operation is called geo-coding and it is performed for each detected address. Coordinate conversion is used to calculate the distance from the user's location and to validate the detected addresses. If no coordinates are found, then the address is discarded. This is one of the most time critical operations because it can be performed tens or hundreds of times per search, depending of the number of addresses found.

4. Converting coordinates into addresses

This operation is called reverse geo-coding and is used for determining the user's address.

2.6.2.2 Implementation and results

Our gazetteer uses a database that contains geographical coordinates attached to address strings for Finland. For the rest of the world, we are using the Nominatim[15] project. Nominatim provides an interface to query OpenStreetMap data and has support for the four common operations we are using.

The speed of our gazetteer can be increased by using a database management system that supports Open Geographical Information System (OpenGIS) specifications, using database indexing or data types and functions that implement OpenGIS standards. Such database management systems include MySQL with spatial extensions, PostgreSQL with PostGIS or Oracle Spatial.

We implemented a postal address database that stores all the addresses in Finland using a MySQL5 database management system. For the design of the database the following factors were considered: the use of MySQL spatial extensions or common data types and the use of database indexing.

For testing purposes, a postal address database of the North Karelia region was created. ***Table 2-1*** shows the database sizes for the considered solutions.

Results show that indexing dramatically increases the database size with more than 90%, whilst using spatial extensions also increases the storage size with more than 12%.

Table 2-1 Database sizes for North Karelia region

Spatial extensions	Indexing	Data size (MB)	Index Size (MB)	Database Size (MB)
No	No	22.7	-	22.7
No	Yes	26.7	25.6	52.4
Yes	No	29.3	-	29.3
Yes	Yes	33.3	48.0	81.3

We tested the query execution times using a benchmark application that randomly chose 500 location points within our

[15] https://nominatim.openstreetmap.org/

region and tested the most common operations described in the previous section. We logged the execution times and we calculated the total time for query execution for the proposed testing scenario using the following formula:

$$T_{total} = T_{query1} + n_1 T_{query2} + n_1 n_2 T_{query3} + T_{query4} \qquad (2.1)$$

Above, n_1 represents the average number of municipalities returned by *query1* and n_2 represents the average number of search results.

The values of n_1=1.7 and n_2=64 were determined experimentally and used in the calculation of average query execution times, see ***Table 2-2***.

Table 2-2 Average query execution times

Spatial extensions	No	No	Yes	Yes
Indexing	No	Yes	No	Yes
Query 1 (s)	886	898	3956	176
Query 2 (s)	715	953	1091	173
Query 3 (s)	670	14	674	13
Query 4 (s)	887	909	3866	215
Total time (s)	75	5	83	2

Results show that the non-indexed solutions are at least 15 times slower comparing to the indexed ones, therefore using a non-indexed database is not justified. Using spatial extensions makes queries run significantly faster on the indexed solutions (at least twice as fast) and slower on the non-indexed solutions (1.09 times slower).

The most efficient solutions for the database are also the most storage-costly solutions. However, in our application, the execution time is more important that storage space. The most time-efficient solution, which is using spatial extensions and indexing, turned out to be the most storage-costly one.

2.6.3 Street Name Detection

Street name detection is the starting point of our address detection algorithm and it can be done either with or without the use of a gazetteer. The methods that do not use gazetteer usually assume that a street name has a certain structure, whilst the methods which use a gazetteer rely on fast word matching. For comparison, we implemented both approaches: a *heuristic method* that does not use a gazetteer and two text matching methods that use data extracted from a street name database.

2.6.3.1 Heuristic Method

Heuristic methods rely on regular expression matching. The structure of most addresses has certain particularities. For example, street names can start with the same prefix or end with the same suffix and they can be preceded or succeeded by standard words or numbers

According to our experiments, this approach has good results for Finnish street names because most of them end in words such as *katu* (street), *tie* (road), *kuja* (lane) or *polku* (path). The heuristic method has the advantage of not needing any predefined data structures to store the street names and it is reasonably fast.

Heuristic methods have the disadvantage of needing to be tailored for every country and language because of the various ways an address block can be constructed.

2.6.3.2 Brute-Force Matching Using Street-Name Arrays

A brute-force text matching method checks every word on a web page against a street name database. We use an optimized brute-force solution that checks the word against all street names that are close to the user's location, for example in the municipality where the user is located.

We use arrays of street names that are created beforehand from the gazetteer. Each array stores all the street names in a municipality. The search is done using language-specific functions and since our search engine is written using PHP scripts, we use the *array_search* and *in_array* functions.

2.6.3.3 Text Matching Using Street-Name Prefix Trees

We store street names into prefix trees, which are created beforehand using the data in our gazetteer. In general, the postal addresses are not unique and the same street name can be found in many cities. A prefix tree is therefore built for each municipality. Because the user's location is known beforehand, we limit the search to its area of interest and we load the prefix trees for the municipalities in that area.

Table 2-3 Gazetteer statistics

	Finland	**Singapore**
Number of municipalities	410	1
Total number of street names	92 572	573
Number of streets per municipality	474	573
Average street name length	11.6	6.1
Total size (MB)	74.4	0.18

Our research project is based in Finland and we additionally used Singaporean street data, as the addresses have a higher degree of variation than Finnish addresses, are written in English and were easily available to us when the experiments were performed. Statistical data about both gazetteers are detailed in ***Table 2-3***.

2.6.3.4 Street-Name Prefix Tree

The prefix tree (or *trie*) is a fast ordered tree data structure used for retrieval [NaRa02]. The prefix tree stores a collection of strings, indexed from the beginning of a word (i.e. prefix). The root node represents an empty string and its children store the first letter of each of the indexed strings, their children store the second letter and so on. The same principle is applied at every level of the tree so that the internal nodes describe all the sub-strings (prefix) of a particular string. The recursive version of the algorithm is presented in **Figure 2-8**.

```
ConstructTrie(streetnames)
  Create empty node root
  FOR i = 1 TO count(streetnames) DO
     AddString(root, streetnames[i], i);

AddString(node, string, index)
  IF (length(string) > 0) THEN
    IF (string[0] is not the key of a child of node)
    THEN
      Create new node child with the value string[0]
    ELSE
      Set child as the child of node with the key string[0]
      AddString(child, substring(string, 1), index)
  ELSE //node is a terminal node
    node.index = index;
```

Figure 2-8 Prefix tree generation pseudocode

The nodes of the prefix tree can also have values associated with them, although the only values that are commonly used are the values of the leaf nodes and some of the values of the inner nodes. In our case, we use the values to mark the end of a street name, which is usually a leaf node and more rarely an inner node, in the case when a street name is a prefix for another street name.

Dictionary search is one of the most common applications of the prefix tree. Searching a word in the prefix tree consists of traversing the prefix tree until a leaf node is reached, or until a node does not have any children whose key contains the desired letter. The recursive version of the prefix tree search algorithm is presented in ***Figure 2-9***.

```
FindString(root, string)
  IF (strlen(string) == 0) THEN
    RETURN root.index; //we have reached last node
  ELSE
    IF (string[0] is not the key of a child of root)
    THEN
      RETURN -1; //string is not found
    ELSE
      Set child as the child of root with the key string[0]
      RETURN FindString(child, substring(string,1))
```

Figure 2-9 Pseudocode of the recursive prefix tree search

In our implementation, we create a prefix tree for the street names of each municipality. Therefore, the street name detection becomes a dictionary search using a prefix tree. Because the Finnish street names usually end with a limited number of suffixes, the names were introduced in the prefix tree in reverse order and the search is done starting from the last letter. ***Figure 2-10*** gives an example of a pre-computed prefix tree.

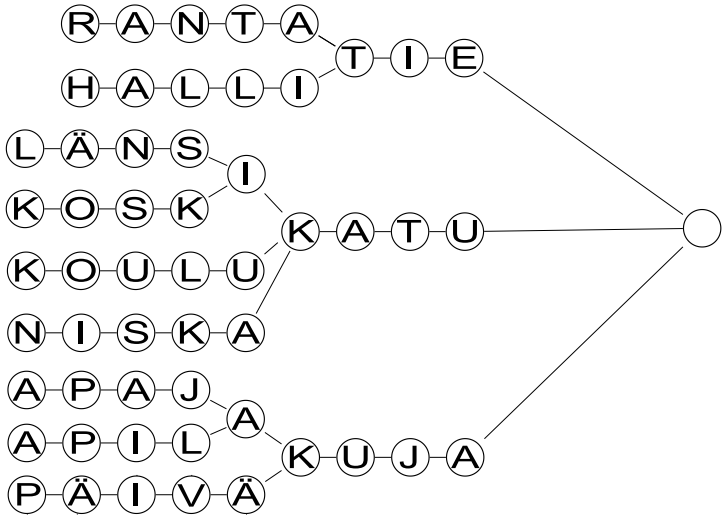


Figure 2-10 A sample prefix tree built from street names

Table 2-4 summarizes the prefix trees we computed for Finland and Singapore. Using prefix trees or other pre-built data structures to access street name data from the gazetteer decreases the needed storage space, in our case from 3 GB to 74 MB. We are still using the gazetteer for address validation and geo-coding.

Table 2-4 Prefix tree statistics

	Finland	**Singapore**
Maximum tree depth	34	14
Average tree depth	12.7	7.4
Average tree width	105	167
Average number of nodes per tree	2338	2335
Total size (MB)	74.4	0.18

2.6.4 Experiments

We tested the proposed method in the MOPSI location-based search engine using 20 different search locations and 10 keywords in English to construct <*keyword*, *city*> queries. We downloaded the content of the first 10 search results for each query of Google's search engine and the downloaded content was used as data input for the MOPSI prototype.

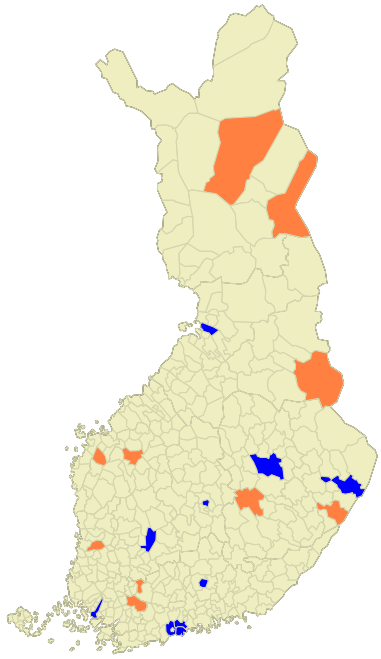


Figure 2-11 Locations used for experiments. The urban locations (blue): Espoo, Helsinki, Joensuu, Jyväskylä, Kuopio, Lahti, Oulu, Tampere, Turku, Vantaa; the rural locations (orange): Forssa, Kitee, Kuhmo, Laihia, Lapua, Pieksämäki, Salla, Sodankylä, Somero, Ulvila

The search locations were divided into two groups: 10 rural 10 urban municipalities (***Figure 2-11***), and the test keywords were divided into five commercial and five non-commercial ones (***Table 2-5***).

Table 2-5 Keywords used for experiments

Commercial	hotel, restaurant, pizzeria, cinema, car repair
Non-commercial	hospital, museum, police station, swimming hall, church

The addresses detected by each method were validated using our gazetteer database. The size of the downloaded data in the rural and urban municipalities is 13.9 and 11.2 MB, respectively.

Table 2-6 shows the average time for address detection and the number of detected addresses for the considered municipalities. The average time is calculated per query over all searches. According to the results, the proposed prefix tree method is considerably faster than our brute force method, and two to three times faster than our heuristic approach, which does not use the gazetteer. Typical search times of the prefix tree are less than one second per query.

Table 2-6 Average search times for the address detection

Method	Time (s)	Standard deviation	Number of validated addresses
	Rural municipalities		
Brute Force	3.01	2.43	3.7
Heuristic	1.54	1.15	2.5
Prefix Tree	0.51	0.35	3.7
	Urban Municipalities		
Brute Force	10.18	7.11	19.8
Heuristic	1.70	1.24	18.6
Prefix Tree	0.87	0.85	19.8
	Total		
Brute Force	6.59	6.40	11.8
Heuristic	1.62	1.20	10.5
Prefix Tree	0.69	0.68	11.8

The results also show that street density and city size do not affect the speed of the heuristic solution. In the prefix tree method, the average search time is slightly longer in urban municipalities (0.51 vs. 0.87 seconds). Street density affects the brute force method most because the search in street arrays is slower than in the prefix trees, resulting in more than three times slower search times in urban areas.

The detected addresses are validated using our gazetteer. The accuracy (number of validated addresses) is higher for the brute

force and prefix tree methods than for the heuristic method. This is because some of the street names do not follow our rules, having either unusual suffixes or multiple word names. The biggest difference between urban and rural municipalities is that the number and the density of streets are much larger in the urban municipalities and, therefore, the methods using gazetteer (prefix tree and brute force) are slower in rural municipalities. Nevertheless, the prefix tree method is the fastest even in this case.

In total, the proposed prefix tree method is on average twice as fast and 10% more accurate than the heuristic method. It reaches the same accuracy than the brute force search but uses only 10% of the processing time.

Our main goal in designing a gazetteer-based street address detector was to increase the accuracy in comparison to the fast heuristic method that was used in the earlier implementation [FKTS10]. This goal was achieved, as the proposed prefix tree solution is 57% faster and 10% more accurate, on average, than the heuristic solution. In comparison to brute force, it is 10 times faster. The resulting solution improves the speed and quality of web-page geo-referencing and removes one bottleneck for creating an efficient location-based search engine as the prototype *MOPSI search.*

2.7 EXTRACTING ASSOCIATED INFORMATION

We complement the search results by extracting a title and a representative image for each location we detect (see ***Figure 2-12***). We consider as title the name of the place or the service that is referenced by the address we detect. We consider the representative image to be the image that best describes the detected service. The location, along with its title, URL and representative image, forms a search result within our application.

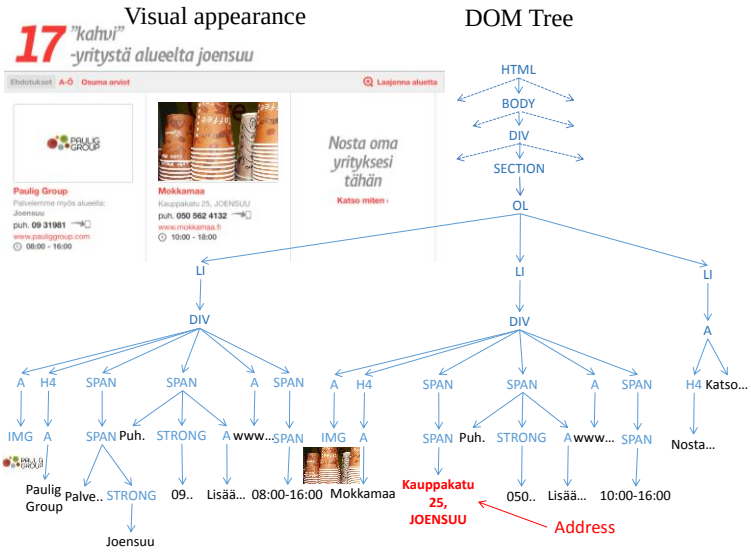


Figure 2-12 DOM tree of a website with location information. The text node that contains the location is marked in red

2.7.1 Extracting Representative Image

Images are used on web pages more than any other type of online content because they can transfer information to the user in a quick and efficient way. Although a large number of images are embedded into web pages, many of them are less relevant to the content of the web page, such as advertisements, navigational banners, icons and images that serve as section headings. A solution is needed to ignore the irrelevant images and find one representative image for the web page.

We define the representative image of a web page as the image that best represents the content of the page to the user. Representative images are important in many applications, especially in cases when bandwidth limitation restricts the total number of images that can be retrieved, or when building a visual category in which a single image must represent an entire category of documents and their associated content. This section presents the work that is detailed in [P5].

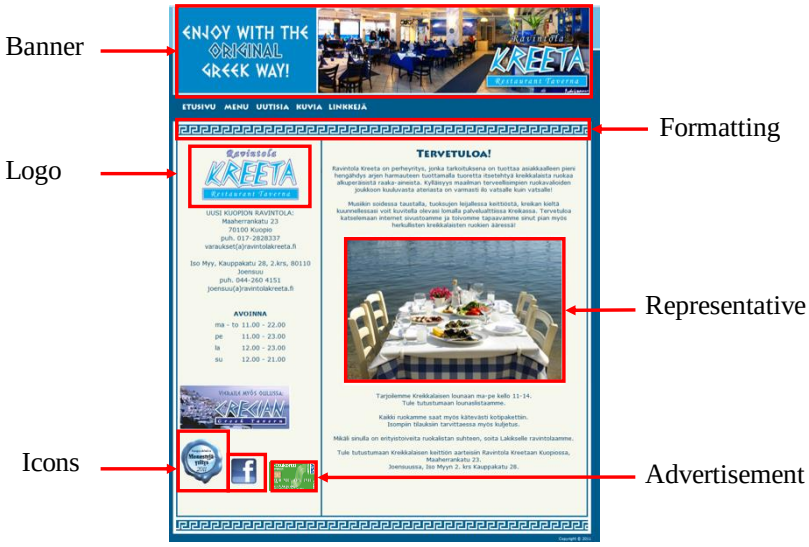


Figure 2-13 Image categories

We define five image categories based on the usage of the image within the web page (see ***Figure 2-13***) and rank them in the following priority order:

- *Representative*: images directly related to the content or the topic of the web page.
- *Logos*: recognizable images used to identify the company or institution that owns the website.
- *Banners*: images placed on a web page, either above, below or on the sides of the content. They are generally used for decoration. Headers and footers are classified in this category.
- *Advertisements*: images promoting products or services that are irrelevant to the topic or the content of the web page.
- *Formatting* and icons: images used to enhance the web page's visual appearance such as spacers, bullets, borders, backgrounds, or pictures used purely for decoration. We also include the small images which are not classified as logos and serve a functional purpose, such as icons which link to the home page or icons which are used for changing language.

src	http://www.ravintolakreeta.fi///images/banner.jpg
alt	--
title	--
from	css
format	jpg
width	945
height	202
size	190,890 px
aspect ratio	4.67
parent tag	
class	header

7

Figure 2-14 Image Features used

We retrieve the list of images from a website, by parsing its HTML, CSS and JavaScript source code. After retrieving the list of images, we analyze the following features of each image and its corresponding HTML element: URL, size, aspect ratio, HTML tag of the parent element, class and ID of the <img> tag and of the parent element (see ***Figure 2-14***).

Table 2-7 Rules used for image categorization

Category	Features	Keywords
Representative	Not in other category	
Logo	Parent is h1 or h2	logo
Banner	ratio > 1.8	banner, header, footer, button
Advertisements		free, adserver, now, buy, join, click, affiliate, adv, hits, counter
Formatting and icons	width < 100px height < 100px	background, bg, sprite

We are using the rules in ***Table 2-7***. In all categories, a predefined set of keywords is used. If any of these are found in the image URL or in the class name of the <img> tag and of the parent element, then the image is assigned to that category. In

addition to this, Banners and Formatting are also categorized according to image features such as image size and aspect ratio. Our rules were derived from direct observation, as, for example, banners usually have high aspect ratio, formatting and icon images are small, and logos, banners and advertisements have specific keywords in their class names.

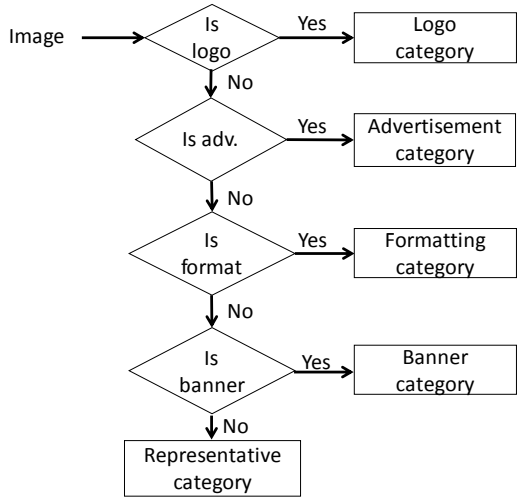


Figure 2-15 Decision tree used for image categorization

Note that the categories are overlapping, so that the same image may meet the requirements of multiple categories. However, in this case we are using a decision tree to assign the image category (see ***Figure 2-15***). An image can belong to the class of representatives only if it does not belong to any other category.

After image categorization, we rank the images within each category according to their size: the bigger the image size in pixels, the higher its rank. The categories are prioritized as shown in the beginning of this section. Neither HTML, CSS or JavaScript are prioritized, and the images in these file types are treated equally. We choose the representative image as the image with the highest rank.

2.7.2 Extracting Service Names

In order to extract the service names, we are using the DOM tree of the web page and we are splitting it into sub-trees until each

sub-tree contains just one postal address node. For each postal address node, we are scoring all the text nodes in their respective sub-trees. This section presents the work that is detailed in [GaTF15].

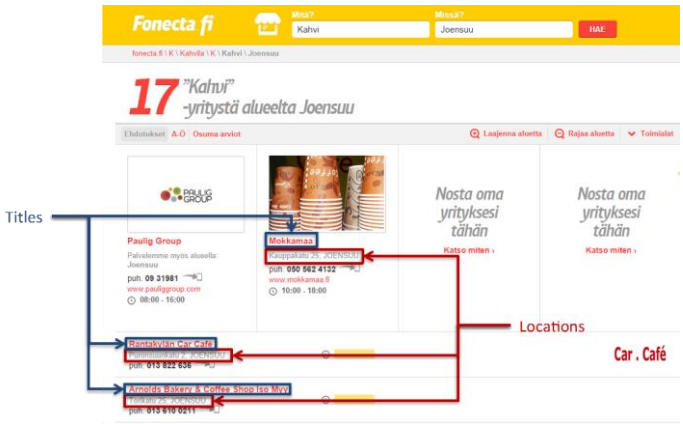


Figure 2-16 Web page that include several titles and addresses

In ***Figure 2-16***, there are several service titles located in the content of the web page that need to be detected, including "*Kauppakatu 25, Joensuu*". Usually service titles are accompanied by other information such as location, telephone numbers and opening hours. After the address has been detected, the service title is needed in order to generate a search result. We use the DOM tree of the web page and split it into sub-trees until each sub-tree contains only one postal address.

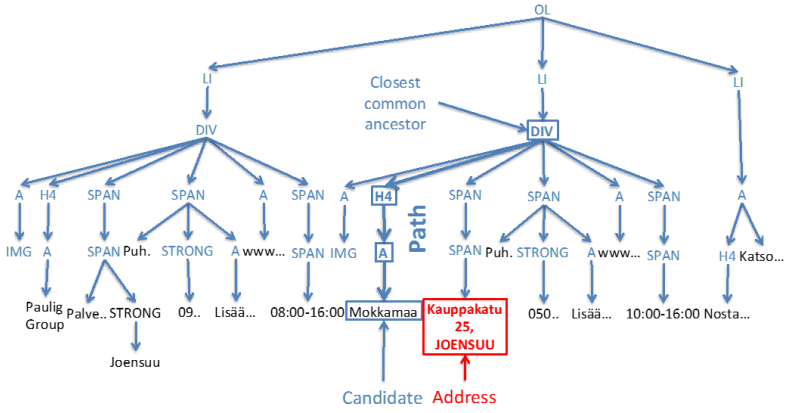


Figure 2-17 DOM sub-tree that contains the address of a service

An example of such a sub-tree is in **Figure 2-17**. For each sub-tree that contains one postal address, we score its text nodes based on visual appearance and distance to the postal address. Finally, we choose the node with the highest score (see **Figure 2-18** and ***Figure 2-19***).

Table 2-8 Scoring HTML tags

Tags	Score	Tags	Score
H1	+7	H5, H6, B, STRONG	+3
H2	+6	I, EM	+2
H3	+5	Others	0
H4, A	+4		

In order to score a text node, we first find its closest common ancestor to the postal address node. For example, in **Figure 2-17**, the detected address "*Kauppakatu 25, Joensuu*" and the candidate text node "*Mokkamaa*" have a DIV as the closest common ancestor. For calculating the visual appearance score, we first assign a score for each intermediary HTML tag along the path between the considered text node and the closest common ancestor node (in our example two SPAN tags) according to ***Table 2-8***.

Furthermore, we parse the CSS style sheet of the text node and we score the differences to the style sheet of the postal address using the following attributes: *color*, *background-color*, *font-size*, *font-weight*, *text-transform* (see ***Table 2-9***.). The perceptual color difference between the *color and background-color* attributes of a text node and the postal address node is scored from 0 to 10 by calculating the Delta E (CIE 2000) difference [LuCR01] and normalizing it to [0, 10].

Table 2-9 Scoring CSS attributes

CSS Attributes	Score
color, background-color	+ perceptual color difference (0 to 10)
font-size	+ (node font size - address node font size)
font-weight	+3 if bold or >500
text-transform	+5 if uppercase

The visual appearance score is calculated by summing the scores for HTML tags and the CSS style sheet differences:

$$S_A = S_{HTML} + S_{CSS} \quad (2.2)$$

Above, S_A is the appearance score, S_{HTML} is the score for an intermediary HTML tag, and S_{CSS} is the score for a CSS attribute difference.

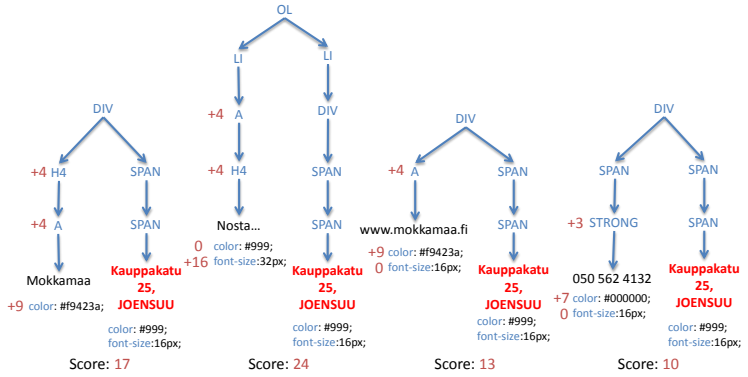


Figure 2-18 Scoring based on visual appearance

The scoring based on visual appearance for our example is detailed in **Figure 2-18** where we chose the four strongest candidates for the detected address, along with the subtree of the closest common ancestor. The candidate with the highest score, "*Nosta…*", is visually the most different from the address node, because the font size is significantly larger. It is also highlighted as a header and has a link. The second candidate, "*Mokkamaa*", is the actual title of the service. It is also highlighted as a header, has a link, and the text color is significantly different, but the font size is the same as that of the address node and its score is therefore lower.

We score the text nodes based on distance to the postal address by using a distance penalty. We use the DOM tree to determine the number of tags along the path from the address node to the closest common ancestor node (see ***Figure 2-19***). We divide the visual appearance score by this distance. We introduce this distance penalty because the nodes that are closer to the

postal address are more likely to contain information related to that address. Using the visual appearance score (2.2), the final score for a node becomes:

$$S_{NODE} = \frac{S_{HTML}+S_{CSS}}{d} \quad (2.3)$$

Above, d is the number of tags along the path from the postal address node to the closest common ancestor node.

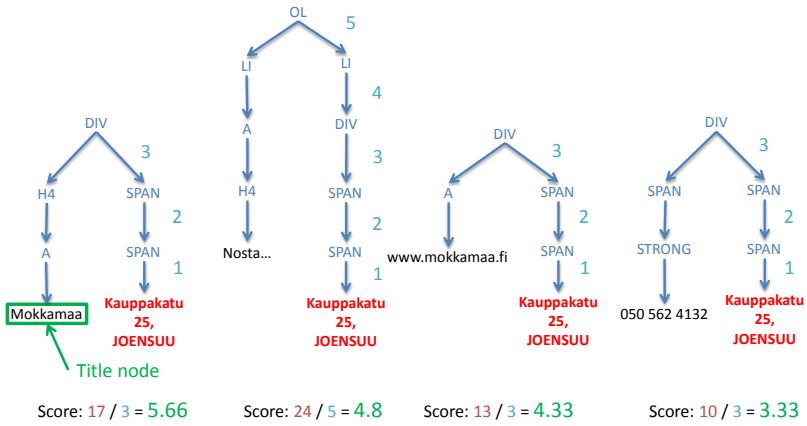


Figure 2-19 Adjusting scores according to distance penalty

Finally, we choose the text node with the highest score as the title of the service (see ***Figure 2-19***). In **Figure 2-18**, because of the distance penalty, the node with the highest visual score "Nosta…" has a smaller score that the node that is closer to the address "*Mokkamaa*".

2.8 EXPERIMENTS

In order to test our prototype application, we simulated the following scenario: the user is in the center of an urban or rural municipality and his search is restricted to that municipality. His targets can be commercial (i.e. restaurant) or non-commercial (i.e. police station).

These tests were performed in 2010 using the prototype described in [FTKH10]. We selected 10 Finnish urban municipalities and 10 Finnish rural municipalities (see ***Figure***

2-11). The urban municipalities were selected according to their size, and geographical location was taking into account when selecting the rural municipalities. The search was performed using 10 English test keywords which were divided into five commercial ones (hotel, restaurant, pizzeria, cinema, car repair) and five non-commercial ones (hospital, museum, police station, swimming hall, church), see ***Table 2-5***.

In addition, the same queries were submitted to two other location based web services: Google Maps and the Finnish Yellow Pages. According to [EgPa10], Google Maps uses multiple sources to provide the results. For instance, information submitted by local business owners or public directories, enhanced content such as reviews, photos submitted by users of various services, user generated content and other websites are crawled. Yellow Pages, on the other hand, is a public directory of local businesses and the data are maintained and verified by users and administrators.

The search results are sorted by distance and filtered according to each municipality. Duplicates were manually removed and the distance was calculated using the gazetteer. There is no standard methodology for testing the overall relevance of the location-based search results that considers the relevance of both topic and distance. We therefore formulated our own criterion based on the evaluation methodology proposed in [JaMo06]. First, we compare the total number of search results from the tested services without considering their relevance. Secondly, we compare the relevance of the search results by evaluating them as (1) *relevant*, (2) *somewhat relevant* or (3) *not relevant*.

As shown in ***Table 2-10***, our prototype provides more results than Google Maps or Yellow Pages, and is slightly more relevant (the smaller the better) than Google Maps, but less relevant than Yellow Pages. With the exception of urban non-commercial queries, our prototype gives more results. Unsurprisingly, Google Maps has more results than Yellow Pages for non-commercial queries and less for commercial ones.

Table 2-10 Number of results for our test scenario

Query type	MOPSI prototype	Google Maps	Yellow Pages
Rural non-commercial	69	29	0
Rural commercial	245	92	189
Urban non-commercial	148	413	37
Urban commercial	1412	813	1337
Total number of results	2352	1405	1597
Overall mean relevancy	1.59	1.66	1.28
Overall std. deviation	0.84	0.89	0.54
Overall std. error	0.02	0.02	0.01

Contrary to Google Maps and Yellow Pages, our prototype relies on the relevance of the results of the external search engine. The search results are basically the addresses found in the results of the external search engine, whilst Google Maps has multiple criteria for relevance and Yellow Pages is controlled by human evaluators. We therefore compare the relevance of the search engines on an average basis, using a combination of two keyword categories for each comparison. A comparison of rural municipals and non-commercial keywords resulted in our search engine (MOPSI) having a mean value less than that of Google Maps (2.35 comparing to 2.48), whilst Yellow Pages did not return any results. This indicates a relatively high number of relevant results obtained by our search engine. The same method was used in **Table 2-11**, in which the results show a diverse number of relevant links from the MOPSI search.

Table 2-11 Mean relevance for our test scenario

Query type	MOPSI prototype	Google Maps	Yellow Pages
Rural non-commercial	2.35	2.48	0
Rural commercial	1.71	1.33	1.36
Urban non-commercial	2.20	2.17	1.59
Urban commercial	1.46	1.41	1.27
Overall mean relevancy	1.59	1.66	1.28

The results show that the relevance of the MOPSI search engine is close to Google Maps, except for the rural municipalities with non-commercial keywords, for which we get a higher number of results, but their overall relevance is smaller. Yellow Pages is the most relevant service, but has the lowest number of results (except for commercial urban keywords). In urban areas, the number of results by the MOPSI search engine is close to Google Maps and Yellow Pages, whilst in rural areas it is higher.

2.8.1 Observations and known problems

One of the main problems is that the search can produce a vast number of irrelevant results. Mobile devices have restricted resources (data transfer bandwidth, small display) and often poor usability. The application should therefore be able to filter out flawed or less relevant data much better than a web-based application to make the browsing of the search results easier. Because of this, the current version downloads only a limited amount of links but further ideas to improve this would be desirable.

Another problem is that, unlike normal web searches, we allow the results to include parts of web pages. This is very useful for finding services that do not have their own web page but exist in ad-hoc service directories such as www.pizza-online.fi. However, an open problem is how to extract only the relevant part from a web page without any prior knowledge about its structure.

Despite the previously mentioned problems, many positive results are also achieved. When it comes to non-commercial services, web pages are good repositories of location relevant information. In the rural areas of Finland where businesses are small, more services are found from web pages than from commercial databases.

2.8.2 Conclusions

The concept of a location-based search engine was outlined, and its design issues and problems were discussed. The MOPSI prototype implementation in Finland was demonstrated with qualitative comparison using typical search examples. The idea itself was also generalized worldwide, although the practical implementation has some issues such as the accuracy of the gazetteer and the detection of the address elements. The chosen address detection method (especially the street name detection), relies on the accuracy of the OpenStreetMap gazetteer. It needs to find the exact string match, and can support multi-word street names or addresses as the street names are indexed as strings that can also contain spaces, but it does not support any variation on the order of the words in a street name, mistyped words, or abbreviations. Methods such as term normalization [AhBo08a] should be considered in future work to improve address detection, but they also require changes in querying the OpenStreetMap gazetteer.

The results indicate that the proposed approach has much potential for practical applications. Most of the problems are related to technical matters and implementation issues. For instance, we use real-time search and page parsing, whereas commercial solutions such as Google can use large computer capacities and avoid computational problems by pre-processing, huge storage (cache), and indexing.

3 Location-Based Mobile Orienteering Game

Combining gaming and physical activity has always been a desirable goal. Furthermore, using mobile games instead of "real world" applications is better for training people in using new technologies. This is because mobile games provide a virtual environment in which users are encouraged to experiment.

One the one hand, using use of mobile devices for playing games is a very popular and long-established idea. On the other hand, mobile gaming that uses the player's location and involves physical activity is a recent idea, although it is starting to gain popularity. With the fast development of computing devices, mobile location-based gaming has evolved from using wearable computers and head-mounted displays [PiTh02] to using mobile phones with access to the Internet and GPS positioning.

We introduce a mobile location-aware game, O-Mopsi, which is based on the classical concept of orienteering and exploits the data available in a geo-tagged user generated photo collection, available at http://cs.uef.fi/mopsi/photos. Users can create games for others to play by defining a set of targets. The players need to visit all the targets in order to complete the game. O-Mopsi was first introduced in [P6]; a more detailed description of the game and its components is documented in [Wan14].

The game can be played using a mobile application that is available for all the modern mobile operating systems (Symbian, Windows Phone, Android and iOS). Players are free to choose how to play the game, regardless whether they are walking and running or using different transportation modes such as a bicycle.

O-Mopsi is available at http://cs.uef.fi/o-mopsi. The mobile client has functionalities such as plotting the targets' photos on the map, displaying compass data and modifying sound

frequency and pitch to indicate the distance to the selected target. The web interface allows game management, real-time player tracking, post-game trail analysis and suggests reference routing by using either greedy heuristics or an ant colony-based optimization.

3.1 RELATED WORK

One of the first examples of location-aware games is *Pirates!* [BFHL01], which shows an example how the player's physical location is integrated into the computer game's design and is used to trigger game events. A large number of location-aware games are adaptations of traditional board games or computer games, including *Quake* [PiTh02], *Pacman* [CGLF04], *Tic-Tac-Toe* [ScKM05], *Monopoly* [LiOO08] *Chase and Catch* [MHKT09] and *Snake* [ChSi12]. The extension of video games from the traditional virtual world to real locations using mobile devices is also discussed in [RMCE06]. Other approaches combine mobile gaming with education and game reality with physical reality.

Savannah, [FJSR04], is a mobile game based on animal behavior and aimed at children, who use GPS enabled devices to navigate in a virtual savannah. The children can observe animals and must learn how to survive. Education is also the goal of *Skattjakt* (*Treasure Hunt*), [SpMi08], which shows that mobile outdoor games are well suited for novel learning activities that involve physical motion, problem solving, inquiry and collaboration. *Treasure Hunt* encourages players to get physically active by solving a mystery surrounding a castle located on the university's campus. Another approach to education and treasure hunting is the game *Tidy City* [WeBO12], which requires solving location-based riddles and city exploration.

"*Song of the North*" [LHNL04] is a game inspired by Nordic mythology that combines game reality and physical reality. The physical world is combined with a virtual spirit world, in which the player can interact by using a mobile device. Another example of mixed reality is "*Can you see me now?"* [BCFD06], in which players are chased by runners through a virtual model of

a city. The runners are professional performers equipped with Wi-Fi and GPS. The rules are similar with the game of catch, with the difference that the "runners" have to traverse actual city streets in order to capture the online (virtual) players.

In a location-aware game, photos can be used in various ways, such as a means of interaction [SuKo06], to provide additional information about game targets, such as O-Mopsi and in other games such as *See It* [NeJu12] or to produce useful information for other application, such as *CityExplorer* [MMSK08].

CityExplorer applies the "games with a purpose" paradigm to users with mobile devices that have to take geo-referenced photos, identify points of interest and categorize them semantically. O-Mopsi uses the location-based data that is produced by other applications, but also encourages users to create games with a purpose by offering the possibility to create sightseeing tours.

O-Mopsi can also be viewed as an exergame, which are video games that require physical activity to play. According to [BoYa13], there are several categories of exergames: mobile location-based games, mobile location-aware fitness and sports applications with social and games features, and location-aware sports gadgets. O-Mopsi belongs to the first category.

3.2 GAME RULES

The goal of the game is to visit a set of targets in the shortest time. A target is identified by the following attributes: location, photo and a short description. A game can be created by selecting photos from the MOPSI photo collection using our web interface.

A game starts when the player visits the first target. The order of targets is not fixed and the player can freely choose which targets to visit. The game ends when the player has visited all the targets. To visit a target, a player has to be closer than 20 meters from the target's location. This threshold was chosen taking into consideration GPS inaccuracies that can occur either when collecting geo-tagged photos or when playing a game.

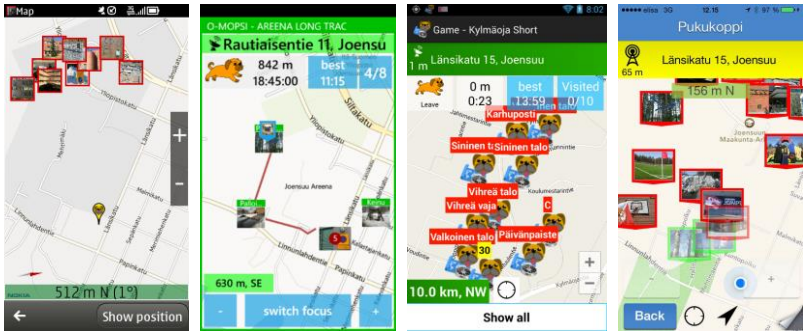


Figure 3-1 Game screen in the 4 supported mobile platforms: Symbian, Windows Phone, Android and iOS

Game results and players' progress can be viewed online in real time using the web interface, which also includes tools for game analysis such as calculating the shortest path. Players are ranked in the order of the completion time. The total distance, the starting target or the order of visiting targets do not affect ranking.

The game shares similarity with the concepts of orienteering and geocaching [Came04], which both require identifying and visiting a number of targets.

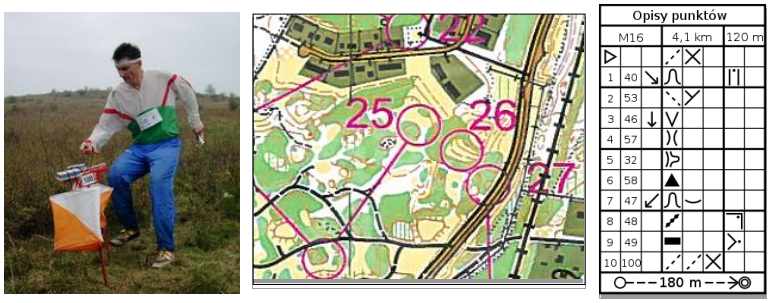


Figure 3-2 Orienteering: control point (left), map (middle) and control description sheet

Orienteering requires navigating from point to point in a predefined order using a map and a compass. Unlike orienteering, in O-Mopsi the targets can be visited in any order, encouraging players to develop different strategies and to choose different starting points. Furthermore, in order to make gameplay easier, O-Mopsi uses digital maps and GPS so that the player can

visualize his position on the map while playing, features which are not available in orienteering.

Figure 3-3 A classic geocache box (source: https://en.wikipedia.org/wiki/Geocaching#/media/File:Classic_Geocache.jpg)

Geocaching requires finding a "hidden treasure" (a collection of things placed in a container called geocache and placed in a location available to the public). The GPS location of the cache is published online and other players need to find the cache and replace an item from the collection with another. O-Mopsi also requires finding a certain GPS location defined by another player, but the location is additionally identified by a photo. Furthermore, in Geocaching, the time does not matter and targets are not grouped into games or other entities.

O-Mopsi uses virtual targets, whereas orienteering and geocaching targets are physical objects.

3.3 WEB INTERFACE

The O-Mopsi web interface can be used for creating and managing a game, displaying the proposed shortest path of a game, displaying game results and viewing real-time player progress.

The architecture of the website, along with the main functions, is presented in ***Figure 3-4***. For displaying games and targets we

use Google Maps, the web interface is developed using PHP and JavaScript. We store game data in a MySQL database with spatial extensions.

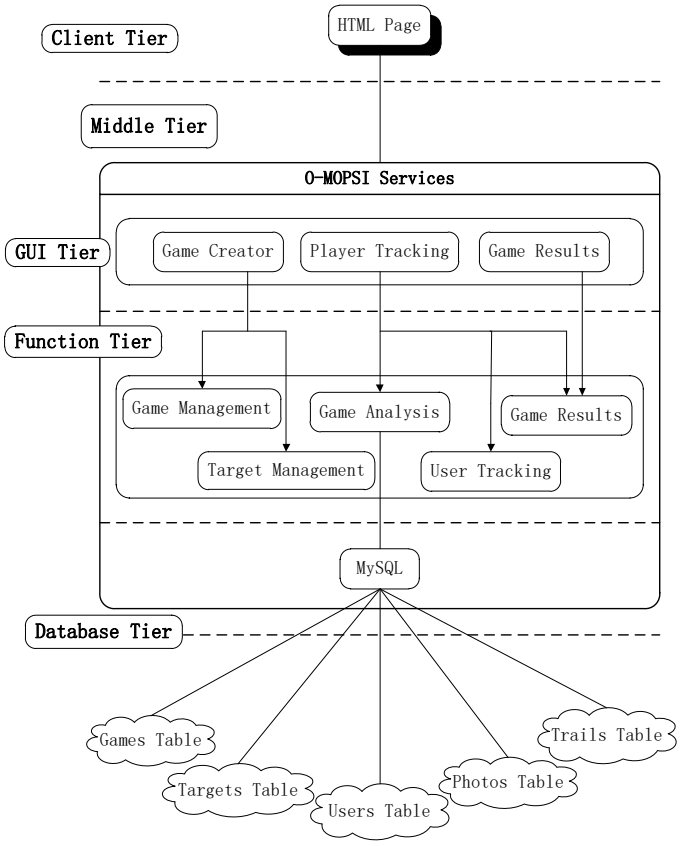


Figure 3-4 Architecture of the O-Mopsi website

The client tier contains the HTML page that is presented to the user. The middle tier is composed of the Graphical User Interface (GUI) tier and the function tier. The GUI tier includes interfaces for the creation of the game, for the live tracking of the players and for the game results. The function tier includes all the functions needed for the server logic: game and target management, game analysis, user tracking and game results. Finally, the database tier consists of the tables needed to store game data (games, targets, users, photos and trails).

Using the web interface, player can manage games with operations such as creating, editing and deleting. An example of such game is shown in ***Figure 3-5***.

Figure 3-5 Sightseeing tour game in Singapore

A typical workflow of creating and editing a game is documented in ***Figure 3-6***, where a user can create a new game and edit or delete games.

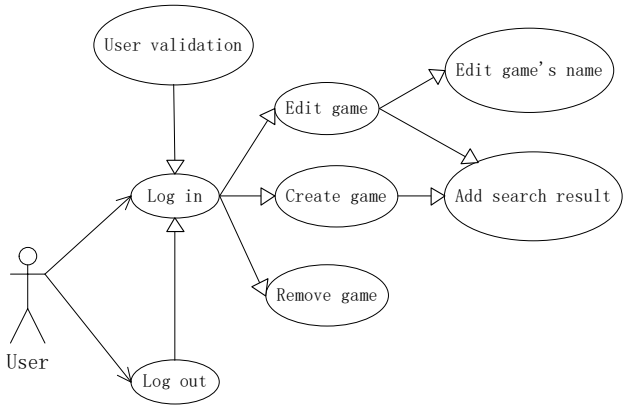


Figure 3-6 Typical workflow for game management

For a game to be playable, the quality of targets is important. Therefore, we created a set of guidelines for creators to follow:

- Outdoor targets only: Targets can be small, such as a postbox, logo on a door or small detail on a building or big, such as a shop, a bridge or another easy to find place. Selecting people, animals and indoor objects as targets is not recommended.
- Reachable and clear target: Players must have a certain object or location to look for. A photo of a large building that is taken from distance could not clearly indicate the place that needs to be visited by the player. Smaller items such as entrances are easier to identify. Also, targets in places that are temporarily open to the public, such as stadiums, are discouraged.
- Accuracy of location: In our photo collection, there are photos which are taken far away from the object and photos that have inaccurate locations caused by GPS errors. This is why we offer the player the option to fine-tune the location on the map.
- Permanent targets: Some targets, such as vegetation, piles of snow, advertisement signs are temporary or season-dependent and we do not recommend choosing them.
- Distance between the targets: If targets are too far from each other, the player will lose interest in finding them. According to the feedback we got from SciFest, most of the players give comments that finding targets is the most interesting thing during the game, but there were some cases when players gave up the game because one of the targets was too far away from the other ones.
- Number of targets: A game should have at least three targets. The game creator can choose as many targets as he or she wishes and can use our analysis tool to check the reference route length. However, choosing a large number of targets could discourage inexperienced players.
- Short but descriptive names for targets: The map view of the mobile screen cannot accommodate long names. We recommend short and descriptive names for targets that should give a hint to the players to locate them. Names "Target 1", "No.1" and empty names must be avoided.

- Descriptive name for the game: A suitable game name can attract more players. For example, game names can be: *Postbox maniac*, *Joensuu Cafes* or *Niinivaara sightseeing*, which give a clear description of the purpose of the game.
- Language: Currently we do not support localization, therefore English is preferred, but the local language can also be used.

3.4 GAME CLIENT

The game client is available for most modern mobile operating systems. After registering or logging in, the player can choose to join a new game or continue an existing active game.

During gameplay, the player's location is tracked and stored on the server. The main screen of the application (see ***Figure 3-7***) shows the current location, accuracy and statistics such as playing time, distance and speed. It also contains shortcuts to the map, an option to highlight the closest target, to view the full list of targets and the game results (see ***Figure 3-8***).

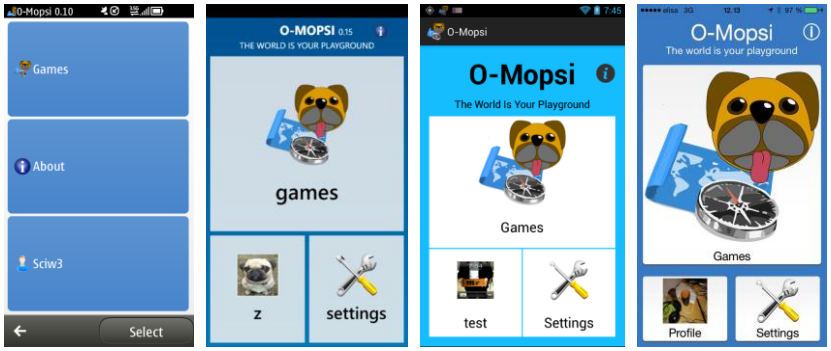


Figure 3-7 The start screen of the application

A target can be identified by its photo, which can be highlighted on the map (see ***Figure 3-9***). To aid navigation, the application displays the distance and bearing to the selected target along with player's orientation taken from the phone's compass sensor.

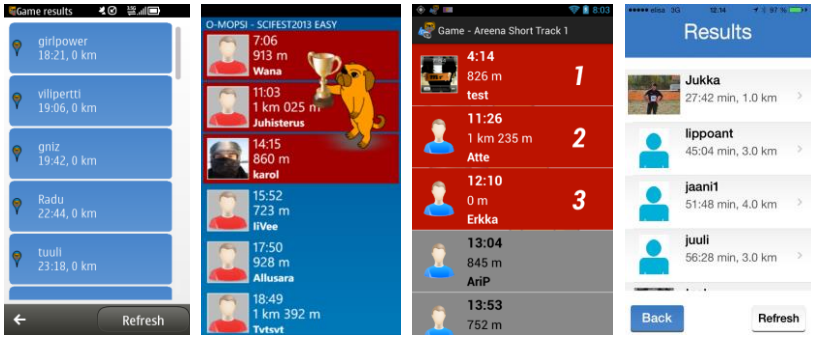


Figure 3-8 Game results screen

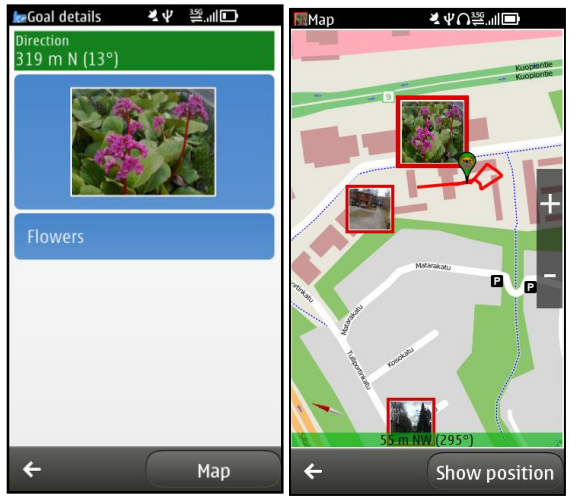


Figure 3-9 Viewing target details (left) and highlighting the chosen target on the map, along with the player's trail (right).

Additionally, the player is guided by sounds while the map is open. When the player approaches a target and is closer than 500m, a beeping sound is played at fixed intervals. The interval between sounds is inversely proportional to the distance, starting from 5 seconds for 500 meters and decreasing by 1 second every 100m (see ***Table 3-1***). The sound frequency also increases or decreases as the player becomes closer or further away from the target. Visiting the target turns off the sound guidance. We implemented this option in order to address an important issue in location-based games: the player's safety. We tried to minimize

the time the player has to look at the phone screen while he is heading for the selected target.

Table 3-1 The sound interval for different distance levels

Distance	Sound interval	Distance	Sound interval
> 500 m	no sound	100-200 m	2 s
400-500 m	5 s	60-100 m	1 s
300-400 m	4 s	20-60 m	0.5 s
200-300 m	3 s	< 20 m	"found" sound

3.5 FEEDBACK

O-Mopsi is mainly designed for the SciFest festival (www.scifest.fi), which is an annual international festival in which thousands of school kids, high school students, and teachers discover new experiences and learn about science, technology and the environment [JoKo10]. SciFest is organized in the city of Joensuu, Finland in the month of April.

O-Mopsi was presented during the 2011–2014 editions of the festival. Because of the limited availability of mobile phones with GPS and data connection, the players were organized into teams. After the game, the playing teams completed a short feedback survey.

Table 3-2 Players' ratings of the game

Feedback	Very good	Good	Needs improvement	Bad
Scifest 2011	3	6	0	0
Scifest 2012	3	7	0	2
Scifest 2013	2	21	6	0
Scifest 2014	8	19	3	0
Scifest 2015	9	9	1	0
Total	25	62	10	2

Feedback (see ***Table 3-2***) shows that the players mostly rated the game as being good or very good. According to the users, game rules are easy to understand and playing the game is enjoyable. Negative feedback was caused by software problems, or in some cases by problems in finding a GPS signal.

3.6 CONCLUSIONS

We presented a mobile location-aware game that is based on the classical concept of orienteering and the data available in a geo-tagged user-generated photo collection. The game can be played using a mobile application available for most of the current mobile operating systems, offering functionalities such as plotting the target photos on the map, displaying compass data, and modifying sound frequency and pitch to indicate the distance to the selected target. The web interface allows game management, real-time player tracking, post-game trail analysis and suggests game solutions. Testing O-Mopsi in a real-world situation during an international festival produced positive feedback.

4 Summary of the Contributions

All our proposed methods have been implemented in the MOPSI framework, which uses real data collected by means of our mobile applications.

In **[P1]** we define four aspects of relevance in sharing location-based media: location, time, content and social network. We study how these aspects of relevance appear in three media sharing platforms: Picasa, Flickr and MOPSI. Experiments show that the location is an important aspect, but not the only one. These results have been used for developing a location-based search and recommendation system.

In **[P2]** we propose a location-aware search engine for the mobile environment. We use a meta-search approach: we retrieve the results of a search engine that is not location-aware and post-process them by detecting locations and associated information. This paper sketches the overall scheme of this MOPSI search engine prototype for Finland, and defines all the core elements needed to make it happen. For rural commercial websites it provides better results than Google Maps and Yellow Pages. Since then, the MOPSI application has been extended in four different platforms that include the search engine component: Nokia Symbian, Windows phone, Android and iOS.

In **[P3]**, we describe the address detection used in the MOPSI engine. We find locations by detecting postal addresses using prefix trees and a gazetteer. Address detections start with street name detections, which are aggregated with other address elements in order to build address candidates that are validated using our gazetteer. We study both heuristic and gazetteer-based methods to identify street names, showing that a gazetteer-based prefix tree search method is the fastest and most accurate. This

method follows the general workflow of most of the state-of-the-art address detection methods, but it is different because it uses portions of the DOM tree and does not impose a certain order of elements, which helps generalization.

[P4] improves the methods in **[P2]** and **[P3]** by replacing plain text extraction with the processing of the DOM representation, and by improving the methods for extracting the title and representative image for each search result. We integrate processes of geo-referencing, geocoding and geo-tagging into a unified location-based system that is able to provide relevant and close information the user. The implementation is not limited to a specific geographic location, although it works for addresses that follow a structure similar to Western European addresses and the accuracy is influenced by the quality of the gazetteer and the string matching method we use.

In **[P5]** we study how to select a representative image to a web page. This solution is needed as part of the summarization of the web page found by the MOPSI search. Despite it being n use, also on social media sites, no good solutions were found in literature. The closest examples can be found in Facebook and Google+, which use simple heuristic solutions for selecting the image with the web link that the user wants to share. We propose a rule-based method that provides 64% accuracy, which is better than that of Google+ (48%) and Facebook (39%). Besides summarizing the MOPSI search results, the proposed method is directly applicable to those two social media sites as well.

In **[P6]**, we propose a location-aware mobile game called O-MOPSI that promotes physical exercise by applying concepts from the classical game of orienteering and uses geo-tagged photo collections created by users. In order to complete a game, a player must visit a set of targets, with photos chosen from a user-generated geo-tagged database. The main contributions are that the game is created by using data from a user-generated multimedia collection and that the players can freely chose the order of the targets, so we encourage them to try different strategies to solve the shortest and fastest path problem. Our

game can be also used for creating sightseeing tours in the form of an orienteering game. O-MOPSI has been presented at an annual international festival which is aimed at introducing science and technology to school children, and has received positive feedback from the players.

5 Conclusions

We have studied how location-based data can be used in two types of applications: web data mining and mobile games. Location is an important factor in personalizing applications where user context is needed.

Typically, websites do not include explicit location data, but postal addresses are the most common way locations are included into web pages. We have proposed a method to detect location-based data, starting from identifying addresses and using them to detect additional information such as title and representative image. This approach works well especially with commercial services and service directories, because they have address information written in a clear way, but it depends on the quality of the gazetteer we use. Detecting service title, address and representative image allows us to provide meaningful results in a compact and informative way that is suitable for the mobile users.

In future work, we will consider how to detect what type of website contains location data. It can be a single entity (person, business or place), a chain of services (such as restaurants, banks or shops) or a service directory that displays all the items that are related to a certain field or area (such as restaurants in one city or local businesses in an area). This would improve the quality of the search results, especially on the websites that are not service directories, because each type of website embeds location data in a different way and because in single service websites the name of the service is often not close to the address. Furthermore, our address detection method needs to be improved to support abbreviations and mistyped words. This can be done by improving our string matching algorithm that currently supports only exact matches. Further study needs to be done for other regions than Finland, which might have higher variation in how addresses are written.

We adapted the classical game of orienteering to be played on mobile phone using GPS, digital maps and geo-tagged photos our users have collected. Our goals were to provide a fun way for doing physical exercise, to create an environment that helps learning map navigation and finding the shortest path, and to encourage our users to collect geo-tagged photos and to create games. We have presented the game at a science festival for five years in a row and we plan to improve it by conducting larger-scale testing and using the feedback data to improve its usability aspects. Aside from the technical and conceptual issues from creating the game, more research is needed to measure the impact in improving the players' attitude towards physical exercise and towards the adoption of new technologies.

Bibliography

[Abit97] Abiteboul, S.: "Querying semi-structured data". In: *Database Theory—ICDT'97*, *Lecture Notes in Computer Science*. vol. 1186. Berlin, Heidelberg : Springer Berlin Heidelberg, 1997, pp. 1–18

[AhBo08a] Ahlers, D. and S. Boll: "Retrieving address-based locations from the web". In: *Proceeding of the 2nd international workshop on Geographic information retrieval - GIR '08*. New York, New York, USA : ACM Press, 2008, p. 27

[AhBo08b] Ahlers, D. and S. Boll: "Urban web crawling". In: *Proceedings of the first international workshop on Location and the web - LOCWEB '08*. New York, New York, USA : ACM Press, 2008, pp. 25–32

[AhCo75] Aho, A. V. and M. J. Corasick: Efficient string matching: an aid to bibliographic search. In: *Communications of the ACM* vol. 18 (1975), pp. 333–340

[AHSS04] Amitay, E., N. Har'El, R. Sivan, and A. Soffer: "Web-a-where". In: *Proceedings of the 27th annual international conference on Research and development in information retrieval - SIGIR '04*. New York, New York, USA : ACM Press, 2004, p. 273

[BCFD06] Benford, S., A. Crabtree, M. Flintham, A. Drozd, R. Anastasi, M. Paxton, N. Tandavanitj, M. Adams, et al.: Can you see me now? In: *ACM Transactions on Computer-Human Interaction* vol. 13 (2006), Nr. 1

[BRWY11] Bennett, P. N., F. Radlinski, R. W. White, and E. Yilmaz: "Inferring and using location metadata to personalize web search". In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 135–144

[BFHL01] Bjork, S., J. Falk, R. Hansson, and P. Ljungstrand: "Pirates! using the physical world as a game board". In: *13th International Conference on Human-Comupter Interaction*. Tokyo, Japan, 2001, p. 8

[BLMD07] Borges, K. A. V., A. H. F. Laender, C. B. Medeiros, and C. Davis Jr.: "Discovering geographic locations in web pages using urban addresses". In: *GIR '07 Proceedings of the 4th ACM workshop on Geographical information retrieval*. Lisbon, Portugal, 2007, pp. 31–36

[BoYa13] Boulos, M. N. K. and S. P. Yang: Exergames for health and fitness: the roles of GPS and geosocial apps. In: *International journal of health geographics* vol. 12 (2013), Nr. 1, p. 18

[BCGG99] Buyukkokten, O., J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar: "Exploiting Geographical Location Information of Web Pages". In: *In Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*. Philadelphia, Pennsylvania, USA, 1999, pp. 1–6

[CYWM03] Cai, D., S. Yu, J.-R. R. Wen, and W.-Y. Y. Ma: VIPS: A vision-based page segmentation algorithm. In: *Microsoft technical report, MSR-TR-2003-79* (2003), pp. 1–29

[CaWJ05] Cai, W., S. Wang, and Q. Jiang: Address extraction: Extraction of location-based information from the web. In: *Web Technologies Research and Development* (2005), pp. 925–937

[Came04] Cameron, L. S.: *The geocaching handbook* : Globe Pequot, 2004

[CQXW05] Can, L., Z. Qian, M. Xiaofeng, and L. Wenyin: "Postal address detection fromweb documents". In: *Proceedings of International Workshop on Challenges in Web Information Retrieval and Integration. WIRI'05.*. Tokyo, Japan, 2005, pp. 40–45

[ChLi10] Chang, C. H. and S.-Y. Y. Li: "MapMarker: Extraction of postal addresses and associated information for general web pages". In: *International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. vol. 1. Toronto, Canada, 2010, pp. 105–111

[ChZF10] Chen, J., Q. Zhao, and P. Franti: "Smart Swap for more efficient clustering". In: *The 2010 International Conference on Green Circuits and Systems*. Shanghai, China, 2010, pp. 446–450

[ChXF12] Chen, M., M. Xu, and P. Fränti: A fast O(N) multiresolution polygonal approximation algorithm for GPS trajectory simplification. In: *IEEE Transactions on Image Processing* vol. 21 (2012), pp. 2770–2785

[CGLF04] Cheok, A. D., K. H. Goh, W. Liu, F. Farbiz, S. W. Fong, S. L. Teo, Y. Li, and X. Yang: Human Pacman: A mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. In: *Personal and Ubiquitous Computing* vol. 8 (2004), pp. 71–81

[ChSi12] Chittaro, L. and R. Sioni: "Turning the classic snake mobile game into a location-based exergame that encourages walking". In: *Lecture Notes in Computer Science*. vol. 7284, 2012, pp. 43–54

[CSLJ09] De Choudhury, M., H. Sundaram, Y. R. Lin, A. John, and D. D. Seligmann: "Connecting content to community in social media via image content, user tags and user communication". In: *Proceedings of 2009 IEEE International Conference on Multimedia and Expo, ICME*. New York, NY, USA, 2009, pp. 1238–1241

[Delo10] Delort, J.-Y.: "Vizualizing Large Spatial Datasets in Interactive Maps". In: *2010 Second International Conference on Advanced Geographic Information Systems, Applications, and Services*. St. Maarten, Netherlands Antilles : IEEE, 2010, pp. 33–38

[DiGS00] Ding, J., L. Gravano, and N. Shivakumar: "Computing geographical scopes of web resources". In: *Proceedings of the*

26th International Conference on Very Large Data Bases. Cairo, Egypt, 2000, pp. 545–556

[DoHu12] Dou, W. and J. Hu: "Automated Web Data Mining Using Semantic Analysis". In: *Advanced Data Mining and Applications*, 2012, pp. 539–551

[EgPa10] Egnor, D. and E. Pasztor: Generating structured information, Google Patents (2010). — US Patent 7,788,293

[FJSR04] Facer, K., R. Joiner, D. Stanton, J. Reid, R. Hull, and D. Kirk: Savannah: Mobile gaming and learning? In: *Journal of Computer Assisted Learning* vol. 20 (2004), pp. 399–409

[FLMN10] Florczyk, A. J., F. J. López-Pellicer, P. Muro-Medrano, J. Nogueras-Iso, and F. J. Zarazaga-Soria: Semantic selection of georeferencing services for urban management. In: *Journal of Information Technology in Construction* vol. 15 (2010), pp. 111–121

[FKTS10] Fränti, P., J. Kuittinen, A. Tabarcea, and L. Sakala: "MOPSI location-based search engine". In: *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*. Sierre, Switzerland, 2010, p. 872

[FTKH10] Fränti, P., A. Tabarcea, J. Kuittinen, and V. Hautamäki: "Location-based search engine for multimedia phones". In: *2010 IEEE International Conference on Multimedia and Expo, ICME 2010*. Singapore, 2010, pp. 558–563

[GaTF15] Gali, N., A. Tabarcea, and P. Franti: Rule-based technique for extracting representative title for services detected in webpage. In: *manuscript* (2015)

[GoWK07] Goldberg, D. W., J. P. Wilson, and C. A. Knoblock: From text to geographic coordinates: the current state of geocoding. In: *URISA journal* vol. 19 (2007), Nr. 1, pp. 33–46

[HaFM02] Hariharan, G., P. Fränti, and S. Mehta: "Data mining for personal navigation". In: *AeroSense 2002*. Orlando, Fl, USA, 2002, pp. 355–365

[HeMS13] Hess, B., F. Magagna, and J. Sutanto: Toward location-aware Web: extraction method, applications and evaluation. In: *Personal and Ubiquitous Computing* vol. 18, Springer (2013), Nr. 5, pp. 1047–1060

[HiFZ99] Hill, L. L., J. Frew, and Q. Zheng: The Implementation of a Gazetteer in a Georeferenced Digital Library. In: *D-Lib Magazine* vol. 5 (1999), pp. 1–17

[HuLR05] Hu, Y.-H., S. Lim, and C. Rizos: *"Georeferencing of Web Pages Based on Context-Aware Conceptual Relationship Analysis"*. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.434.1333&rep=rep1&type=pdf

[HuFi10] Huitema, P. and P. Fizzano: "A crawler for local search". In: *4th International Conference on Digital Society, ICDS 2010*. St. Maarten, Netherlands Antilles, 2010, pp. 86–91

[JaMo06] Jansen, B. J. and P. R. Molina: The effectiveness of Web search engines for retrieving relevant ecommerce links. In: *Information Processing & Management* vol. 42 (2006), Nr. 4, pp. 1075–1098

[JoKo10] Jormanainen, I. and P. Korhonen: "Science festivals on computer science recruitment". In: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. Koli, Finland, 2010, pp. 72–73

[KaKa09] Kasemsuppakorn, P. and H. A. Karimi: "Pedestrian network data collection through location-based social networks". In: *5th International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Washington D.C., USA, 2009

[Küpp05] Küpper, A.: *Location-Based Services: Fundamentals and Operation*. 1st. ed. : Wiley Online Library, 2005

[LHNL04] Lankoski, P., S. Heliö, J. Nummela, J. Lahti, F. Mäyrä, and L. Ermi: "A Case Study in Pervasive Game Design: The Songs of North". In: *Proceedings of the third Nordic conference on Human-computer interaction - NordiCHI '04*. Tampere, Finland, 2004, pp. 413–416

[LeLM07] Lee, H. C., H. Liu, and R. J. Miller: "Geographically-Sensitive Link Analysis". In: *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*. Sillicon Valley, USA : IEEE, 2007, pp. 628–634

[LeLL13] Leung, K. W. T., D. L. Lee, and W.-C. Lee: Pmse: A personalized mobile search engine. In: *IEEE Transactions on Knowledge and Data Engineering* vol. 25, IEEE (2013), Nr. 4, pp. 820–834

[LiOO08] Li, M., M. O'Grady, and M. P. O'Hare, Gregory: "Geo-gaming: The mobile monopoly experience". In: *Fourth International Conference on Web Information Systems (WEBIST2008),*. Madeira, Portugal, 2008, pp. 220–223

[LGCZ12] Li, W., M. F. Goodchild, R. L. Church, and B. Zhou: "Geospatial Data Mining on the Web: Discovering Locations of Emergency Service Facilities". In: *Advanced Data Mining and Applications* : Springer, 2012, pp. 552–563

[LiRG10] Liu, C., P.-L. P. Rau, and F. Gao: Mobile information search for location-based information. In: *Computers in industry* vol. 61, Elsevier (2010), Nr. 4, pp. 364–371

[LiMM10] Liu, W., X. Meng, and W. Meng: ViDE: A Vision-Based Approach for Deep Web Data Extraction. In: *IEEE Transactions on Knowledge and Data Engineering* vol. 22, IEEE (2010), Nr. 3, pp. 447–460

[LuCR01] Luo, M. R., G. Cui, and B. Rigg: The development of the CIE 2000 colour-difference formula: CIEDE2000. In: *Color Research and Application* vol. 26 (2001), pp. 340–350

[Mari13] Mariescu-Istodor, R.: *"Detecting user actions in MOPSI"*, University of Eastern Finland, 2013

[MTSF14] Mariescu-Istodor, R., A. Tabarcea, R. Saeidi, and P. Fränti: "Low Complexity Spatial Similarity Measure of GPS Trajectories". In: *WEBIST 2014*. Barcelona, Spain, 2014, pp. 62–69

[MCSL05] Markowetz, A., Y.-Y. Chen, T. Suel, X. Long, and B. Seeger: "Design and Implementation of a Geographic Search Engine". In: *Eighth International Workshop on the Web and Databases (WebDB 2005)*. Baltimore, USA, 2005

[MMSK08] Matyas, S., C. Matyas, C. Schlieder, P. Kiefer, H. Mitarai, and M. Kamata: "Designing location-based mobile games with a purpose: collecting geospatial data with CityExplorer". In: *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. Yokohama, Japan, 2008, pp. 244–247

[Mccu01] McCurley, K. S.: "Geospatial mapping and navigation of the web". In: *Proceedings of the 10th international conference on World Wide Web*. vol. WWW '01: P. Hong Kong, 2001, pp. 221–229

[MiMG99] Mikheev, A., M. Moens, and C. Grover: "Named Entity recognition without gazetteers". In: *Proceedings of the 9th conference on European chapter of the Association for Computational Linguistics*. Bergen, Norway, 1999, pp. 1–8

[MHKT09] Misund, G., H. Holone, J. Karlsen, and H. Tolsby: "Chase and Catch - simple as that?". In: *Proceedings of the International Conference on Advances in Computer Enterntainment Technology - ACE '09*. Athens, Greece, 2009, p. 73

[NaRa02] Navarro, G. and M. Raffinot: *Flexible pattern matching in strings: practical on-line search algorithms for texts and biological sequences* : Cambridge University Press, 2002

[NeJu12] Neustaedter, C. and T. K. Judge: "See it". In: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion - CSCW '12*. New York, New York, USA, 2012, p. 235

[PBMW99] Page, L., S. Brin, R. Motwani, and T. Winograd: The PageRank citation ranking: Bringing order to the web. In: *Technical report, Stanford Digital Library Technologies Project*, Stanford InfoLab (1999)

[PaMP03] Patterson, C. A., R. R. Muntz, and C. M. Pancake: Challenges in location-aware computing. In: *Pervasive Computing, IEEE* vol. 2, IEEE (2003), Nr. 2, pp. 80–89

[PCRC08] Petit, M., C. Claramunt, C. Ray, and G. Calvary: A design process for the development of an interactive and adaptive GIS. In: *Lecture Notes in Computer Science* vol. 5573 (2008), Nr. Web and Wireless Geographical Information Systems, pp. 96–106

[PiTh02] Piekarski, W. and B. Thomas: ARQuake: the outdoor augmented reality gaming system. In: *Communications of the ACM* vol. 45 (2002), pp. 36–38

[PCJA07] Purves, R. S., P. Clough, C. B. Jones, A. Arampatzis, B. Bucher, D. Finch, G. Fu, H. Joho, et al.: The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the Internet. In: *International Journal of Geographical Information Science* vol. 21, Taylor & Francis (2007), Nr. 7, pp. 717–745

[QXFX10] Qin, T., R. Xiao, L. Fang, X. Xie, and L. Zhang: "An efficient location extraction algorithm by leveraging web contextual information". In: *Proceedings of the 18th*

SIGSPATIAL international conference on advances in geographic information systems. San Jose, CA, USA, 2010, pp. 53–60

[RMCE06] Rashid, O., I. Mullins, P. Coulton, and R. Edwards: Extending cyberspace: location based games using cellular phones. In: *Computer Entertainment* vol. 4 (2006), p. 4

[ScVo04] Schiller, J. and A. Voisard: *Location Based Services.* 1st. ed. : Morgan Kaufmann Publishers Inc., 2004

[ScKM05] Schlieder, C., P. Kiefer, and S. Matyas: "Geogames: A conceptual framework and tool for the design of location-based games from classic board games". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* vol. 3814 LNAI, 2005, pp. 164–173

[SMRS13] Schmidt, S., S. Manschitz, C. Rensing, and R. Steinmetz: "Extraction of Address Data from Unstructured Text using Free Knowledge Resources". In: *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*. Graz, Austria, 2013, p. 7

[SeBD09] Setlur, V., A. Battestini, and X. Ding: "Travel scrapbooks: Creating rich visual travel narratives". In: *2009 IEEE International Conference on Multimedia and Expo*. New York, NY, USA : IEEE, 2009, pp. 1314–1317

[ShBa11] Shi, G. and K. Barker: "Extraction of geospatial information on the Web for GIS applications". In: *IEEE 10th International Conference on Cognitive Informatics and Cognitive Computing (ICCI-CC'11)*. Banff, AB, Canada, 2011, pp. 41–48

[SMCA06] Silva, M. J., B. Martins, M. Chaves, A. P. Afonso, and N. Cardoso: Adding geographic scopes to web resources. In: *Computers, Environment and Urban Systems* vol. 30 (2006), pp. 378–399

[SDBD05] Souza, L. A., C. A. Davis Jr., K. A. V. Borges, T. M. Delboni, and A. H. F. Laender: "The Role of Gazetteers in

Geographic Knowledge Discovery on the Web". In: *Third Latin American Web Congress (LA-WEB'2005)*. vol. 2005. Buenos AIres, Argentina : IEEE, 2005, pp. 157–165

[SpMi08] Spikol, D. and M. Milrad: "Combining physical activities and mobile games to promote novel learning practices". In: *Proceedings - 5th IEEE International Conference on Wireless, Mobile, and Ubiquitous Technologies in Education, WMUTE 2008*. Beijing, China, 2008, pp. 31–38

[SuKo06] Suomela, R. and A. Koivisto: "My photos are my bullets–using camera as the primary means of player-to-player interaction in a mobile multiplayer game". In: *Entertainment Computing-ICEC 2006*. Cambridge, UK, 2006, pp. 250–261

[TaFM09] Tabarcea, A., P. Fränti, and V. Manta: Using a Spatial Database in a Location-Based Search Application. In: *Buletinul Institutului Politehnic Iasi* vol. LV (LIX) (2009), Nr. 3, pp. 55–62

[Tsai11] Tsai, F. S.: Web-based geographic search engine for location-aware search in Singapore. In: *Expert Systems with Applications* vol. 38, Elsevier (2011), Nr. 1, pp. 1011–1016

[WTCF12] Waga, K., A. Tabarcea, M. Chen, and P. Fränti: "Detecting Movement Type by Route Segmentation and Classification". In: *Conference on Collaborative Computing: Networking, Applications and Worksharing*. Pittsburgh, Pennsylvania, United States, 2012, pp. 508–513

[WaTF12] Waga, K., A. Tabarcea, and P. Fränti: "Recommendation of points of interest from user generated data collection". In: *CollaborateCom*. Pittsburgh, Pennsylvania, United States, 2012, pp. 550–555

[WTMF13] Waga, K., A. Tabarcea, R. Mariescu-Istodor, and P. Fränti: "Real Time Access to Multiple GPS Tracks.". In: *WEBIST*. Achen, Germany, 2013, pp. 293–299

[Wan14] Wan, Z.: *"O-Mopsi: Location-based Orienteering Mobile Game"*, University of Eastern Finland, 2014

[WXWL05] Wang, C., X. Xie, L. Wang, Y. Lu, and W.-Y. Y. Ma: "Detecting geographic locations from Web resources". In: YORK:ACM, N. (ed.): *Proceedings of the 2005 Workshop in Geographic Information Retrieval*. Bremen, Germany, 2005, pp. 17–24

[WaAm03] Watters, C. and G. Amoudi: GeoSearcher: Location-based ranking of search engine results. In: *Journal of the American Society for Information Science and Technology* vol. 54, Wiley Online Library (2003), Nr. 2, pp. 140–151

[WeBO12] Wetzel, R., L. Blum, and L. Oppermann: "Tidy city". In: *Proceedings of the International Conference on the Foundations of Digital Games - FDG '12*. Raleigh, North Carolina, USA, 2012, p. 238

[ViNa05] Viola, P. and M. Narasimhan: "Learning to extract information from semi-structured text using a discriminative context free grammar". In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. vol. 272, 2005, pp. 330–337

[Väns04] Vänskä, I.: *"Using location information in web documents"*, University of Eastern Finland, 2004

[YoTM01] Yokoji, S., K. Takahashi, and N. Miura: Kokono Search: A Location Based Search Engine. In: *10th International World Wide Web Conference (WWW10)*. Raleigh, North Carolina, USA (2001)

[YKSJ09] Yu, Y. H., J. H. Kim, K. Shin, and G. S. Jo: Recommendation system using location-based ontology on wireless internet: An example of collective intelligence by using "mashup" applications. In: *Expert Systems with Applications* vol. 36 (2009), pp. 11675–11681

[ZJLY12] Zhang, Q., P. Jin, S. Lin, and L. Yue: "Extracting focused locations for web pages". In: *Web-Age Information Management* : Springer, 2012, pp. 76–89

Paper P1

P. Fränti, J. Chen, A. Tabarcea, "Four aspects of relevance in sharing location-based media: content, time, location and network", Int. Conf. on Web Information Systems and Technologies (WEBIST'11), Noordwijkerhout, Netherlands, 413–417, May 2011.

FOUR ASPECTS OF RELEVANCE IN SHARING LOCATION-BASED MEDIA: CONTENT, TIME, LOCATION AND NETWORK

Pasi Fränti, Jinhua Chen and Andrei Tabarcea
Speech & Image Processing Unit, School of Computing, University of Eastern Finland, Joensuu
franti, jinchen, tabarcea @cs.joensuu.fi

Keywords: Data sharing, relevance, location-based applications.

Abstract: Sharing information via internet is popular but the key problem is how to find relevant information. Two new features are becoming more popular: location and the social network of the user. We hypothesize that the relevance of data is defined by four aspects: content, time, location, and user network. We study how the location aspect is used in a media-sharing service called MOPSI.

1 INTRODUCTION

Location-based services are becoming widely used due to the fast development of positioning systems in multimedia phones. Location provides additional information that can be expressed as a point of interest, route, or geographic area. The location can be considered information as itself but it is often attached to other data, and shared via location-based service or photo sharing site.

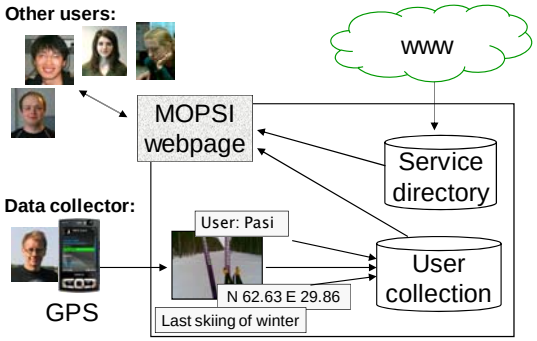


Figure 1: Diagram of the MOPSI data collection and services. Available on-line: http://cs.joensuu.fi/mopsi/

In this paper, we study mobile location-based media sharing via internet by a case study based on MOPSI service, which is prototype service for sharing location-based media. The overall structure of the system is outlined in Fig. 1 consisting of two main parts: user collection and service directory.

The main limitation of this kind of ad hoc information sharing is unawareness of the material of others especially if the users are not directly linked with each other. The data itself may be available in the service but the problem is how to find *relevant* data from service with a large number of users. We argue that relevance can be defined by the following aspects:

(1) Content of the data
(2) Location
(3) Time
(4) Author and his/her network

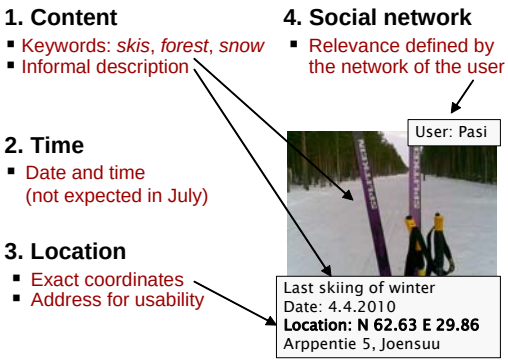


Figure 2: Four aspects of relevance in practice.

These four aspects are demonstrated in Fig. 2 by a concrete example where a person wanted to capture the following scenario. From the photo and its description we can see skis, forest and snow, which relate to wintertime activity. The data also reveals when and where the picture was taken. In 4th April 2010, there was skiing tracks available, which was not self-evident even for citizens of Joensuu. Knowing a proper location was essential. The last piece of information is the identity of the user

himself. Strangers may not benefit much of this information but those who know him and share the same hobbies are more likely to find this useful.

2 CONTENT, TIME AND LOCATION

We discuss the three main aspects of MOPSI system, of which location is an essential part. We describe the system as its current state, and discuss its design alternatives.

2.1 Content

Traditionally the relevance is defined by the content either by user-given keywords, or using a predefined format in database system, and then retrieved using SQL queries. This requires well-designed static database where the service provider models the user behavior beforehand and provides information in form of service directory.

In internet, well-defined attributes are not used but relevant content can still be found from free text using search engine if the content matches to the keywords provided by the user. Tagging of the photos can also be done afterwards, but usually free-form textual explanation is simpler. It also serves the purpose of social media.

In MOPSI, free-form text description is supported instead of manual tagging. For browsing the data on web, queries based on time, location and content have been implemented. A simple recommendation framework is also in MOPSI based on user location and rating of the photos.

Further analysis of the relevance, content-based image retrieval could be done based on color, texture and shape features. Automatic image categorization, aims at converting visual content into a set of keywords to describe the content. In (Choudhury et al., 2009, Yu et al., 2009) both visual content and user tagging are jointly applied to recommend the group, where a photo should best fit in.

2.2 Location

Exploiting the l*ocation* of the user has become popular due to wide availability of GPS positioning in multimedia phones. In case of lacking GPS, positioning can also be provided by cellular network of mobile phone, or even using the IP address for a rough estimation of location. Once the location is known, it gives significant additional relevance that can be utilized in several different ways. In MOPSI, location is the key element and it provides additional relevance in the following ways:

(1) Browsing data collection on a map
(2) Show location of other users
(3) Track the movements of the user
(4) Filter relevant search results for the service directory

Fig. 3 demonstrates the map view in MOPSI where photos have been clustered and then shown using GoogleMaps API.

Figure 3: Map view of the data collection.

Location of users has been visualized in Fig. 4 using a so-called *smart swap* algorithm (Chen et al., 2010) that provides accurate clustering in real-time. For representing the clusters, approaches using icons, grids, Voronoi diagrams, and coloring by the density have been considered in (Delort, 2010). We use a color bubble attached with the text representing the most recent users in the cluster. The browsing is supported by zooming operation to get inside bigger clusters. Details of this solution will be reported later elsewhere.

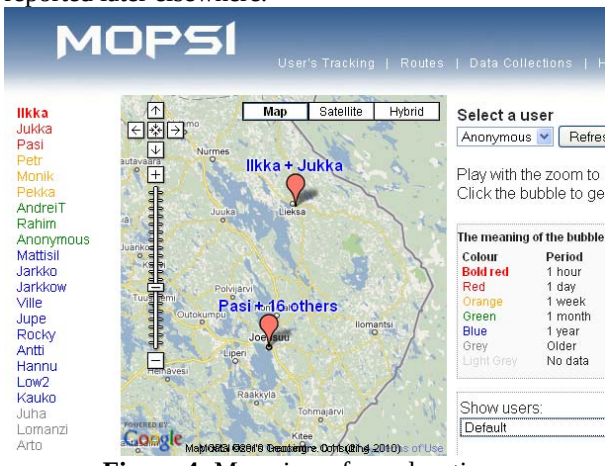


Figure 4: Map view of user locations.

The collection can also be used as a part of service directory in MOPSI either in mobile phone or on web, see Fig. 5. Given location, user makes query by keyword, but instead of providing relevant search results by the content alone, results nearby are given if they exist in a local database (green), or found in the user collection (yellow).

Additional information (red) is provided by *location-based search* (Fränti et al., 2010), which is a combination of traditional location-based service and search engine. Following the idea in (Huitema and Fizzano, 2010), MOPSI allows user to transfer search engine results (red) into the service directory (green) by adding proper keywords similarly, and by using photos from the user collection (yellow).

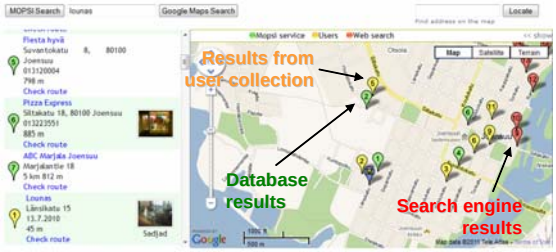


Figure 5: Web page interface to the service directory

2.3 Time

Time can be added to the relevance of the data in several ways. Firstly, the information may be relevant only within certain time period. A concert or a sport event happens in certain time and day, and it is essential information for participants. In photo collection, the information can also be relevant to know when the photo was taken. In MOPSI collection, we utilize this by providing time line view to the data as shown in Fig. 6. Similar layout was considered in (Setlur et al., 2009), with the addition that also links to Wikipedia are supported to provide more information besides just the photos.

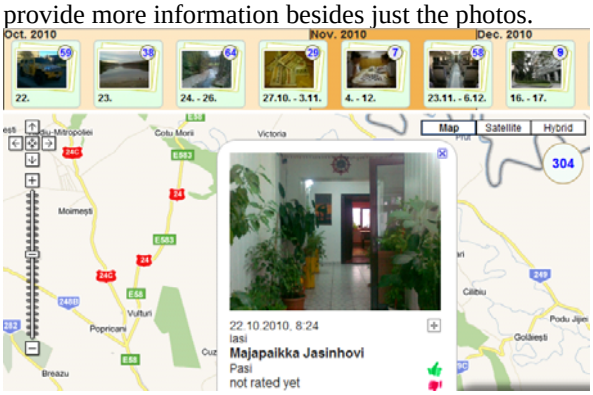


Figure 6: Time-line view of the data collection.

Secondly, the time and location themselves can be the essential data from an exercise session. For example, the jogging track shown in Fig. 7 records the length, duration and average speed. This is typical book-keeping for a long distance runner in his training. Although specialized GPS sport trackers exist, the use of MOPSI service and mobile phone allows automatic sending of the data into the server for user convenience. Moreover, photos can also be taken from the same session by the same device, and presented later jointly with the trajectory of the user as proposed in (Petit et al., 2008).

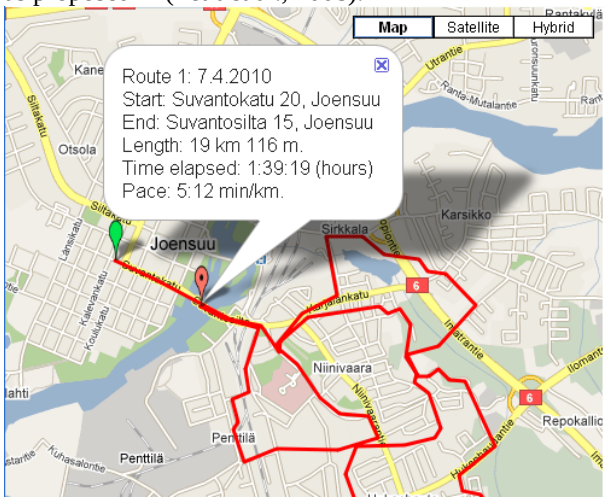


Figure 7: Joint time and location for tracking sport activity.

In MOPSI collection, tracking user's routes is one of the main functions. The web interface provides also navigation from the current location to the location of the search result using GoogleMaps API based on road maps. An interesting idea for future consideration would be to use the route collection of all users to offer better navigation for pedestrians and hikers instead of the road network more suitable for cars (Kasemsuppakorn et al., 2009).

Third possibility to utilize the time information is to consider the age of data. The newer the information the more likely it is still valid as the life expectancy of cafeterias, for example, in typical metropolitan area are often measured in months rather than in years. Moreover, information such as weather condition is needed right here and right now, so to speak. In Fig. 1, the skiing condition is recorded for 4th April, but it hardly relevant for users in July.

3 EXPERIMENTS

We next give overview of the data in the user collection so far as on 25th October 2010. The collection includes lots of test photos, and the number of users is small, which may somewhat skew the results. Nevertheless, some trends and observations can be seen.

In total, there were 3589 photos of which most are city views (839), then pictures of nature (801) and other people (279). Few pictures are also taken from events (90), documents (40) and animals (59). In addition, there are photos that are counted as test photos or failure pictures.

Another point of view is what kind of descriptions has been typed in by the users. Due to the experimental stage, a large amount of the photos (27%) are without any description. The lack of descriptions is also caused by the difficulty to type by mobile phone, but descriptions can be added later from the web interface.

Among the photos that have some kind of description, significant amount of photos (35%) have just garbage, some test word (*Symbian_test*), or very generic object description (*Mug*, *Wires*, *Mouse*) indicating test use. In total, 65% of all photos have a meaningful description. Mostly documented descriptions are travel photos of places (685), nature (579), general objects (263), architecture (212) people (210), and few general descriptions of events and animals.

People are often described by their names, or by their roles (*runner*, *floorball player*). Only few are related to place (*Untung / STMIK*), age (*Young Andrei*) or relationship to the person (*my son Amir*).

Events are significantly more often found in the user description than could be concluded by content analysis alone. In our case, events include mostly work-related meetings described by their acronyms (*ecse*, *abi*, *mopsi meeting*, *ubiikki*) but also running competition (*Åland half marathon*) and actions attached with feelings (*quality time in skiing elevator*).

Another difference between content and user description are travel photos. The location is not easy to recognize from content but it could be concluded from the positioning data. For example, *Clarke Quay*, *Geger beach*, *Suceava*, *Tahkovuori* and *Aholansaari* are locations whereas the following descriptions include additional details: *Petronas Towers* (building complex), *Heureka* (science center), *Singapore flier* (Ferris wheel) and *Olavin linna* (*castle*). The extreme case is *Musta Pekka mutkan takana* (Black Pete behind the curve) where Black Pete is the name of a particular slope in Tahko skiing resort.

Table 1: Distribution of keywords (tags) used in Picasa and Flickr, in comparison to the user descriptions of MOPSI collection.

Description:	Picasa	Flickr	MOPSI	
			All	Real
Places	---	28%	21%	32%
Events and action	31%	17%	5%	7%
People	6%	7%	6%	10%
Objects	---	5%	8%	12%
Architecture and nature	25%	21%	23%	37%
Animals	---	3%	2%	2%
Other	20%	16%	---	0%
Garbage	19%	2%	35%	---

Table 1 compares the textual description used in MOPSI with two other photo sharing sites. The main difference is that, in MOPSI, location is provided automatically without any user interaction.

In Picasa, users provide the location by dragging the photo on GoogleMap. Keywords and location are thus provided explicitly as two different entities, and consequently, users tend not to type any location related keywords. Flickr has somewhat more complicated interface based on *Yahoo! Maps*. Only a predefined set of keywords are allowed, which explains the quality of tags (only 2% garbage).

Despite the automatic positioning in MOPSI, it does not reflect on the distribution of the type of descriptions written. Unlike in Picasa, users still tend to describe the location anyway for travel pictures, probably because the position is not confirmed in the device, but it happens hidden in the background. Overall, the distribution of topics is rather similar to that of Flickr. There are slightly more people and objects described, but these could be just artifacts from the system being at testing stage.

For photo collecting, two mobile applications were developed (Java and Symbian C++). A large number of failures were caused by the Java version, which lacks several important features. Firstly, there is an unavoidable delay from the *click* sound and when the photo is actually taken. People tend to move the camera right after they hear the sound and before the actual picture will be taken. Secondly, Auto-focus supported by Symbian helps a lot with picture quality but it was not available in Java. Other typical failures originate from low quality cameras that do not work well in low illumination. Few damaged pictures were caused by irrecoverable transmission error. Samples are shown in Fig. 8.

Figure 8: Photos of the first row are examples of software problems (click sound), the second row of low illumination and broken transmission problems. The rest are successful photos.

4 CONCLUSIONS

We have presented a case study of MOPSI location-based media collection and sharing service, and studied how different aspects of relevance appear in the system. So far the system has been used for collecting user data (mainly photos and routes), served as a test bench of new ideas, and a prototype service directory. In all these, the location is a key factor.

The media collection tool is also in professional use by partnering companies for documenting purpose, and can be used later for mobile location-based games and an educational tool for teaching principles of GIS, and other classes such as biology. The fourth aspect of relevance, social network, will be studied in future.

REFERENCES

Choudhury M.D., Sundaram H., Lin Y.-R., John A., Seligmann D.D., "Connecting content to community in social media via image content, user tags and user communication", *ICME 2009*, 1238-1241, New York City, July 2009.

Yu J., Joshi D., Luo J., "Connecting people in photo-sharing sites by photo content and user annotations", *ICME 2009*, 1464-1467, New York City, July 2009.

Chen J., Zhao Q., and Fränti P., "Smart swap for more efficient clustering", *Int. Conf. Green Circuits and Systems (ICGCS'10)*, Shanghai, China, June 2010.

Delort J.-Y., "Vizualizing large spatial datasets in interactive maps", *IEEE Int. Conf. Advanced Geographic Information Systems, Applications, and Services*, St. Maarten, Netherlands Antilles, 33-38, Feb 2010.

Fränti P. , Tabarcea A., Kuittinen J., Hautamäki V., "Location-based search engine for multimedia phones", *IEEE Int. Conf. on Multimedia & Expo (ICME'10)*, Singapore, July 2010.

Huitema P. and Fizzano P., "A Crawler for Local Search", *IEEE Int. Conf. Digital Society (ICDS)*, St. Maarten, Netherlands Antilles, 86-91, Feb 2010.

Setlur V., Battestini A., Ding X., "Travel scrapbooks: creating rich visual travel narratives", *ICME 2009*, 1314-1317, New York City, July 2009.

Petit M., Claramunt C., Ray C. and Calvary G., "A design process for the development of an interactive and adaptive GIS", *W2GIS*, 96-106, Shanghai, China 2008.

Kasemsuppakorn P., Karimi H.A., "Pedestrian network data collection through location-based social networks", *Collaborate COM*, Crystal City, Washington DC, Nov 2009.

Paper P2

P. Fränti, A. Tabarcea, J. Kuittinen, V. Hautamäki, "Location-based search engine for multimedia phones", IEEE Int. Conf. on Multimedia and Expo (ICME'10), Singapore, 558–563, July 2010

LOCATION-BASED SEARCH ENGINE FOR MULTIMEDIA PHONES

Pasi Fränti[1], Andrei Tabarcea[1], Juha Kuittinen[1] and Ville Hautamäki[2]

[1]Speech and Image Processing Unit, University of Eastern Finland, Joensuu
{franti, tabarcea, jkuitti}@cs.joensuu.fi

[2]Institute for Infocomm Research, A*STAR, Singapore
vishv@i2r.a-star.edu.sg

ABSTRACT

Location-based search engine is an alternative approach for information retrieval to traditional location-based services based on fixed databases. This is a relatively new concept that aims at utilizing the location of user but without restricting to any fixed location-based service. In this paper, we outline a prototype solution for multimedia mobile phones based on web search, ad-hoc georeferencing, prefix tree structure and gazetteer. Experimental results show that the proposed solution finds search results that have higher or equal mean relevance than that of the GoogleMaps and YellowPages.

Keywords— Search engine, LBS, mobile device, location information, personal navigation, WWW.

1. INTRODUCTION

Exploiting the l*ocation* of the user has become popular during recent years due to increasingly wide availability of GPS positioning in multimedia phones. For instance, according to Nokia's estimate more than half of their phones will include GPS by 2010-2012. Moreover, in case of lacking GPS, positioning can also be based on cellular network or even use IP address for a rough estimation of location.

Locations-based services (LBS) such as Yellow Pages[1], Google Maps [2] and Nokia Ovi Services [3] are therefore expected to emerge very fast to our everyday life via mobile phones and other consumer electronics. Their main limitation, however, is that they are fully or partially based on databases where the entries must be explicitly georeferenced beforehand when added.

Search engines, on the other hand, are efficient in finding information from internet without any prior knowledge or explicit search structure. Their limitation is that the location of the user is not yet well utilized in the current solutions. There are two reasons for this. Firstly, the location of user was not as widely available as it is nowadays. Secondly, the information in WWW is rarely attached with the location for which it would be relevant.

In this paper, we propose an alternative solution based on web search and ad-hoc georeferencing. We denote this as *location-based search engine* to emphasize its seemingly small but significant distinction from location-based services. It aims at combining the benefits of web search and traditional location-based services exploiting the location.

The main problem of this approach is that only very few pages have explicit georeferencing in form of geotagging, using address field or by other means. However, it is rather common that web pages include street or postal addresses as free (non-tagged) text. According to [9] most of relevant services (especially commercial ones) can be found in this way. Based on this observation, the authors have constructed a prototype solution called *MOPSI Search*.

The general idea behind the solution uses the idea originally outlined in [5], but not implemented in practice until now. We describe here the technical solutions for implementing the system on multimedia mobile phone and provide experimental comparison of its search capability in comparison to existing LBS solutions such as GoogleMaps and YellowPages. The technical solution for the ad-hoc georeferencing uses the same principle as in [11].

The workflow of the proposed solution is as follows. Once the location is given, we search for relevant web pages based on given keywords, extract potential address information, and compare them to the entries in a gazetteer. Positive results are returned as search results according to their distance relative to the user location. The location can also be used for plotting the target location on the map or by giving navigational information towards the location.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we give detailed description of our solution called MOPSI search engine (see Fig. 1). It uses Google as a search engine and

[1] http://en.02.fi/yellow+pages/
[2] http:/maps.google.com/
[3] http://www.ovi.com/services/

978-1-4244-7493-6/10/$26.00 ©2010 IEEE

post-processes its search results to obtain information relevant to user's location. A prototype applied for Finland is demonstrated in Section 4. Experimental search results are also shown and briefly compared to two commercial solutions. Conclusions are then drawn in Section 5.

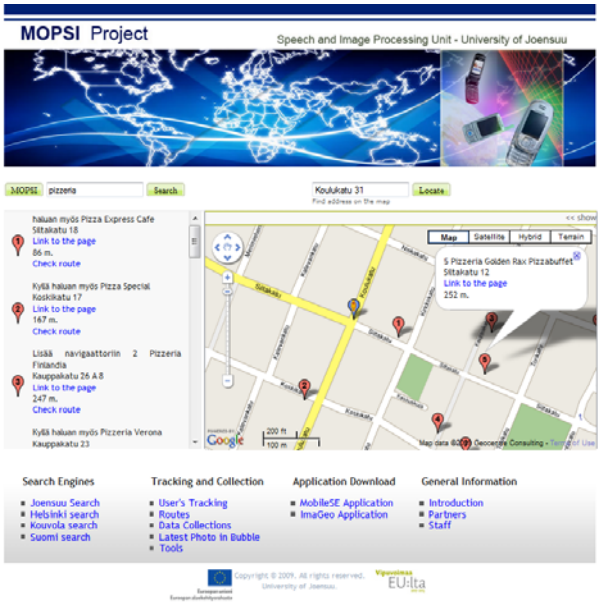


Fig. 1. Web interface of the MOPSI search engine.

2. RELATED WORK

In the recent years there have been related efforts to implement location aware web search engines. Most of the documented efforts are focused on creating explicit location information detectors that are used in the form of tags, in contrast to our approach where no explicit tags are used. Otherwise the approaches are similar: they rank the web pages according to their spatial contents. In some solutions, the found web pages are indexed for future search.

2.1. Web Search

The application in [2] uses web pages as the main source of location-based information. It relies on web crawling, which is targeted to create topical web indexes. The goal is to locate only links relevant to the topic of concern, and avoiding exhaustive crawls. The approach is similar to ours in a sense that its uses open web for detecting postal addresses and descriptive information. However, we don't use a web crawler but we rely on an external search engine.

The application in [7] consists of an indexed collection of web pages and supports both pure text and spatial indexing, which includes spatial relationships such as "inside", "outside" and "near". Conversely, it enables user to specify a search query and a geographical location. The queries are then tested for ambiguity and alternative options are presented for disambiguation, which is done before submission to the search engine by using geographical ontology. Relevance ranking is executed with respect to the non-spatial and spatial elements of the query.

In respect to existing commercial services such as Google Maps, Bing Local [4], Yahoo Local [5] and Yellow Pages, our goal is the same: provide location-relevant information to the user. However, the existing applications are mainly based on commercial databases, and only into limited extent, exploit the results on real-time web search.

2.2. Address Extraction

An essential part of our application is to detect and extract locations (addresses) from web pages. In the reviewed literature, various types of address detection methods have been published during recent years.

Methods of detecting the location of a web resource are initially found in [3, 9]. In [3], "whois" records are used for retrieving phone numbers of network administrators, which are used together with zip code and area database to assign coordinates to Class A and B domains. In [9] hyperlinks, meta tags and postal addresses are also used for additional information. The addresses are detected using a postal code database with latitude/longitude information. We also detect the postal addresses from database with latitude/longitude information, but our granularity is higher, as we use street names and street numbers.

In [8], regular expressions are used to detect patterns of typical address elements and the results are validated using a database, but the focus of the paper is how to rank the search results. Earlier prototypes of our application were also based on regular expression to detect Finnish street names. However, we later change this and started to use gazetteer in order to cover the addresses more thoroughly.

One way to detect addresses from free form text is to build a classifier and let it detect addresses from the web-pages as in [12]. However, customizing the classifier to other languages and countries takes a considerable work as new ground truth tagged text corpus must be created by hand. In our approach, no ground truth tagging is needed. The only things needed are a gazetteer and simple rules on how the street name appears relative to other address fields. Efficient use of the gazetteer is possible because we know the user's current location and the interest area is limited to services close to him. We can therefore build fast access structure to a partial gazetteer.

In [1], database is used to detect and validate addresses from web-page. In addition, different variations of street names are detected and the text is verified for occurrences of address elements such as street names, city names and zip codes. As a result, correspondence between data and text can be examined. We use similar idea for identifying

[4] http://www.bing.com/local/
[5] http://local.yahoo.com/

address elements in our geoparsing algorithm and validating the address by geocoding. The difference is that we use explicit geocoded database and rely on street-name detection, while [1] uses freely available geocoders.

In [4], a syntactic approach to postal address detection is proposed consisting of two steps: vision-based text segmentation and syntactic pattern recognition. The text segmentation analyses the html tags and detects cue blocks (for the purpose of indications, annotation, and explanation) and body blocks (main text body content). Recognition of postal addresses relies on calculating the confidence of the detected blocks, which in turn is based on tokenization of the words. The tokenization process uses city names, state names, street and organization suffixes, but not street names. Our approach is simpler, as we filter out all the html tags, and our address detection relies on street-name detection.

3. MOPSI SEARCH ENGINE

3.1. Overall Scheme

MOPSI search engine consists of the following components:

1. User interface for web and mobile devices.
2. Core server software.
3. Geocoded address database.

These components and their relations will be discussed in the following subsections.

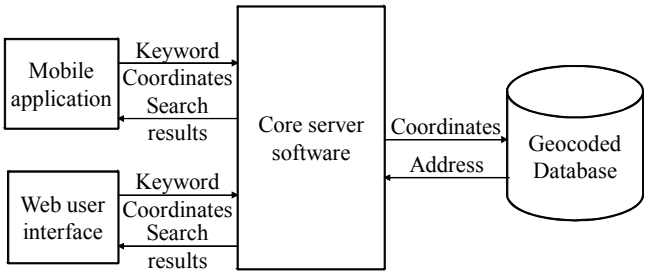


Fig. 2. Overall scheme of the MOPSI search engine.

3.1. User interfaces

The user can access the search engine either by using the mobile application (considering he has no access to a computer) or by using the web interface (considering he has no means of GPS positioning).

The mobile application is based on Java and compatible with most smartphones on the market today. It can use the internal GPS receiver of the phone or external Bluetooth device. The mobile application has also tracking and data collection modules, but these are not the focus of this paper. User can make a query to the MOPSI search engine using his current coordinates (if retrieved from the GPS module) or using previously stored coordinates. The results will be displayed as a list ordered by the distance to the user's location, or displayed on a map.

In order to use the location-based search engine, the mobile application has to be connected to the Internet. Communication between the mobile device with the corresponding client applications and the MOPSI search engine is implemented by HTTP protocol. The mobile application uses Java ME and is compatible with the S60 platform, which in turn supports information about positioning through Location API or through the Bluetooth protocol to connect to an external receiver. A C++ version for Symbian OS is currently under development.

The web interface is designed using PHP language, having AJAX capabilities and is used for accessing the location-based search engine from a computer. User can input the location by the street address or by selecting a location on the map. Search results are shown as a list, and on Google Maps, optionally with a route to the location.

3.2. Core server software

Considering that most web-pages contain street or postal addresses rather than explicit location information, our approach to the location-based search is to detect address portions from web-pages and use the distance from the user's location as an additional relevance criterion.

For implementing this concept we use an external search engine for performing query-based search and post-process its search results by extracting postal addresses, which are then translated into coordinates using the Geocoded database. The core server software performs these operations and is the main component of our location-based search engine.

The software (Fig. 3) has the following components:

1. Relevant municipalities detector
2. Page parser
3. Address and description detector
4. Address validator

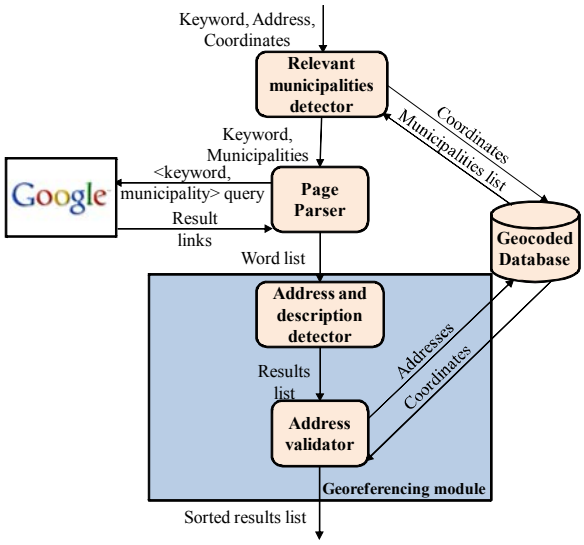


Fig. 3. Overall scheme of the MOPSI search engine.

The *Relevant municipalities detector* uses geocoded database to find all municipalities within a predefined range (e.g. 10 km) user's location.

The *Page parser* then uses an external search engine to perform <keyword, municipality> query for every municipality detected in the previous step. It downloads the web-pages found, strips the html tags, and extracts the text.

The *Address and description detector* searches for address blocks in the word list returned by the Page parser. It uses optimized prebuilt data structures (prefix trees) which contain all the street names for each municipality used for the search. It uses a simple text-matching algorithm to detect the address block, the description of the search result and the telephone number.

The *Address validator* verifies the addresses from the result list from the previous step using the Geocoded database. The addresses which are not found in the Geocoded database are discarded. The remaining search results are sorted by the distance to user's location, and filtered using a distance threshold in order to avoid excess information. The results are then shown as a sorted list on the mobile display.

3.3. Geocoded database

The process of assigning geographic coordinates is known as geocoding. Finding corresponding coordinates of a given street address requires a gazetteer, which is a geocoded street-name database that connects any given addresses to its exact locations (coordinates). Such databases are commonly available (although not necessarily free), and can be purchased for given (or specified) regions.

The *Geocoded database* is used to store all the addresses in the form of <street, number, municipality>, and their corresponding geographical coordinates. It is used every time an address needs to be converted into a location and vice versa. Furthermore, the Geocoded database is used for creating data structures (arrays of prefix trees) needed in the detection of the address blocks in the web-pages.

The speed and accuracy of the database is enhanced by using a database management system that implements *Open Geographical Information System* (OpenGIS) specification, or other spatial and location-data standards for faster query results. The database management systems that can be used include MySQL with spatial extensions, PostgreSQL with PostGIS or Oracle Spatial. In MOPSI, we use MySQL.

4. PROTOTYPE SEARCH ENGINE FOR FINLAND

We implemented a prototype application for location-based web search in Finland. The user interface of the engine was first a web page, accessible both from PC and multimedia phone, but we later developed Java-application for the mobile version. In PC (Fig. 1), user gives location as physical address, which is converted to coordinates in server-side using the Geocoded database. In mobile phone (Fig. 4), the software obtains user's coordinates by GPS, and then queries the server using HTTP Post. The mobile version also displays the nearest known address of the user.

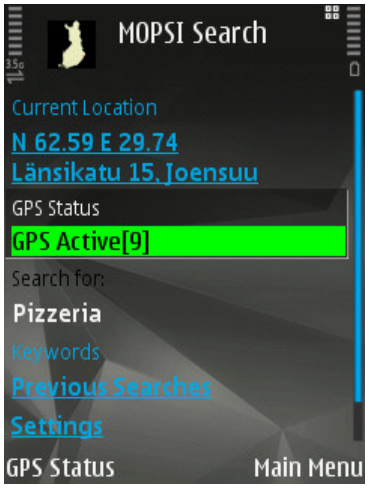


Fig. 4. Mobile application

4.1. Workflow of the prototype

When user inputs keyword(s), the server side application then searches all municipalities within a certain distance from user's location. The application creates *keyword-municipality* pairs for every commune and executes Google search using these as the search query. The contents of the first 10 returned web links are downloaded to the server and analyzed to find addresses. We use pattern matching technique that relies on *prefix trees* to detect the street names. If a match is found, the application searches for other typical address elements, such as street numbers, postal codes and municipal names followed by the candidate street name. If other elements are found, they are collected together as a street address candidate.

Unlike traditional web search, the search result is not necessarily the entire web page but can be only a part of it, most likely in case of a service directory that lacks a formal structure. In this case, the title of the entire page is not enough to identify the relevant part. However, the text closer to the address element can contain a description that could separate the search result from the rest of the web page, and possibly from several other non-relevant entities in the same page. In the current version, we extract description simply by taking a part of the text preceding the address.

The search result is built from the following: description phrase, address, web link, map link and distance from user's location. The current prototype uses free map service provided by Google Maps on server-side and commercial map service provided by the Finnish National Land Survey on mobile-side. Finally, the application

displays the results as a list ordered by the distance, see Fig. 5.

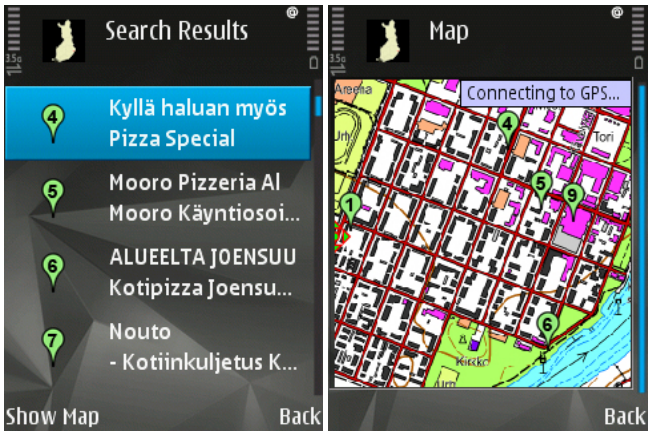


Fig. 5. Example of search results for keyword pizzeria.

4.2. Experiments

In order to test our prototype application, we simulated the following scenario: the user is in the center of an urban or rural municipality and his search is restricted to that municipality. His targets can be commercial (i.e. restaurant) or non-commercial (i.e. police station).

We selected 10 Finnish urban municipalities and 10 Finnish rural municipalities (see Table 1). The urban municipalities were selected according to their size, and geographical location was taking into account when selecting the rural municipalities. The search was performed using 10 test keywords which were divided into 5 commercial ones (hotel, restaurant, pizzeria, cinema, car repair) and 5 non-commercial ones (hospital, museum, police station, swimming hall, church).

Table 1. Municipalities used for testing

Type	Municipalities
Urban	Helsinki, Espoo, Lahti, Turku, Tampere, Jyväskylä, Vantaa, Oulu, Kuopio, Joensuu
Rural	Kuhmo, Ulvila, Lapua, Pieksämäki, Sodankylä, Forssa, Somero, Laihia, Kitee, Salla

In addition, the same queries were submitted to two other location based web services: Google Maps and Finnish Yellow Pages. According to [10], Google Maps uses multiple sources for providing the results. For instance, information submitted by local business owners or public directories, enhanced content such as reviews, photos submitted by users to various services, user generated content and other websites crawled. Yellow Pages, on the other hand, is a public directory of local business and the data is maintained and verified by users and administrators.

The search results are sorted by distance and filtered according to each municipality. Duplicates were manually removed and the distance was calculated using the gazetteer. There is no standard methodology for testing the overall relevance of the location-based search results considering relevance both by topic and distance. We therefore formulated our own criterion based on the evaluation methodology proposed in [6]. First, we compare the total number of search results from the tested services without considering their relevance. Secondly, we compare the relevance of the search results by evaluating them as (1) *relevant*, (2) *somewhat relevant* or (3) *not relevant*.

Table 2. Number of results for our test scenario

Query type	MOPSI prototype	Google Maps	Yellow Pages
Rural non-commercial	69	29	0
Rural commercial	245	92	189
Urban non-commercial	148	413	37
Urban commercial	1412	813	1337
Total number of results	2352	1405	1597
Overall mean relevancy	1.59	1.66	1.28
Overall std. deviation	0.84	0.89	0.54
Overall std. error	0.02	0.02	0.01

As shown in Table 2, our prototype provides more results than Google Maps or Yellow Pages, and is slightly more relevant (smaller the better) than Google Maps, but less relevant than Yellow Pages. With the exception of urban non-commercial queries, our prototype gives more results. Unsurprisingly, Google Maps has more results than Yellow Pages for non-commercial queries and less for commercial ones.

Contrary to Google Maps and Yellow Pages, our prototype relies on the relevance of the results of the external search engine. The search results are basically the addresses found in the results of the external search engine, whilst Google Maps has multiple criterions for relevance and Yellow Pages is controlled by human evaluators. We therefore compare relevance of the search engines on an average basis using a combination of two keyword categories for each comparison.

A comparison of rural municipals and non-commercial keywords resulted in our search engine (MOPSI) having mean value less than that of Google Maps (2.35 comparing to 2.48), whilst Yellow Pages did not return any results. This indicates a relatively high number of relevant results obtained by our search engine. The same method was used in Table 3, in which the results show a diverse number of relevant links by the MOPSI search.

Table 3. Mean relevance for our test scenario

Query type	MOPSI prototype	Google Maps	Yellow Pages
Rural non-commercial	2.35	2.48	0
Rural commercial	1.71	1.33	1.36
Urban non-commercial	2.20	2.17	1.59
Urban commercial	1.46	1.41	1.27
Overall mean relevancy	1.59	1.66	1.28

The results show that the relevance of MOPSI search engine is close to Google Maps, except for the rural municipalities with non-commercial keywords, for which we get higher number of results, but their overall relevance is smaller. Yellow Pages is the most relevant service, but having the lowest number of results (except for commercial urban keywords). In urban areas, the number of results by MOPSI search engine is close to Google Maps and Yellow Pages, whilst in rural areas it is higher.

4.3. Observations and known problems

One of the main problems is that the search can produce vast amount of irrelevant results. Mobile devices have restricted resources (data transfer bandwidth, small display) and often poor usability. The application should therefore be able to filter out flawed or less relevant data much better than a web-based application to make the browsing of the search results easier. Because of this, the current version downloads only a limited amount of links but further ideas to improve this would be desirable.

Another problem is that, unlike normal web searches, we allow the results include parts of web-page. This is very useful for finding services that do not have their own web page but exist in ad hoc service directories such as: http://www.pizza-online.fi/. However, an open problem is how to extract only the relevant part from a web page without any prior knowledge about its structure.

Despite previously mentioned problems, a lot of positive results are also achieved. When it comes to non-commercial services, the web pages are good repository of location relevant information. In rural areas of Finland where business is small, more services are found from web pages than from commercial databases.

5. CONCLUSIONS

The concept of location-based search engine was outlined, and its design issues and problems were discussed. MOPSI prototype implementation in Finland was demonstrated with qualitative comparison using typical search examples. The idea itself can be generalized worldwide although practical implementation would need a local gazetteer, or at least enough knowledge to be able to extract addresses from the web pages with reasonable accuracy.

The results indicate that the proposed approach has a lot of potential for practical applications. Most of the problems are related to technical matters and implementation issues. For instance, we use real-time search and page parsing whereas commercial solutions such as Google can use large computer capacity and avoid computational problems by pre-processing, huge storage (cache), and indexing.

6. REFERENCES

[1] Ahlers D. and Boll S. 2008. Retrieving address-based locations from the web. Int. Workshop on Geographic Information Retrieval, 27-34, Napa Valey, CA.

[2] Ahlers D. and Boll S. (2008b). Urban Web Crawling. ACM Int.workshop on Location and the web. Vol. 300, 25–32. Beijing, China.

[3] Buyukkokten O., Cho J., Garcia-Molina H., Gravano L. and Shivakumar N. (1999). Exploiting geographical location information of web pages. WebDB (Informal Proceedings), – dbpubs.stanford.edu

[4] Can L., Qian Z., Xiaofeng M. and Wenyin L. (2005). Postal address detection from web documents. Web Information Retrieval and Integration. Int. Workshop on Challenges in Web Information Retrieval and Integration, 40 - 45

[5] Hariharan G., Fränti P. and Mehta S. (2002). Data mining for personal navigation. SPIE Conf. on Data Mining and Knowledge Discovery, vol. 4730, 355-365.

[6] Jansen B.J., Molina P.R. 2006. The effectiveness of Web search engines for retrieving relevant ecommerce links, Inf. Processing & Management, 42 (4), 1075-1098

[7] Jones C.B., Abdelmoty A.I. , Finch D., Fu G. and Vaid S. (2004). The SPIRIT spatial search engine: Architecture, ontologies and spatial indexing. LNCS Lecture Notes in Computer Science, Springer.

[8] Lee H.C., Liu H. and Miller R.J. (2007). Geographically-Sensitive Link Analysis. IEEE/WIC/ACM Int. Conf. on Web Intelligence, Silicon Valley, CA, 628–634.

[9] McCurley, K.S. (2001). Geospatial mapping and navigation of the web. Int. Conf. on WWW, 221-229.

[10] Pasztor E., Egnor D. 2006. Generating structured information, US Patent Application 20060200478

[11] Tabarcea A., Hautamäki V. and Fränti P. (2010). Ad-Hoc georeferencing of web-pages using street-name prefix trees. WEBIST Int. Conf. on Web Inf. System and Technology, Valencia, Spain.

[12] Viola P. and Narasimhan M. (2005). Learning to extract information from semi-structured text using a discriminative context free grammar. ACM SIGIR Conf. on Research and Development in Inf. Retrieval, Salvador, Brazil, 330–337.

Paper P3

A. Tabarcea, V. Hautamäki, P. Fränti, "Ad-hoc georeferencing of web-pages using street-name prefix trees", Int. Conf. on Web Information Systems and Technologies (WEBIST'10), Valencia, Spain, vol.1, 237–244, April 2010

AD-HOC GEOREFERENCING OF WEB-PAGES USING STREET-NAME PREFIX TREES

Andrei Tabarcea
Faculty of Automatic Control and Computer Engineering, Technical University of Iaşi, Romania
tabarcea@cs.joensuu.fi

Ville Hautamäki
*Institute for Infocomm Research, A*STAR, Singapore*
vishv@i2r.a-star.edu.sg

Pasi Fränti
Speech and Image Processing Unit, University of Eastern Finland, Joensuu, Finland
franti@cs.joensuu.fi

Keywords: Search engine, LBS, Database, Prefix tree, Georeferencing, Mobile device, Location information, Personal navigation, WWW.

Abstract: A bottleneck of constructing location-based web searches is that most web-pages do not contain any explicit geocoding such as geotags. Alternative solution can be based on ad-hoc georeferencing which relies on street addresses, but the problem is how to extract and validate the address strings from free-form text. We propose a rule-based solution that detects address-based locations using a gazetteer and street-name prefix trees created from the gazetteer. We compare this approach against a method that doesn't require a gazetteer (a heuristic method that assumes that street-name has a certain structure) and a method that also uses data structures created from the gazetteer in the form of street-name arrays. Experiments using our location based search engine prototype (MOPSI) for Finland and Singapore, show that the proposed prefix-tree solution is twice as fast and 10% more accurate than its rule-based alternative and 10 times faster if an array structure is used when accessing the gazetteer.

1 INTRODUCTION

Location-based services (LBS) have become popular during recent years due to increasingly wide availability of GPS positioning in multimedia mobile phones. For instance, according to Nokia's own estimate more than half of their phones would include GPS by 2010-2012. In case of lacking GPS, positioning can also be based on cellular network or even on IP address for rough estimation. It is therefore expected that location-based services are emerging very fast to our everyday life via mobile phones and other consumer electronics.

Locations-based services such as YellowPages[1] , Google Maps[2] and Nokia Ovi Services[3] are traditionally based on databases where all entries have been explicitly georeferenced when stored in the database. An alternative approach has been outlined in (Hariharan et al., 2002) and (Fränti et al., 2010) based on web search and using ad-hoc georeferencing of the web-pages. We denote this approach as *location-based search engine* and emphasize it has seemingly small but significant distinction from traditional location-based services.

The bottleneck of this approach is that only very few pages have explicit georeferencing in form of geotaging, using address field or by other means. On the other hand, it is rather common that web-pages include street or postal addresses as free (non-tagged) text. According to (McCurley, 2001), most of relevant services (especially commercial ones) can be found in this way. The main problem however, is how to find valid address elements from the web-pages both reliably and efficiently.

[1] http://en.02.fi/yellow+pages/
[2] http:/maps.google.com/
[3] http://www.ovi.com/services/

In this paper, we propose a method for extracting street names based on street-name prefix tree and a gazetteer. A potentially relevant web-page (by its content) is first analyzed by extracting all potential street address elements. The hypothesized addresses are then validated by the gazetteer. The pages (or part of them) with validated address are attached by the exact location obtained from the gazetteer and a prototype solution can be found at the *MOPSI Search* website[4].

Extraction of the potential street-name portion of the address field in most languages is very regular. It usually ends to *way*, *drive*, *road*, or in Finnish language to a suffix such as *-katu*, *-kuja*, *-tie*. A simple *heuristic*, used earlier in (Fränti et al., 2010), performs a search for regular expressions with predefined endings (suffixes). However, not all street-names follow the predefined pattern and street-names that have a different suffix, such as *Neulavahe*, would not be detected. We therefore process all strings from the web-page since it can be done at the same cost when parsing the document.

Another problem is that we might detect as an address a portion of text that is not an actual address, causing a false detection. We therefore validate all hypothesized addresses by a gazetteer and discard the false detections. Our gazetteer is a *geocoded* database that contains geographical coordinates attached to address strings. As a side-product, the validation process provides the geocoding, i.e. converts the given address to a pair of coordinates. The process of recognizing geographic context is referred to as *geoparsing* and the process of assigning geographic coordinates to an address is known as *geocoding*.

One way to detect addresses from free form text is to build a classifier and let it detect addresses from the web-pages as in (Viola et al., 2005). However, customizing the classifier to other languages and countries takes a considerable work as new ground truth tagged text corpus must be created by hand. In our approach, no ground truth tagging is needed. The only things needed are a gazetteer and simple rules on how the street name appears in relation to other address fields. Efficient use of the gazetteer is possible because we know the user's current location and its interest area consists only on those services that are close to him. Therefore, we can build fast access structure to that partial gazetteer.

Matching of the potential address strings can be done *brute force* by comparing each word in the document to the retrieved table of street-names. However, this can be rather inefficient if the database is large. We therefore use the prefix tree as a search structure, which is critical for the performance of the matching. A set of prefix trees is constructed from all street-names in a given municipality and the ones in the proximity of the user's location are used. The proposed solution is faster and more accurate than the heuristic solution alone and much faster than the brute force.

2 RELATED WORK

There has been a lot of progress in location-based search during the last years, starting with commercial services like Google Maps, Yahoo! Local[5], Bing Maps[6] and Yellow Pages, or with research projects such as (Jones et al., 2004), (Morimoto et al., 2003) and (Ahlers et al., 2008a).

A spatially-aware search engine (SPIRIT) was developed in (Jones et al., 2004) using geographic ontology, textual and spatial indexing of web-pages. In (Morimoto et al., 2003), a system for extracting geographic information from web-pages gathered by crawling programs is presented, whilst the system in (Ahlers et al., 2008a) relies on web crawling which is targeted to create topical web indices. Our approach differs from these since we don't rely on explicit indexing, but apply ad-hoc georeferencing by detecting postal addresses from free-text.

A categorization scheme of web queries is defined in (Gravano et al., 2003) based on global or local geographic locality. In this view, our search engine handles local queries. In (Wang et al., 2005), three types of locations from web resources are defined: *provider location* (physical location of the provider who owns the web resource), *content location* (the geographic location of the content) and *serving location* (the geographical scope it can reach). Our goal is to search for the content location.

Methods of detecting tagged location of a web resource are found in (Buyukkokten et al., 1999) and (McCurley, 2001). In (Buyukkokten et al., 1999), "whois" records are analyzed and phone numbers of network administrators are used jointly with zip code and area database to assign coordinates to so-called Class A and B domains and to determine the "globality" of a web-site. In (McCurley, 2001), the sources for geospatial context are classified as being for the hosts of a web-page (usually found in "whois" databases and the way the traffic is routed

[4] http://cs.joensuu.fi/mopsi/

[5] http://local.yahoo.com/

[6] http://www.bing.com/maps/

on the Internet), and for its content (postal addresses and codes, telephone numbers, geographic feature names). Additional geographical information is found from hyperlinks and meta tags.

In (Hill et al., 1999), a gazetteer is defined as a geospatial dictionary of geographic names and its minimum components as a geographic name, a geographic location represented by coordinates, and a type designation. Our gazetteer is a geocoded database which contains postal addresses and their corresponding coordinates.

On the other hand, *name entity recognition* without gazetteers is discussed in (Mikheev et al., 1999) and it turns out to work well with people and organizations, but bad with locations. Our solution of postal address detection without a gazetteer (the heuristic method) is much simpler and exploits structural characteristics of postal addresses.

The majority of location-based search systems use gazetteers. For example, the system in (Amitay et al., 2004) uses a three-step process: spotting, disambiguation and focus determination. Our address detection algorithm uses the first two steps.

In (Borges et al., 2007), an ontology-based approach that extracts geographic knowledge is presented. The address is divided into 3 parts: basic address (street and building number), complement (optional, may be neighborhood name) and location identifiers (phone number, postal code, city name). It can be complete, incomplete or partial. The address recognition consists of the processes of geoparsing and geocoding, which uses a gazetteer as described in (Souza et al., 2005). A spatial index (geoindex) is built for each page. The geoparsing process relies on a set of rules and creating patterns implemented as regular expression from four elements: *basic address*, *postal code*, *phone number* and *city/state*. Our approach is different in a sense that it relies merely on text matching, although our heuristic uses matching via regular expression.

In (Can et al., 2005), a syntactic approach to postal address detection is proposed. It consists of two steps: a vision-based text segmentation and a syntactic pattern recognition method. The text segmentation analyses the html tags and detects cue blocks (for the purpose of indications, annotation, and explanation) and body blocks (main text body content). The recognition of postal address relies on calculating the confidence of the detected blocks, which in turn is based on tokenization of the words, which uses city names, state names, street and organization suffixes, but not street names. Our approach is simpler, as we filter out all the html tags before the matching process, and different, as our address detection relies on street-name detection.

In (Cai et al. 2005), location-based data is retrieved by recognizing postal addresses. The method is ontology-based conceptual information retrieval combined with graph matching. The concepts (knowledge/address elements) in a document are identified and linked in the graph by semantic relations. A set of rules is used on the graph and graph matching methods are used to compute similarity and map concept nodes. The concept set used is actually a gazetteer.

In (Silva et al., 2006), a graph-ranking algorithm for assigning the geographic scope of a web-page is proposed. Georeferencing is aided by a geo-ontology knowledge base, which uses a set of rules, relationships and heuristics.

In (Lee et al., 2007), regular expressions are used to detect patterns of typical address elements and database to validate results. The detected street name candidate is then retrieved from the address database to compare all street names for a specific area. In case of a positive match, house numbers are searched and the final address is validated through the database. Our heuristic address detection algorithm is similar to this solution.

In (Ahlers et al., 2008b) a geoparser that identify address level location is built using a database rather than rely on metadata or other structured annotation. The database used by the geoparser contains postal codes, city names, street names, and also every city-postal code combination is also used for validation. The address detection assumes that the address blocks have a certain structure, and that there are certain dependencies between the address elements. We utilize the idea of identifying a number of address elements in our geoparsing algorithm, and validating the address by geocoding it. However, our contribution is that we use own geocoded database and rely on street-name detection based on prefix-trees data structures, while (Ahlers et al., 2008a) uses freely available geocoders.

3 LOCATION-BASED SEARCH ENGINE

3.1 System Description

A location-based search engine is one of the practical applications of the proposed ad hoc georeferencing of web-pages. The basic idea behind the location-based search engine has been presented already in (Hariharan et al, 2002) and the first prototype application, *MOPSI Search*, has been

described in (Kuittinen, 2006) and (Fränti et al., 2010). It consists of the following components:

1. User interfaces for mobile devices and web.
2. Core server software: search engine and database administrator.
3. Geocoded street-name database with spatial indexing: the gazetteer of the project.

Our approach to the location-based search is to use an external search engine for query-based searching and to post-process the search results provided by that engine, extracting the street or postal addresses. These addresses are then translated into coordinates using a geocoded street-name database for result validation and ranking.

The core server software (Figure 1) is the key component in the system as it implements the georeferencing module. It consists of:

1. Relevant municipalities detector
2. Page parser
3. Address and description detector
4. Address validator

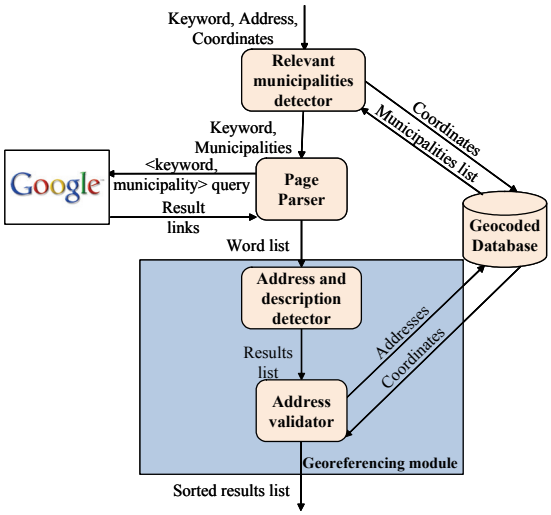


Figure 1: Architecture of the core server software.

The *Relevant municipalities detector* uses the Geocoded database to find all the municipalities that are within a predefined search range (e.g. 10 km square) centered in the user's location.

The *Page parser* uses an external search engine to perform a *<keyword, municipality>* query for every municipality detected at the previous step. It downloads the web-pages found, strips the html tags, and extracts the text.

The *Address and description detector* searches for address blocks, descriptions and telephone numbers in the web-pages found by the Page parser. The *Address validator* uses the Geocoded database to convert street addresses from the previous step to geographical coordinates, then validates and filters the addresses according to a distance threshold. The validated addresses are used to georeference the web-pages.

3.2 Street-address Detection

Current prototype uses a rule-based pattern matching algorithm, which starts with the detection of street-names (Figure 2).

```
StreetNameDetection(words)
{
WHILE i < count(words) DO
   {
   IF words[i] = street name THEN
         {
         Search for street number, postal code and other
         address elements near words[i].
         IF address elements found THEN
               {
               Create address block
               Get coordinates using Geocoded database
               IF coordinates found THEN
                  Add address block to address list
               }
         }
   i = i + 1;
   }
}
```

Figure 2: Pseudocode for address detection.

If a street-name is found, an address-block candidate is constructed by detecting other typical address elements, such as street numbers, postal codes and municipal names. The application looks for these elements in the vicinity of the found street-name. Variations about how addresses are constructed are taken into account. An address-block candidate is validated using the Geocoded database. If the detected address has corresponding coordinates in the database, it is considered as valid; otherwise it is discarded.

Since a plain address without any additional information is not a useful search result alone, the application extracts descriptive information relative to the address. The current implementation simply extracts a part of the text preceding the address as descriptive information. This information is used to create a search result, which is composed of the following: *descriptive phrase*, *telephone number* (if detected), *address*, *web link*, *map link* and *Euclidian distance* between user and the target location.

3.3 Street-name Detection

Street-name detection is the starting point of the address detection. One practical issue is the availability of a gazetteer, as it can be used as a street-name database. Such databases are commonly available, but not necessarily free for commercial purposes. Our application uses a gazetteer of National Land Survey of Finland, and for Singapore, we use the street data from OpenStreetMap[7].

The methods that don't use gazetteer usually assume that a street-name has a certain structure, whilst the methods which use a gazetteer rely on fast word matching. For comparison, we implemented both approaches: a *heuristic method* that does not use a gazetteer, and two text matching methods that use data extracted from a street-name database.

3.3.1 Heuristic Method

Our heuristic method relies on regular expression matching. The structure of most of the addresses has certain particularities. For example, street-names can start with the same prefix or end with the same suffix, they can be in the vicinity of standard words and they are always followed or preceded by a number. In this case, the address block detection also starts with the street-name detection and relies on a set of regular expressions.

According to our experiments, this approach has very good results for Finnish street-names, because most of them end in words like “katu” (street), “tie” (road), “kuja” (lane) or “polku” (path) and has the advantage that it does not need any database or other data structures to store the street-names, and it is reasonably fast.

The accuracy may vary from country to country and the main disadvantage is that the method has to be tailored for every country and language because of the various ways an address block can be constructed. For example, in Finland it is common that the address block has <*street-name, street number, postal code, municipality*> structure, with the street type (e.g. road, lane, street, avenue) as a suffix, whilst in Singapore the <*street number, street name, street type, municipality, postal code*> is more common, but more variations exist. For example, in Singapore a street-name can be written using abbreviations such as Av. instead of Avenue, which is much rarer in Finnish addresses.

[7] http://www.openstreetmap.org/

3.3.2 Brute-force Matching using Street-name Arrays

A brute-force text matching method checks every word in a web-page against a street-name database. We use an optimized brute-force solution that checks the word against all street-names in the proximity of the location the query is made, for example the street-names in the municipality where the user location is.

We use arrays of street-names that are created beforehand from the gazetteer. Each array is used to store all the street-names in a municipality and the search is done using language-specific functions. Since our search engine is written using PHP scripts, we use the *array_search* and *in_array* functions optimized to find an object in an array.

3.3.3 Text Matching using Street-name Prefix Trees

This method uses prefix trees of street-names, which are created beforehand using the information in our gazetteer. The gazetteer used in the project is a geocoded database which contains all the postal addresses in Finland with their corresponding coordinates. For Singapore, the prefix trees were constructed from street names extracted from freely available map data. Statistical data about both gazetteers are detailed in Table 1.

Table 1: Gazetteer statistics.

	Finland	**Singapore**
Number of municipalities	410	1
Total number of street names	92 572	573
Number of streets per municipality	474	573
Average street name length	11.6	6.1
Total size (MB)	2 982	0.18

In general, the postal addresses are not unique, and the same street-name can be found in many cities. A prefix tree is therefore built for each municipality and just the prefix trees corresponding to the search area are loaded during a search.

4 STREET-NAME PREFIX TREE

The prefix tree (or *trie*) is a fast ordered tree data structure used for retrieval (Navarro et al., 2002). The prefix tree stores a collection of strings, indexed from the beginning of a word (i.e. prefix). The root node represents an empty string and its children

store the first letter of the string. The same principle is applied at every level of the tree so that the internal nodes describe all the sub-strings (prefix) of the particular string. The recursive version of the algorithm is presented in Figure 3.

```
ConstructTrie(streetnames)
{
    Create empty node root
    FOR i = 1 TO count(streetnames) DO
        {
        AddString(root, streetnames[i], i);
        }
}
AddString(node, string, index)
{
    IF (length(string) > 0) THEN
        {
        IF (string[0] is not the key of a child of node)
        THEN
            Create new node child with the value
string[0]
        ELSE
            Set child as the child of node with the key
            string[0]
        AddString(child, substring(string, 1), index);
        }
    ELSE  //node is a terminal node
        {
        node.index = index;
        }
}
```

Figure 3: Prefix tree generation pseudocode.

The nodes of the prefix tree can also have values associated with them, although the only values that are commonly used are the values of the leaf nodes and the values of some inner nodes. In our case, we use the values to mark the end of a street name in the tree structure. Usually, only the leaf nodes are the end of a street name, but if a street name is a prefix of another street name, then an inner node can also be the end of a street name.

Dictionary search is one of the most common applications of the prefix tree. It traverses the prefix tree until it reaches a leaf node, or when a node does not have any children whose key contains the desired letter. The recursive version of the prefix tree search algorithm is presented in Figure 4.

In our implementation, we create prefix trees from street-names of each municipality. Therefore, the street-name detection becomes a dictionary search using prefix tree. Because the Finnish street-names usually end with a limited number of suffixes, the names were introduced in the prefix-tree in reverse order and the search in the prefix-tree is done starting from the last letter. Figure 5 gives an example of a prefix tree pre-computed from the Geocoded database.

```
FindString(root, string)
{
    IF (strlen(string) == 0) THEN
        RETURN root.index;  //we have reached last
    node
    ELSE
    {
        IF (string[0] is not the key of a child of root)
        THEN
            RETURN -1; //string is not found
        ELSE
        {
            Set child as the child of root with the key
            string[0]
            RETURN FindString(child,
        substring(string,1))
        }
    }
}
```

Figure 4: Pseudocode of the Prefix Tree search.

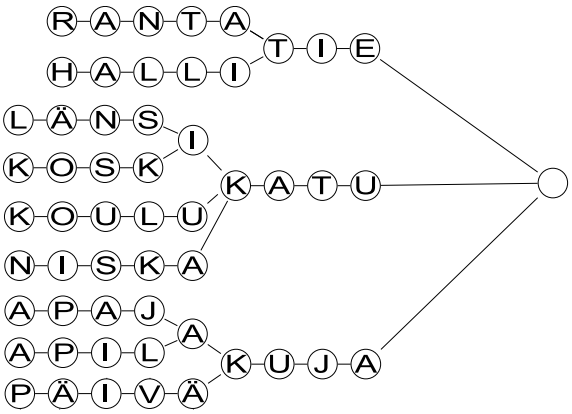


Figure 5: A sample prefix tree built from street-names.

Table 2 summarizes the computed Prefix Trees for Finland and Singapore. It highlights the fact that the gazetteer obtained from the OpenStreetMap is not complete. One of the main advantages of using prefix trees or other pre-built data structures to access street-name data from the gazetteer is the fact that the storage size is reduced (from 3 GB to 74 MB) and the gazetteer is used only for address validation and geocoding.

Table 2: Prefix tree statistics.

	Finland	Singapore
Maximum tree depth	34	14
Average tree depth	12.7	7.4
Average tree width	105	167
Average number of nodes per tree	2338	2335
Total size (MB)	74.4	0.18

5 EXPERIMENTS

We tested the proposed MOPSI location-based search engine using 20 different search locations and 10 keywords to construct <keyword, municipality> queries. We downloaded the content of the first 10 search results for each query of Google search engine and the downloaded content was used as data input for the MOPSI prototype.

The search locations were divided into 2 groups: 10 rural 10 urban municipalities (Figure 6), and the test keywords were divided into 5 commercial and 5 non-commercial ones (Table 3).

Table 3: Keywords used for experiments.

Commercial	hotel, restaurant, pizzeria, cinema, car repair
Non-commercial	hospital, museum, police station, swimming hall, church

The addresses detected by each method were validated using our geocoded database. The size of the downloaded data in the rural and urban municipalities is 13.9 and 11.2 MB, respectively.

Table 4 shows the average time for address detection and the number of detected addresses for the considered municipalities. The average time is calculated per query over all searches. According to the results, the proposed Prefix Tree method is considerably faster than the Brute Force method, and 2-3 times faster than the Heuristic approach, which does not use the gazetteer. Typical search times of the Prefix Tree are less than 1 second per query.

The detected addresses are validated using our gazetteer. The accuracy (number of validated addresses) of the Brute Force and Prefix Tree methods are higher than that of the Heuristic method. The biggest difference between urban and rural municipalities is that the number and the density of streets are much larger in the urban municipalities and, therefore, the methods using gazetteer (Prefix Tree and Brute Force) are slower in rural municipalities. Nevertheless, the Prefix Tree method is the fastest even in this case.

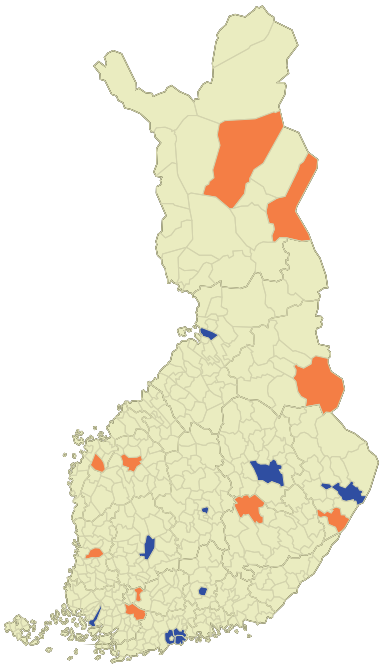


Figure 6: Locations used for experiments. The urban locations (blue): Espoo, Helsinki, Joensuu, Jyväskylä, Kuopio, Lahti, Oulu, Tampere, Turku, Vantaa; the rural locations (orange): Forssa, Kitee, Kuhmo, Laihia, Lapua, Pieksämäki, Salla, Sodankylä, Somero, Ulvila.

The results also show that, for the Heuristic solution, street density and city size do not affect much the search times. In case of Prefix Tree method, the average search time is somewhat bigger in rural municipality (0.51 vs. 0.87 seconds). The Brute Force method is affected most by the street density as the search times in the street array are bigger than the ones in the prefix tree, resulting in more than 3 times longer search time in urban areas.

In total, the proposed Prefix Tree method is twice as fast as and 10% more accurate than the Heuristic method, on average. It reaches the same accuracy than the Brute Force search but using only 10% of the processing time.

6 CONCLUSIONS

Our main goal to design a gazetteer-based street address detector was to increase the accuracy in comparison to the fast heuristic method that was used in the earlier implementation (Fränti et al., 2010). This goal was achieved, as the proposed

prefix tree solution is 57% faster and 10% more accurate, on average, than the heuristic solution. In comparison to Brute Force, it is 10 times faster.

The resulting solution improves the speed and quality of web-page georeferencing and removes one bottleneck for creating efficient location-based search engine as the prototype *MOPSI search.*

Table 4: Average search times for the address detection.

Method	Time (s)	Standard deviation	Number of validated addresses
Rural municipalities			
Brute-Force	3.01	2.43	3.7
Heuristic	1.54	1.15	2.5
Prefix Tree	0.51	0.35	3.7
Urban Municipalities			
Brute-Force	10.18	7.11	19.8
Heuristic	1.70	1.24	18.6
Prefix Tree	0.87	0.85	19.8
Total			
Brute-Force	6.59	6.40	11.8
Heuristic	1.62	1.20	10.5
Prefix Tree	0.69	0.68	11.8

ACKNOWLEDGEMENTS

The research was supported by EU/EAKR and the work of Ville Hautamäki by the Academy of Finland, under project 131298.

REFERENCES

Ahlers D. and Boll S. (2008a). Retrieving address-based locations from the web. *Int. Workshop on Geographic Information Retrieval*, 27-34, Napa Valey, CA.

Ahlers D. and Boll S. (2008b). Urban Web Crawling. *ACM Int.workshop on Location and the web*. Vol. 300, 25–32. Beijing, China.

Amitay E., Har'El N., Sivan R. and Soffer A. (2004). Web-a-where: geotagging web content. *ACM SIGIR Conf. on Research and Development in Information Retrieval*, Sheffield, UK, 273–280.

Borges K., Laender A., Medeiros C. and Davis Jr. C. (2007). Discovering geographic locations in web pages using urban addresses. *ACM Workshop on Geographical Information Retrieval. Lisbon, Portugal*, 31-36.

Buyukkokten O., Cho J., Garcia-Molina H., Gravano L. and Shivakumar N. (1999). Exploiting geographical location information of web pages. *WebDB (Informal Proceedings)*, – dbpubs.stanford.edu

Cai W., Wang S. and Jiang Q. (2005). Address Extraction: Extraction of Location-Based Information from the Web. *Web Technologies Research and Development - APWeb 2005*, Volume 3399/2005, 925-937

Can L., Qian Z., Xiaofeng M. and Wenyin L. (2005). Postal address detection from web documents. Web Information Retrieval and Integration. *Int. Workshop on Challenges in Web Information Retrieval and Integration*, 40 - 45

Fränti P., Kuittinen J., Tabarcea A. and Sakala L. (2010). MOPSI Location-based Search Engine: Concept, Architecture and Prototype. *ACM Symposium on Applied Computing*, Sierre, Switzerland.

Gravano L., V Hatzivassiloglou V. and Lichtenstein R. (2003). Categorizing web queries according to geographical locality. *Int. Conf. on Information and Knowledge Management*, New Orleans, LA, 325–333.

Hariharan G., Fränti P. and Mehta S. (2002). Data mining for personal navigation. *SPIE Conf. on Data Mining and Knowledge Discovery*, vol. 4730, 355-365.

Hill L., Frew J. and Zheng Q. (1999). Geographic names: The implementation of a gazetteer in a georeferenced digital library. *D-Lib Mag.*, January 1999, 5 (1)

Jones C.B., Abdelmoty A.I. , Finch D., Fu G. and Vaid S. (2004). The SPIRIT spatial search engine: Architecture, ontologies and spatial indexing. *LNCS Lecture Notes in Computer Science*, Springer.

Kuittinen J. (2006). Using location information in search engines. *MSc thesis*, Univ. of Joensuu (in Finnish)

Lee H.C., Liu H. and Miller R.J. (2007). Geographically-Sensitive Link Analysis. *IEEE/WIC/ACM Int. Conf. on Web Intelligence*, Silicon Valley, CA, 628–634.

McCurley, K.S. (2001). Geospatial mapping and navigation of the web. *Int. Conf. on WWW*, 221-229.

Mikheev A., Moens M. and Grover C. (1999). Named entity recognition without gazetteers. *Conf. on European Chapter of the Association for Computational Linguistics*, Bergen, Norway, 1–8.

Morimoto Y., Aono M., Houle M.E. and McCurley K.S. (2003). Extracting spatial knowledge from the web. *Symposium on Applications and the Internet*, 326–333.

Navarro G. and Raffinot M. (2002). *Flexible Pattern Matching in Strings*. Cambridge University Press.

Silva M.J., Martins B., Chaves M., Afonso A.P. and Cardoso N. (2006). Adding geographic scopes to web resources. *Computers, Environment and Urban Systems*, 30 (4), GIR, 378–399.

Souza L.A., Davis C.A. Jr.,Borges, K.A.V., Delboni T.M. and Laender A.H.F. (2005). The role of gazetteers in geographic knowledge discovery on the Web. *3rd Latin American Web Congress*, 9.

Viola P. and Narasimhan M. (2005). Learning to extract information from semi-structured text using a discriminative context free grammar. *ACM SIGIR Conf. on Research and Development in Information Retrieval*, Salvador, Brazil, 330–337.

Wang C., Xie X., Wang L., Lu Y. and Ma W.Y. (2005). Detecting geographic locations from web resources. Workshop *on Geographic Information Retrieval*, Bremen, Germany, 17 – 24.

Paper P4

A. Tabarcea, N. Gali, P. Fränti, " Location-aware information extraction from the web" (manuscript), 2015

(submitted to Data and Knowledge Engineering)

Location-aware information extraction from the web

Andrei Tabarcea, Najlah Gali and Pasi Fränti

Speech and Image Processing Unit
School of Computing, University of Eastern Finland
Länsikatu 15, FI-80101, Joensuu, Finland
Email: {tabarcea, najlaa, franti}@cs.uef.fi

Abstract

A large part of the multimedia data on the internet are nowadays generated with devices that automatically annotate them with location information, but free-form content such as websites does not implicitly contain any geographical information. This is the biggest challenge for building a location-aware search engine. In this paper, we study how to extract location-aware information from the web. The key challenges are to detect location from a web page and to extract relevant information related to that location. We detect locations by identifying postal addresses using freely available gazetteers. Additional information for summarizing the search result are title and representative image, which we mine from the content by using a simple rule-based approach utilizing the structure of the web page. The results can be used to personalize search results for mobile users that are relevant to their location.

Keywords Location-based search, address detection, web mining, prefix trees

1. Introduction

The volume of geospatial data has been increasing over the years as more and more devices have access to the Internet and positioning technology [1]. A large part of these multimedia data are nowadays generated with devices that automatically annotate them with location information, but free-form content such as websites does not implicitly contain any geographical information. Mobile search engines allow users to find information anytime and anywhere, however, the search performance is influenced by the type of mobile device and the user's context, which is influenced by aspects such as location, profile, previous activity, time of year and social network [2].

Location is an important factor in personalizing web search. This is because the content of a website has a limited area of interest that influences its relevance [3]. *Location-based search* aims at finding a business or place of interest around a specific geographical location. This is supported by search engines that support geographical preferences [4], and the relevance of a search result depends on the distance between the user-specified location and the location of the service [5]. Location-based search changes the search from web-oriented to service-oriented, which makes it a more challenging task due to two reasons. First, it is not enough just to find a relevant web page for the user, as in traditional web search. Instead, we also need to detect the location that the web page is relevant to. Second, we need to extract information from the web page. This is either a simple summary of the content (such as title and image), but, and in case of service directories, we need to extract the part belonging to that particular service. It requires both identification of geographical data and automatic information extraction from web pages.

Locations can be embedded explicitly into websites as geographical coordinates (usually latitude and longitude), both in source code as HyperText Markup Language (HTML) meta tags named *geo-tags*[1], as <*address*> tags and in the content of the websites as plain text. According to studies such as [6], [7] and our own experiments, very few web pages contain explicit locations. Therefore locations are mostly embedded implicitly as *geographical references* in many ways such as: postal addresses, place names, description in natural language and driving directions. Identifying geographical references and associating a website with one or multiple locations is a process called *geo-referencing*. A particular case of geo-referencing is *geo-tagging*, which means the assignment of geographical coordinate metadata to multimedia such as photos, videos and websites, and *geo-coding*, which means finding geographical coordinates from other types of location data, such as street addresses or postal codes.

[1] http://www.w3.org/2003/01/geo/

In this paper, we propose a method for extracting locations from web pages, and complement it by extracting a title and a representative image for each search result. Our goal is to integrate the processes of geo-referencing, geo-coding and geo-tagging into a unified location-based system that can extract information relevant to the user's location. This information has to be close to the user's location, related to the keywords provided by the user, and extracted from the content of the websites found by standard search engines. The prototype of the proposed method is implemented in the MOPSI platform, see [8] and [9]. Besides location-based search, MOPSI offers tools for collecting, processing and displaying location-based data, such as photos or trajectories, along with social media integration. A typical workflow of a location-based search application that also applies to our system is shown in Figure 1. Using the location of the user and a set of keywords, the application detects and validates locations, identifies service information and presents a ranked list of the results consisting of the following: short text summary (title), link, image thumbnail, address and distance.

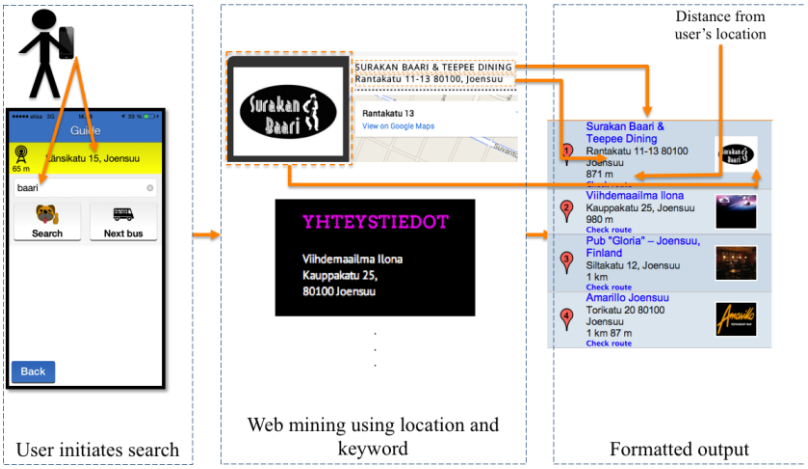


Figure 1 Web mining using location and keyword

Our proposed system is based on a meta-search approach similar to [10]. We use the results of an external search engine which is not location-aware, and post-process the search results by extracting locations from the provided websites. The meta-search approach has the advantage of allowing us to find relevant websites without crawling, indexing and storing websites, but it makes our system dependent on the relevance of other search engines and system vulnerable to changes in the external search engines we use.

We extract a title and representative image for each detected location to provide a brief summary of the search results similarly as in [11]. We extract the title and the image using DOM tree representation of the provided websites as in [14], contrary to many methods that concentrate just on plain text extraction. We do not associate a single location to a single website but allow the same page to contain multiple locations. Finally, we build an ordered search result list, which we rank by distance from the user's location. The search results are general and not limited to a theme or a type, such as products or companies.

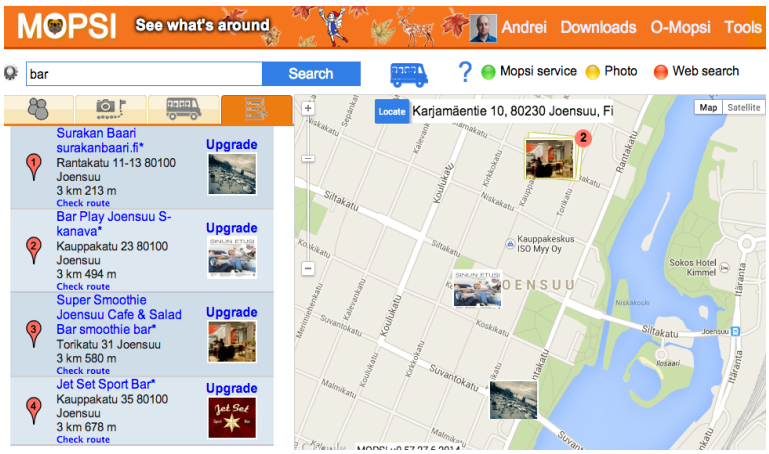


Figure 2 Web interface of MOPSI search

We find locations by detecting postal addresses. Our approach is similar to that of [12] as we rely on gazetteers in the address detection step, but we also use contextual information in order to detect the entities that are related to the detected locations. The address elements are identified individually and then aggregated in order to build an address candidate, in a method similar to [13]. The difference is that we start the aggregation with the street name and we detect the additional elements in the words close to the street name. We identify address elements using a gazetteer and regular expressions and we validate the detected addresses by using the gazetteer to find their respective coordinates. We optimize the gazetteer search by using fast prefix-tree based text search. This method was tested against heuristic and brute-force approaches having better results [14], but it does not allow spelling mistakes, name variations or abbreviations. Our approach is lightweight as it does not require training, but it is dependent on the quality of the gazetteer data and the addresses in the web pages.

Because our approach is service-oriented, we need complete addresses that can be converted into coordinates, not just the areas, towns or other more general descriptions. An advantage of this is that disambiguation is not a big problem because a complete address has very small chance of repeating in different areas and no chance of representing a non-geographical entity.

In respect to existing commercial services such as Google Maps[2], Bing Local[3] or Yahoo Local[4], our goal is the same: provide location-relevant information to the user. However, these applications are mainly based on commercial databases, user input and pre-collected data resulted from web crawling, and only exploit the results in real-time web search to a limited extent. In information retrieval, a location-based search engine is an alternative approach to traditional location-based services based on fixed databases. It aims at utilizing the location of the user, but without restriction to any fixed location-based service.

Our system is flexible and can detect locations from any set of web pages, not just the results of an external search engine. The implementation is not limited to specific geographical areas, although it is dependent on the accuracy of the gazetteer data we use. For this purpose, we use a gazetteer build using OpenStreetMap data, which is available for most countries.

The rest of the paper is organized as follows. In Section 2, we study related work regarding location-based search and address detection and compare it to our methods. Section 3 outlines the prototype meta-search that has been implemented in MOPSI, along with address validation and content extraction (title and representative images). The experiments are described in Section 2, and conclusions are drawn in Section 3.

2. Related work

Location-aware search is an alternative approach for traditional location-based services using pre-collected data. An example is a location-based search engine for the Singapore area [15] that finds services by using the location of the user, and filters for area names, building names, landmark types, business names and business categories using a catalogue of businesses and landmarks. Our approach does not rely on pre-collected databases of services but is based on a real-time search and automatic detection of locations.

According to [16], there are three types of locations that can be inferred from websites: *provider location* (where the owner of the page is), *content location* (where the content is pointing to) and *serving location* (the area for which the website is relevant). Provider location is detected using a set of heuristic rules such as referred frequency, URL levels and spatial positions of the address strings in the website. Locations are obtained by extracting all geographical references using probabilities to measure the reliability of each source. These are calculated using power and spread of a geographical reference [17], and using *country-state-city* location hierarchy in the form of a geographic tree. Serving location is found in a similar way, but additionally using links between the pages and user visit logs. Our application is focused on detecting the content location.

To make a search engine location-aware, the key challenge is to detect the location that the website relates to. Websites are designed to be browsed by humans and contain geographical references that are complex, informal, diverse, ambiguous and difficult to be processed by a computer [18]. There is no widely used standard on how to code the location on a web page and the concept of the location itself is not uniquely defined. It can be exact geo-coordinates (latitude and longitude), their approximation by GPS, postal addresses that have the accuracy of the houses, area, town or country, or even driving directions or plain text description. However, one can adopt an unsupervised approach to extract the implicit

[2] http://maps.google.com
[3] http://www.bing.com/local
[4] http://local.yahoo.com

locations from websites by several different strategies as outlined in [19]: text matching using gazetteers, rule-based linguistic analysis, text matching based on regular expressions, identification of host location and reading geographic meta-tags. We use text matching based on gazetteers and regular expressions because most of the locations from web pages are postal addresses. Furthermore, according to [20], detecting addresses is more accurate when using a gazetteer.

Early methods of assigning locations to web resources in [21] and [22] rely on identifying the host location, which is the location of the owner or the administrator of the website. A single location is assigned for each website by querying its *Whois*[5] records for the address and telephone number of the network administrator. This method is expanded in [23] by additionally using hyperlinks, meta tags and postal addresses as sources for location information.

Geo-tags and <address> tags are the most direct way to define locations for a website, but they are rarely used. However, postal address is the *de facto* standard to define geographical references on websites [24]. They can be converted into locations by services providing *geo-coding*. We use two freely available geo-coding services: OpenStreetMap and geo-coded database for Finland with publicly available data. Despite of the occasional unavailability of the service and data inaccuracy, free geo-coding services such as OpenStreetMap or Geonames[6] are best suited for applications that require near-accurate (but not necessarily exact) geo-tagging because they have extensive coverage [25]. Other applications such as public health or underground cable locations (safe-to-dig.com, kaivuulupa.fi) would require much higher precision of data.

One of the biggest challenges when identifying postal addresses is the ambiguity of terms, which can be between locations with the same name (*geo/geo ambiguity*) or between location names and non-geographical entities (*geo/non-geo ambiguity*) [23]. Both types of ambiguity can be resolved using heuristic rules. The algorithm in [26] attempts to resolve the geo/geo ambiguity using an algorithm similar to Google's Page Rank [27]. In our case, ambiguity is not a problem because we detect complete postal addresses, which, in most of the cases, are unique within the same country.

There are several approaches in detecting addresses, such as training a classifier [28] or syntactic pattern recognition [29], but most of the address detection methods, such as [10], [30], [31], [32], [33], [34] and ours roughly follow the same process. First, address elements are identified using regular expressions and ontologies, which in most of the cases are gazetteers with hierarchical relationships between elements. Second, address candidates are built and they can be aggregations of the detected elements, graphs or blocks of elements. Finally, the address candidates are validated using a pattern matching method such as regular expression validation, gazetteer matching or graph matching. The patterns are built either by training or by using a set of rules, relationships and heuristics.

There has been several works reported in the literature for location-aware search. A personalized mobile search engine enhanced with capturing users' preferences in the form of click-through data is proposed in [10]. The users' preferences are captured in the form of concepts, which are modeled as ontology and separated into location concepts and content concepts. The search engine also considers users' GPS location and uses content and location entropies to balance the content and the location concepts.

In [35], location tags are assigned to websites with a precision up to street level. Words are extracted from the website and checked against free gazetteers (Geonames and OpenStreetMap) using the Aho-Corasick algorithm [36] for string matching, which is a fast dictionary-matching algorithm. Validation and disambiguation of the locations is done by using all the geographical references from the text to create a context. The method was reported to outperform a commercial solution (Yahoo! Placemaker) despite of detecting only 60% of the locations correctly. The authors demonstrate the applicability of their method by describing three practical applications derived from their work: location-aware web surfing through a mobile device, browsing by using nearby tags and location tagging through social networks. Our address detection approach is similar because we also use free gazetteers and a fast string matching algorithm, in our case prefix tree search.

A system that is capable of handling geographical queries of the triplet of <theme> <spatial relationship> <location> is described in [37]. It handles spatial relationships such as *inside, near, north--of, south-of* and *geo-references*, stores and indexes websites using both pure text and spatial indexes. Using both types of indexes enables a full set of geographical query operators, graphical query formulation and ranking of results according to conceptual as well as spatial criteria. Geographical ontologies are used for query expansion and for disambiguating the queries and the extracted locations. The system relies on web crawling and pre-processed indexes and combines textual and geographical relevance. Locations

[5] http://www.whois.net
[6] http://www.geonames.org

are detected using a gazetteer lookup approach, which is enhanced with context rules and additional name lists used for filtering.

A system that leverages on contextual information, which can be used in the named entity recognition and disambiguation steps is described in [12]. A set of location evidence is built, updated and used to provide geographical contextual evidence.

A method to identify address data by combining patterns and gazetteers from free sources such as OpenStreetMap is used in [13] to identify companies from websites. The websites are pre-processed by removing HTML tags, extracting text, line splitting, tokenizing and part-of-speech tagging. Single attributes *postal codes*, *city names*, *street names*, *street numbers*, *company names* are identified on the pre-processed data using regular expressions and heuristic rules. The attributes are then aggregated starting with the company name. We are using a similar method to aggregate address elements, but we have a different method of identifying service and company names, which uses scoring based on visual appearance and distance from the detected address. Furthermore, our goal is more general as we do not restrict our search just to company websites, although most of our results are businesses and companies.

A knowledge-based web-mining tool that adopts a geospatial ontology, a rule based screening algorithm and inductive learning for automated location retrieval is described in [38]. Address detection is customized to discover the locations of emergency service facilities; other detected addresses are discarded. Our method is simpler, as it does not require any training or learning. Compared to [38], we aim at a broader scope for our application without limiting its use to a certain type of service.

A method for extracting postal addresses and associated information using sequence labeling algorithm is introduced in [11]. Unlike most existing methods, postal addresses are not detected by using gazetteers. Instead, addresses are detected by pre-processing data with a named entity recognition tool, extracting features from text and training models using support vector machines and conditional random fields. Pattern mining is applied to identify the boundaries of address blocks and to extract the associated information for each detected address. The associated information is defined as information that refers to the detected addresses and allows for better comprehension.

The method in [39] was designed to extract product data from company websites but applies to our application as well. The leaf nodes of the DOM tree are analyzed and used to generate semantic information vectors for the other nodes, which in turn are used to generate a maximum repeating semantic vector pattern. The generated pattern is used to detect product data regions and to build product templates, which are used along with a semantic tree matching technique to identify product information. This method is effective, and can also be applied for detecting locations and associated information, but is limited to the websites that contain a list of items with a clear and repeating pattern. We also use the leaf nodes of the DOM tree, but our approach is different: we mark the nodes that contain location information and detect company data by ranking the nearby nodes based on how close and how visually different they are to the node that contains location information.

An alternative to the DOM tree is the visual block tree-based approach of extracting data records proposed in [40]. The visual block tree [41] is composed of rectangular data blocks. These data blocks are filtered, clustered and regrouped to identify data records. The visual block tree is built using the visual features that humans can capture from websites; it uses the layout of the page and attributes such as fonts and background color. The data records are extracted using the structure of the visual block tree and the visual features of its elements.

3. Extracting Location-aware Information

3.1 MOPSI Prototype

We have implemented a prototype meta search engine that works in the Mopsi platform on the web (cs.uef.fi/mopsi/) and on smart phones [9]. The system takes search keyword and user location as input, and outputs an ordered list of search results that contains the following information: rank, title, web link, address, representative image and distance to the location (see Figure 1). We next define all components needed to implement this. The search workflow is detailed in Figure 3.

The search starts by finding websites that are relevant to the location and the keywords provided by the user. This is done by the *website provider*, which takes the user location as input and converts it to a city using our gazetteer based on OpenStreetMap data. The city and the keywords are then input to a conventional search engine as <*keyword*, *city*> phrase, and outputs a list of websites found. The city is added to increases the probability for the search result being related to the area of the user. Otherwise, we would need to process a much higher number of websites that are relevant by their content but not by their location. This is an unavoidable drawback of the meta search approach.

For the search, we use two different search engines: BOSS API by Yahoo![7] and Custom Search by Google[8], as they allow third parties to build search products by using the infrastructure of their search engines. The website provider relies on the relevance of the results of the external search engine. It uses the keyword and the city provided by the user and it does not expand the query in any other way. To keep the computation reasonable, only the first 10 results of the query are the output of this module.

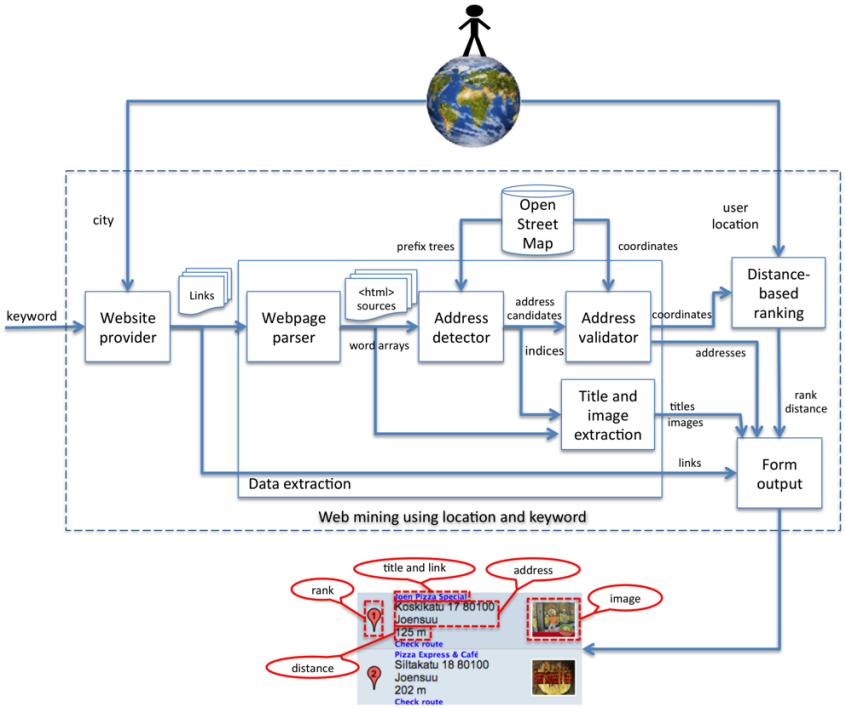


Figure 3 Location-based search workflow

The websites processed by the *data extraction* module. It consists of mostly self-built components based on Document Object Model (DOM), which is a tree structure representation of a website. The DOM tree is used in all stages: for detecting the websites and the addresses, extracting the titles such as services names, and extracting representative images that correspond to the detected services. The output is a list of the found address candidates and the associated information.

First, the *website parser* downloads all the detected websites, and converts their HTML sources into DOM tree representation. The *address detector* then searches for postal addresses using a text matching algorithm based on street name prefix trees [14]. It operates on the text nodes of the DOM tree to identify the following address elements: street name, street number, postal code and city. Address candidates are constructed by aggregating address elements that are close to each other in the text of the website. The prefix trees are constructed on demand for each city using our own gazetteer for Finland, see [42], and OpenStreetMap for the rest of the world. The address detector outputs a list of address candidates and marks the nodes that hold valid locations.

Title and image extraction module processes the marked nodes to extract a representative title and image for each detected address. It uses the same DOM tree and searches for text and images in the sub-trees of the nodes that contain location information. The output is a list of geo-referenced entities that contains the information described in Figure 3. The address candidates are validated at the same time by the *address validator*, which uses our own gazetteer built on OpenStreetMap. This is a geo-coding service that validates the addresses detected at the previous steps and converts them to geographical coordinates.

Finally, we aggregate all the attributes as entities in the search results list, which is displayed to the user sorted using *distance-based ranking*. All the information is assembled by the *form output* as a list of search results. The coordinates

[7] http://developer.yahoo.com/boss
[8] https://developers.google.com/custom-search

are also used for displaying the results on the map and for computing the distance from the user's location. The postal address candidates that are not validated by the geo-coding services are discarded.

3.2 Parsing Web Pages

HTML documents are considered to be semi-structured data, which are neither raw nor strictly typed [43]. HTML documents do not conform to a formal data model, do not have a fixed schema, and their elements typically hold information solely for rendering. They are not completely unstructured because their HTML tags and tree structure can be used to guide data extraction.

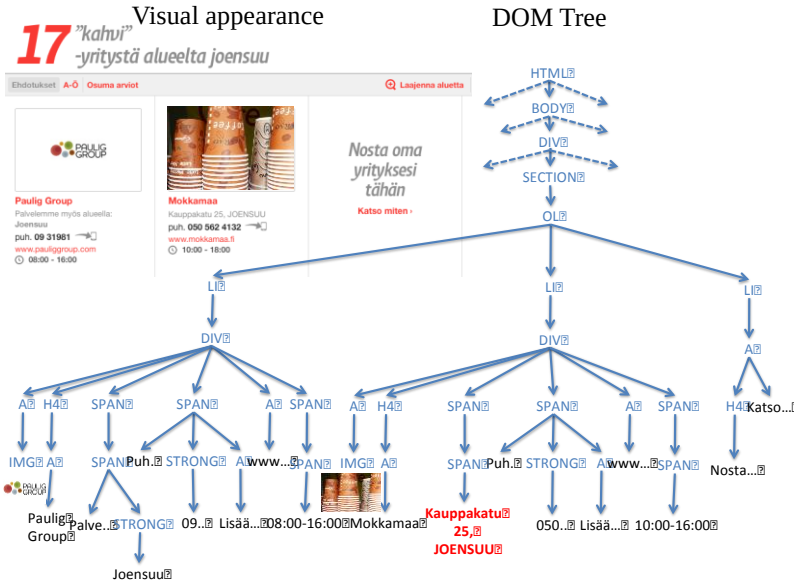


Figure 4 Part of a web page that contains location information: visual appearance and DOM tree

A HTML document has a DOM representation, which is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents[9]. A DOM tree is composed from HTML elements and their parent-child relationship, having the <html> as the root of the tree. Figure 4 shows a simplified example of a HTML page, where we display just the sub-tree that contains the location. This DOM tree is used for address, title and image extraction. Figure 5 shows the result for this particular case. The three components are detailed separately in the following sub-sections.

Figure 5 The title and the representative image for a detected address

[9] http://www.w3.org/DOM

3.3 Address Detection and Validation

Typically, a postal address includes a subset of the following elements: *street name*, *street description*, *street number*, *postal code*, *neighborhood*, *city*, *region* and *country*. Figure 6 shows as few examples from addresses found in our service data set in Mopsi.

Kaislakatu 8, 80130, Kanervala, Joensuu, Finland
Torikatu 25, 80100 Joensuu, Finland
Parppeintie 6, 82900 Ilomantsi, Finland
Aleksanterinkatu 25, 15140 Lahti, Finland
Vene 18, 10140 Tallinn, Estonia
Carrer de la Marina, 266-270, Barcelona, Spain
2 Rue Pasteur, 06500 Menton, France
Pulchowk Rd, Lalitpur 44600, Nepal
20 Chả Cá, Hàng Đào, Hoan Kiem District, Hanoi, Vietnam
East Coast Park Service Road 1, Singapore

Figure 6 Address examples

We use a rule-based text-matching algorithm to separately identify each type of the address element (see Figure 7). We use the text nodes from the DOM tree of the web page to extract the text from their associated sub-tree. This allows us to find addresses that are spreading through several nodes (for example if one element is bold) or through tables. The text is segmented into words and each word is verified as to whether it is an address element. In order to identify a postal address, we first build an address candidate by aggregating the address elements that are close to each other in the text of the considered sub-tree. The maximum distance for two address elements to be considered close is 10 words. We define that an address candidate must contain at least the following three elements: street name, street number, and postal code or city name. We validate the candidates using our gazetteer and mark all the nodes that satisfy the criterion that they contain validated locations, and will use them later in the content extraction stage.

```
nodes = GetDOMTree(url)
cities = RetrieveCities(nodes)
FOREACH city IN cities
  prefixTree = RetrieveStreetPrefixTree(city)
  FOREACH node IN nodes
    AddressDetection(node, prefixTree)

AddressDetection(node, prefixTree)
  words = ExtractSubtreeText(node)
  FOREACH word in words DO
    IF prefixTree CONTAINS word THEN
      Search for number, postal code, city name near word using regular expressions
      IF number, postal code or city name found THEN
        Aggregate found elements into address candidate
        Get coordinates of address candidate using gazetteer
        IF coordinates found THEN
          Add address candidate to address list
          Mark node as containing address information
```

Figure 7 Pseudocode for address detection

The first step of the address detection algorithm is to identify the city names from the content of the web page. This is done by using a city-name dictionary. The next step is to identify the street names using the data from each city that were detected at the previous step. For each considered city we build a prefix tree of street names [14]. An example of a prefix tree is shown in Figure 8, but for the sake of simplicity it does not contain real street names, but shorter words that can be endings of street names.

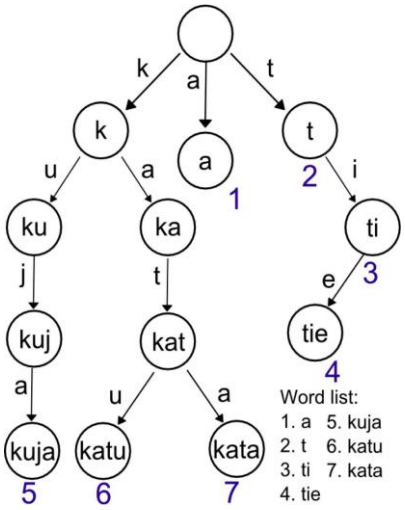


Figure 8 Prefix tree example

In order to build a street name prefix tree for a city, we require a gazetteer that contains all addresses in the region we are interested in. To make our location-based solution widely available, our gazetteer uses free data from the Nominatim[10] project, which is based on OpenStreetMap. Because building a prefix tree for each city is a long process, we generate prefix trees on demand when a search is made and we cache the generated prefix trees on our server.

Every word on a web page is checked as to whether it is a street name, by searching for it in the prefix tree. After the street names have been identified, we store their positions on the page and use regular expressions to identify street numbers, city names and postal codes close to the detected positions. An address element is considered close to a street name if the distance to it is maximum 10 words. Finally, when we identify address entities containing at least the minimum set of elements: {street, number, city} or {street, number, post code}, we use it as an address candidate and validate it using the geo-coding functions provided by Nominatim. If the geo-coding service returns valid coordinates, we consider the address as valid and use the coordinates to calculate the distance from the user's position.

We tested our system using the Finnish services collected by MOPSI users (see Section 2). Finnish addresses have concatenated street names and street descriptions, with street name being the prefix and street description being the suffix. Examples of street names are: *Kauppa-* (market), *Alexanterin-* (Alexander's) or *Kaisla-* (Reed) and descriptions: *-katu* (street), *-tie* (road), *-polku* (way) or *–kuja* (alley). Finnish addresses have a fixed order: street name, street number, neighborhood, postal code and city, but neighborhood and postal code are optional. Although the websites we use to test our methods are Finnish, our address detection algorithm is not customized for Finnish addresses and does not use a predefined order of address elements.

The proposed address detection is based on the method described in [14] with a few improvements. First, we are browsing the DOM tree of the web page and mark the nodes that contain addresses instead of extracting plain text from the web page. This is because the words and paragraphs do not always appear in the same order as on the web page, especially when the web page contains tables or columns. Using the DOM tree also allows exploiting visual or structural features from the web page that would not be possible from the plain text. This also helps the detection of content such as service titles and images. The second improvement is that we used a local database of Finnish addresses and extended its scope by using OpenStreetMap data.

The chosen address detection method (especially the street name detection), relies on the accuracy of the OpenStreetMap gazetteer because it needs to find the exact string match. It supports multi-word street names and addresses because the address elements are indexed as strings that can also contain spaces, but it does not support any variation on the order of the words in a street name, mistyped words or abbreviations. Although the websites we use to test our methods are from Finnish services, our address detection algorithm is not tailored for a specific country or language, however it only detects addresses that follow a structure similar to Western European addresses. Our method does not use a predefined order of address elements, therefore it tolerates variations in the order of elements, as we search for elements both before and after the detected street names, but it could produce false positives since it does not consider

[10] http://nominatim.openstreetmap.org

any semantic relationship between the elements. Supporting abbreviations depends on the data in the gazetteer, which needs to have a separate entry for each possible abbreviation and is difficult to generate and maintain. Furthermore, the text matching method we use only supports exact matching and does not detect misspelled street names or municipalities. In future work, our text matching method can be improved with methods such as term normalization [7].

3.4 Title Extraction

Extracting a representative title for a service in a website is not trivial. By default, the title tag could be used, but according to our experiments, it provides useful information only in case of 72% of the websites. In the other cases, the title tag contains useless data such as "*Homepage*", "*Contact*", or long descriptions including slogans or advertisements such as "*Joensuu Center | Intersport - Sport to the people*". Search results contain also a lot of service directories where the title belongs to the owner of the website rather than to the service in the content. We use the method proposed in [44], and review its main idea briefly for the sake of completeness.

We consider two types of websites separately: *individual service* and *service directories*. A page for an individual service contains a single address and its content is related to a single service, business or place. A service directory is a website that lists services offered by others. They have one or more common attributes: type, location or owner. Because the structure of these two categories of pages is different, we consider these two cases separately. We use a rule-based classifier to detect to which of these two types a given website belongs. It analyzes the web link and the content of possible menu lists on the page. The page is concluded as service directory if it fulfills a criterion that measures the diversity of the content of the menu items: the more heterogeneous the content, the more likely the page is a service directory.

For individual services, we use the content of the title <*title*> and meta tags <*meta name=title*>. We use three different rules. The first one gives higher weight to phrases in the beginning and the end of the string. This is a useful rule of thumb when having very long title strings. The second rule analyzes how often the same phrases are used in header tags, and the third rule whether the phrase also appears in the web link. The candidate phrase with the highest score is selected.

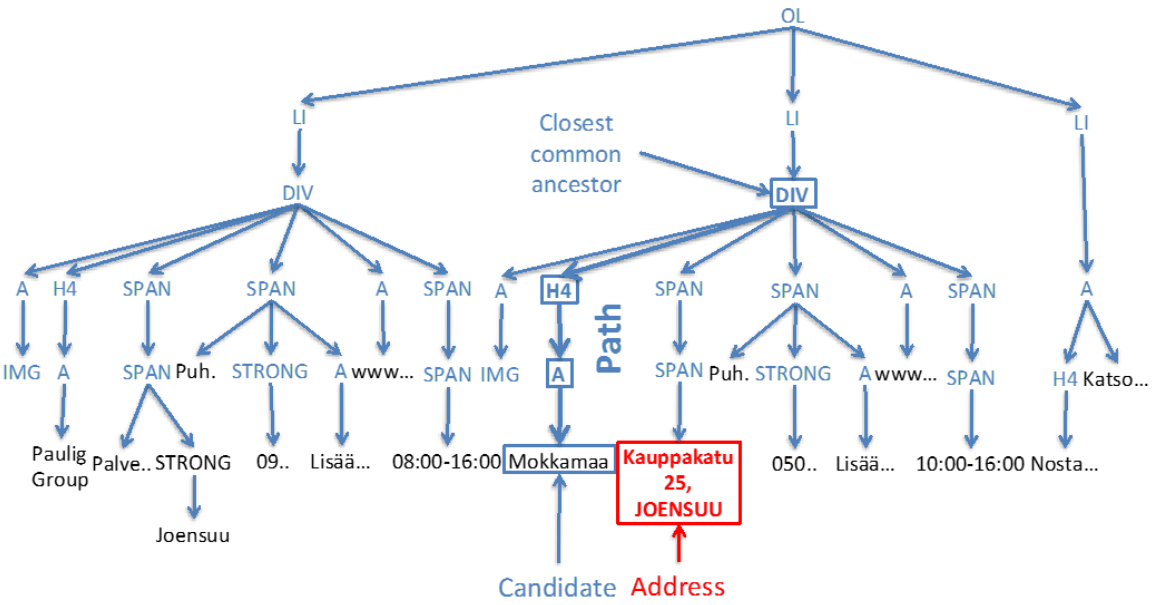
Table 1 Scores for HTML tags

Tags	Score
H1	+7
H2	+6
H3	+5
H4, A	+4
H5, H6, B, STRONG	+3
I, EM	+2
Others	0

With service directories, the title tag or meta tags are not usually related to the service but rather to the service directory itself. In this case, we consider text nodes as potential title candidates. We analyze the DOM tree starting from the node containing the validated address, and measure the distance to the common ancestor of the address node and the candidate text node. The closer the distance the more likely the candidate relates to this address. Visual cues such as emphasis (Table 1) and color (Table 2) are also used. The perceptual color difference (CIE 2000 [45]) between the *color* and *background-color* attributes of the candidate text node and the postal address node is scored between [0, 10]. In order to extract the CSS attributes shown in Table 2, we render the page and calculate the CSS for each considered node. However, we only use features that can be computed directly from the source code of the page instead of for example analyze the layout of the page. All candidates are scored based on the distance and the visual factors; the highest scored candidate phrase is chosen.

Table 2 Scoring CSS attributes

CSS Attributes	Score
color, background-color	+ perceptual color difference (0 to 10)
font-size	+ (node font size - address node font size)
font-weight	+3 if bold or >500
text-transform	+5 if uppercase



1.

Figure 9 DOM sub-tree that contains the address of a service

For example, in Figure 9, the detected address is "*Kauppakatu 25, Joensuu*", and the candidate text node is "*Mokkamaa*". They have a DIV as the closest common ancestor. HTML tags along the path from the candidate text node to the closest common ancestor node are scored as shown.

1.1 Image Extraction

Images are used on websites more than any other type of online content because they can transfer information to the user in a quick and efficient way. Although a large number of images are embedded into websites, many of them are less relevant to the content of the website, such as advertisements, navigational banners, icons and images that serve as section headings. A solution is therefore needed to ignore the irrelevant images and find a good representative image for the website.

Representative images are important in many other applications, especially in cases when bandwidth limitation restricts the total number of images that can be retrieved, or when building a visual category in which a single image must represent an entire category of documents and their associated content. We next briefly review the method that was presented in [46].

Figure 10 Image categories

All images found in the HTML, CSS and JavaScript source code of the website are considered as candidate images. We classify images into categories based on their expected functionality: *representative*, *logo*, *banners*, *advertisement*, *formatting and icons*. We choose the image with the highest score from the representative category when available. If there are no images in this category, we proceed to the next category in the priority shown above, until an image is found. The categories are detailed below, and an example is shown in Figure 10.

- *Representative*: images that are directly related to the content of the website.
- *Logos*: recognizable images that identify the service provider (company or institution)
- *Banners*: images placed usually either above or below the website, or besides the content. They are generally used for decoration. Headers and footers are classified in this category
- *Advertisements*: images that promote products or services not directly related to the website.
- *Formatting and icons*: images that are used to enhance the visual appearance of the website. These can be spacers, bullets, borders, backgrounds, or pictures used purely for decoration. These also include small images that serve a functional purpose, such as links to the home page, or icons used for changing language.

Table 3 Rules used for image categorization

Category	Features	Keywords
Representative	Not in other category	
Logo	Parent is h1 or h2 (not used)	logo
Banner	ratio > 1.8	banner, header, footer, button
Advertisements		free, adserver, now, buy, join, click, affiliate, adv, hits, counter
Formatting and icons	width < 100px height < 100px	background, bg, sprite

For classifying the images we use the rules shown in Table 3*Table 3*. In most categories, a predefined set of keywords is used. If any of these are found in the image link, or in the class name of the <*img*> tag and of the parent element, then the image is assigned to this particular category. Banners and formatting are also categorized according to image size and aspect ratio. An example of the features is shown in Figure 11. Note that the categories are overlapping. If the same image meets the requirements of multiple categories, it will be classified using the order of priority shown above. However, an image can belong to the class of representatives only if it does not belong to any other category.

src	http://www.ravintolakreeta.fi///images/banner.jpg
alt	--
title	--
from	css
format	jpg
width	945
height	202
size	190,890 px
aspect ratio	4.67
parent tag	
class	header

Figure 11 Image features we used

2. Experiments

We test the proposed approach for address and data extraction using the MOPSI dataset. It contains services submitted by users of the application and confirmed by administrators. Each service has the following fields: title, website, coordinates, postal address, image, keywords and free-form text description. Except title, postal address and coordinates, all the other fields are optional, and we therefore only consider the services that also have a website. The titles have been manually entered by the users, and the geo-location is obtained either by GPS positioning (in case of mobile input) or manually tuned on map (in case of web input). Although there has been no common practice how to input the data, the most relevant data (title and address) exist for all services, and can be used as ground truth against which the results of the data extraction results can be compared.

The dataset contains 364 services from Finland. The services have been stored using both Finnish and English language. Some users have systematically used both languages (kirkko, church), but not all. Administration has been quite limited and cursory; it mainly constituted of removing duplicate and invalid entries, fixing obvious errors, and sometimes improving on the content. The following are the most popular words appearing either in the title or in the set of keywords (Finnish-English): Kahvila-Cafe (64-9), Ravintola-Restaurant (61-17) Loma-Holiday (45-1), Kampaamo-Barber (31-21) Sauna (25), Kirkko-Church (20-20), Pizza (20), Baari-Bar-Pub (17-6-6), Market (15). Other common keywords include Post, Chinese, Kebab, Pharmacy, Park, Bank. Occasionally the following are also found: Frisbeegolf, Parckour, Golf, Locksmith.

For the experiments, we downloaded all the websites for each of the 364 services between in the period 29.4.-15.5.2015, and analyzed their content. We first studied whether a location exists either as a geo-tag or a postal address. If it exists, we compare it against the ground truth address stored within the service. Having services input by users manually verified by the administrator ensures that the address associated with the service is the correct one, but its website might contain a different address on no addresses.

Firstly, we checked for geo-tags and <address> tags. The results in Table 4 show that just 10 (3%) of the websites have a geo-taged location, only 13 (4%) have an address tag, but in 145 (40%) of the cases our method found addresses in the text content of the website, out of which 82% were accurate addresses. Considering their low number, we manually checked addresses found from geo-tags and address tags and found that they were all valid. Among all websites, our algorithm found the correct address 119 (33%) times, and different addresses 26 (7%) times.

The high number of websites containing postal addresses was expected because we considered websites that advertise businesses, local services or tourist locations. The low number of geo-tags and useful addresses tags emphasizes the importance of data mining methods for identifying locations. This is the spirit of the web: users do not follow uniform formalism on a wider scale, and even professional website designers use their freedom. The web is built for people, not for search engines.

Table 4 Results of the address detection

	All websites	**Location found in…**			**Address is…**		
		Geotag	Address tag	Text content	Correct	Different	None
Number	364	10	13	145	119	26	219
Percentage	100%	3%	4%	40%	33%	7%	60%

For each correctly detected address, we tested the title extraction algorithm of Section 3.4. For comparison, we report the result when the title was taken from the *title tag* as such (baseline method) or the corresponding *meta tag* (meant for robots). A result is considered correct if the similarity of the strings (extracted and the ground truth) are higher than a threshold 0.5. The results in Table 5 show that the correct title is found 29% of the cases, if we exclude the websites in which we do not detect any address. This is significantly lower than that of the baseline method (72%). In [44], a similar study was conducted as a normal web mining task without considering the location aspect. The method described in Section 4.2 was applied only to service directory pages, having 65% true detection. The reported results for Google and Yahoo! were 67% and 64%, respectively. The results here are lower because we applied our method to all website types. Our method performs worse on websites of individual companies or places because it gives high scoring to text that is close to the postal address. In these types of websites, the title is not always repeated close to the address and our algorithm detects wrong titles such as: “Post address:”, “Company info:” or “You can contact us at:”. The quality of the results can

be improved by using website classification and the methods described in [44], in which we first classify a website as a single service, brand or service directory and depending on its type we use either the method described in Section 3.4 or a method that is based on segmenting the title tag and comparing it with the headers and the lists on the web page. The method tested in this paper shows clear limitations for websites that are not service directories, but our experiments in [44] show more promising results.

Table 5 Results of the title and image extraction

	All websites	**Title found in…**		**Image found on…**		
		Title tag	Text content	WebIma	Facebook	Google+
Number	298	214	42	185	143	166
Percentage	100%	72%	29% out of 145	62%	48%	56%

The accuracy of the image extraction was measured by comparing how many times the image extraction component (WebIma) selects the same image that was marked as ground truth by the volunteers using our data collection tool [34]. Different services can have the same website, so we filtered out duplicates and unavailable websites, after which 298 web pages remained. The results show that in 62% of the cases the image extraction component found the correct image, which outperforms the comparative results of Google+ (56%) and Facebook (48%). The results can be only indirectly compared to those reported in [46], where a more extensive set of 1032 websites were evaluated. The accuracy reported there was 64%, which was significantly higher than that of Google+ (48%) and Facebook (39%).

3. Conclusions

Detecting locations and associated information from websites is the key challenge in creating a location-aware search engine. Our paper describes the components needed to build such an engine, including automatic address detection, title and image extraction. The results demonstrated that in 40% of the websites used in our experiments, a standard search engine can be converted to location-aware by these data analysis components alone. Since the test was limited to service websites only, it is expected that results with other types of websites, such as blogs and news stories, the success rate would be lower.

References

[1] C. A. Patterson, R. R. Muntz, and C. M. Pancake, "Challenges in location-aware computing," *Pervasive Comput. IEEE*, vol. 2, no. 2, pp. 80–89, 2003.

[2] C. Liu, P.-L. P. Rau, and F. Gao, "Mobile information search for location-based information," *Comput. Ind.*, vol. 61, no. 4, pp. 364–371, 2010.

[3] P. N. Bennett, F. Radlinski, R. W. White, and E. Yilmaz, "Inferring and using location metadata to personalize web search," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 135–144.

[4] A. Markowetz, Y.-Y. Chen, T. Suel, X. Long, and B. Seeger, "Design and Implementation of a Geographic Search Engine," in *Eighth International Workshop on the Web and Databases (WebDB 2005)*, 2005.

[5] S. Yokoji, K. Takahashi, and N. Miura, "Kokono Search: A Location Based Search Engine," *10th International World Wide Web Conference (WWW10)*. 2001.

[6] I. Vänskä, "Using location information in web documents," University of Eastern Finland, 2004.

[7] D. Ahlers and S. Boll, "Retrieving address-based locations from the web," in *Proceeding of the 2nd international workshop on Geographic information retrieval - GIR '08*, 2008, p. 27.

[8] P. Fränti, J. Kuittinen, A. Tabarcea, and L. Sakala, "MOPSI location-based search engine," in *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, 2010, p. 872.

[9] P. Fränti, A. Tabarcea, J. Kuittinen, and V. Hautamäki, "Location-based search engine for multimedia phones," in *2010 IEEE International Conference on Multimedia and Expo, ICME 2010*, 2010, pp. 558–563.

[10] K. W. T. Leung, D. L. Lee, and W.-C. Lee, "Pmse: A personalized mobile search engine," *Knowl. Data Eng. IEEE Trans.*, vol. 25, no. 4, pp. 820–834, 2013.

[11] C. H. Chang and S.-Y. Y. Li, "MapMarker: Extraction of postal addresses and associated information for general web pages," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, 2010, vol. 1, pp. 105–111.

[12] T. Qin, R. Xiao, L. Fang, X. Xie, and L. Zhang, "An efficient location extraction algorithm by leveraging web contextual information," in *proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, 2010, pp. 53–60.

[13] S. Schmidt, S. Manschitz, C. Rensing, and R. Steinmetz, "Extraction of Address Data from Unstructured Text using Free Knowledge Resources," in *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, 2013, p. 7.

[14] A. Tabarcea, V. Hautamäki, and P. Fränti, "Ad-hoc georeferencing of web-pages using street-name prefix trees," in *Int. Conf. on Web Information Systems and Technologies (WEBIST'10)*, 2011, vol. 1, pp. 237–244.

[15] F. S. Tsai, "Web-based geographic search engine for location-aware search in Singapore," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 1011–1016, Jan. 2011.

[16] C. Wang, X. Xie, L. Wang, Y. Lu, and W.-Y. Y. Ma, "Detecting geographic locations from Web resources," *Proc. 2005 Work. Geogr. Inf. Retr.*, pp. 17–24, 2005.

[17] J. Ding, L. Gravano, and N. Shivakumar, "Computing geographical scopes of web resources," *Proc. 26th Int. Conf. Very Large Data Bases*, pp. 545–556, 2000.

[18] G. Shi and K. Barker, "Extraction of geospatial information on the Web for GIS applications," *IEEE 10th Int. Conf. Cogn. Informatics Cogn. Comput.*, pp. 41–48, 2011.

[19] Y.-H. Hu, S. Lim, and C. Rizos, "Georeferencing of Web Pages based on Context-Aware Conceptual Relationship Analysis," 2006.

[20] A. Mikheev, M. Moens, and C. Grover, "Named Entity recognition without gazetteers," in *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics* -, 1999, pp. 1–8.

[21] O. Buyukkokten, J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar, "Exploiting Geographical Location Information of Web Pages," in *In Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*, 1999, pp. 1–6.

[22] C. Watters and G. Amoudi, "GeoSearcher: Location-based ranking of search engine results," *J. Am. Soc. Inf. Sci. Technol.*, vol. 54, no. 2, pp. 140–151, 2003.

[23] K. S. McCurley, "Geospatial mapping and navigation of the web," in *Proceedings of the 10th international conference on World Wide Web*, 2001, vol. WWW '01: P, pp. 221–229.

[24] D. W. Goldberg, J. P. Wilson, and C. A. Knoblock, “From text to geographic coordinates: the current state of geocoding,” *URISA J.*, vol. 19, no. 1, pp. 33–46, 2007.

[25] A. J. Florczyk, F. J. López-Pellicer, P. Muro-Medrano, J. Nogueras-Iso, and F. J. Zarazaga-Soria, “Semantic selection of georeferencing services for urban management,” *J. Inf. Technol. Constr.*, vol. 15, pp. 111–121, 2010.

[26] Q. Zhang, P. Jin, S. Lin, and L. Yue, “Extracting focused locations for web pages,” in *Web-Age Information Management*, Springer, 2012, pp. 76–89.

[27] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web.,” *Tech. report, Stanford Digit. Libr. Technol. Proj.*, 1999.

[28] P. Viola and M. Narasimhan, “Learning to extract information from semi-structured text using a discriminative context free grammar,” in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, vol. 272, pp. 330–337.

[29] L. Can, Z. Qian, M. Xiaofeng, and L. Wenyin, “Postal address detection fromweb documents,” in *Web Information Retrieval and Integration, 2005. WIRI’05. Proceedings. International Workshop on Challenges in*, 2005, pp. 40–45.

[30] K. A. V. V. Borges, A. H. F. F. Laender, C. B. Medeiros, and C. a. Davis Jr., “Discovering geographic locations in web pages using urban addresses,” in *GIR ’07 Proceedings of the 4th ACM workshop on Geographical information retrieval*, 2007, pp. 31–36.

[31] W. Cai, S. Wang, and Q. Jiang, “Address extraction: Extraction of location-based information from the web,” *Web Technol. Res. Dev.*, pp. 925–937, 2005.

[32] M. J. Silva, B. Martins, M. Chaves, A. P. Afonso, and N. Cardoso, “Adding geographic scopes to web resources,” *Comput. Environ. Urban Syst.*, vol. 30, pp. 378–399, 2006.

[33] H. C. Lee, H. Liu, and R. J. Miller, “Geographically-Sensitive Link Analysis,” in *IEEE/WIC/ACM International Conference on Web Intelligence (WI’07)*, 2007, pp. 628–634.

[34] D. Ahlers and S. Boll, “Urban web crawling,” in *Proceedings of the first international workshop on Location and the web - LOCWEB ’08*, 2008, pp. 25–32.

[35] B. Hess, F. Magagna, and J. Sutanto, “Toward location-aware Web: extraction method, applications and evaluation,” *Pers. Ubiquitous Comput.*, vol. 18, no. 5, pp. 1047–1060, Oct. 2013.

[36] A. V. Aho and M. J. Corasick, “Efficient string matching: an aid to bibliographic search,” *Communications of the ACM*, vol. 18. pp. 333–340, 1975.

[37] R. S. Purves, P. Clough, C. B. Jones, A. Arampatzis, B. Bucher, D. Finch, G. Fu, H. Joho, A. K. Syed, S. Vaid, and B. Yang, “The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the Internet,” *Int. J. Geogr. Inf. Sci.*, vol. 21, no. 7, pp. 717–745, Aug. 2007.

[38] W. Li, M. F. Goodchild, R. L. Church, and B. Zhou, “Geospatial Data Mining on the Web: Discovering Locations of Emergency Service Facilities,” in *Advanced Data Mining and Applications*, Springer, 2012, pp. 552–563.

[39] W. Dou and J. Hu, “Automated Web Data Mining Using Semantic Analysis,” in *Advanced Data Mining and Applications*, Springer, 2012, pp. 539–551.

[40] W. Liu, X. Meng, and W. Meng, “ViDE: A Vision-Based Approach for Deep Web Data Extraction,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 3, pp. 447–460, Mar. 2010.

[41] D. Cai, S. Yu, J.-R. R. Wen, and W.-Y. Y. Ma, “VIPS: A vision-based page segmentation algorithm,” 2003.

[42] A. Tabarcea, P. Fränti, and V. Manta, “Using a Spatial Database in a Location-Based Search Application,” *Bul. Institutului Politeh. Iasi*, vol. LV (LIX), no. 3, pp. 55–62, 2009.

[43] S. Abiteboul, “Querying semi-structured data,” in *Database Theory—ICDT’97*, vol. 1186, F. Afrati and P. Kolaitis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 1–18.

[44] N. Gali, A. Tabarcea, and P. Franti, “Rule-based technique for extracting representative title for services detected in webpage,” *manuscript*, 2015.

[45] M. R. Luo, G. Cui, and B. Rigg, “The development of the CIE 2000 colour-difference formula: CIEDE2000,” *Color Res. Appl.*, vol. 26, pp. 340–350, 2001.

[46] N. Gali, A. Tabarcea, and P. Fränti, “Extracting Representative Image From Web Page,” in *WEBIST 2015*, 2015.

Paper P5

N. Gali, A. Tabarcea, P. Fränti, "Extracting representative image from web page". Int. Conf. on Web Information Systems and Technologies (WEBIST'15), Lisbon, Portugal, May 2015

Extracting Representative Image from Web Page

Najlah Gali, Andrei Tabarcea and Pasi Fränti
Speech and Image Processing Unit, School of Computing, University of Eastern Finland, Joensuu, Finland
{najlaa, tabarcea, franti}@cs.uef.fi

Keywords: Representative Image, Image Extraction, Web Page Information Extraction, Web Mining.

Abstract: A web page typically contains a blend of information. For a particular user, only informative data such as main content and representative images are considered useful, while non-informative data such as advertisements and navigational banners are not. In this work, we focus on selecting a representative image that would best represent the content of a web page. Existing techniques rely on prior knowledge of website specific templates and on text body. We extract all images, analyze and rank them according to their features and functionality in the web page. We select the highest scored image as the representative image. Our method is fully automated, template independent, and not limited to a certain type of web pages.

1 INTRODUCTION

The web today is a world of information, filled with videos, images and interactive content. To deal with these forms of data, different techniques have been developed to deliver the informative parts of a web page to the user (see Figure 1), such as information retrieval (Yu et al., 2003), main content extraction (Kim et al., 2013) and image extraction (Kherfi et al., 2004).

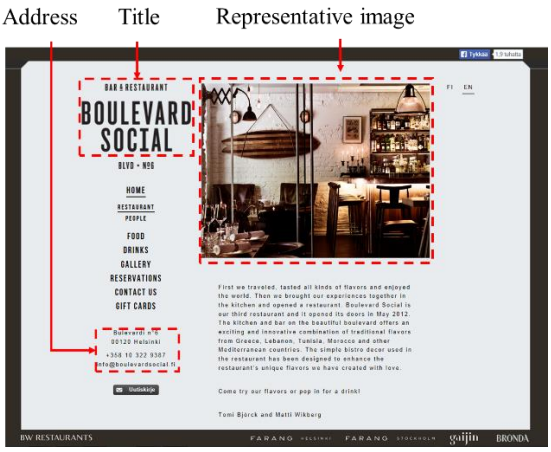


Figure 1: A sample web page and its relevant content to the user: title, image and location.

Images are used in web pages because they can transfer information to the user in a quick and efficient way, they are more informative than text at a glance. Even though a large number of images are embedded into web pages, many of them are less relevant to the content of the web page, such as advertisements, navigational banners, icons and images that serve as section headings (Azad et al., 2014). A solution is needed to ignore the irrelevant images and find a representative image for the web page.

We define the *representative image* of a web page as the image that best represents the content of the page to the user. Representative images are important in many applications, especially in cases when bandwidth limitation restricts the total number of images that can be retrieved or when building a visual category in which a single image must represent an entire category of documents and their associated content (Helfman and Hollan, 2000). Representative image is also important for location based applications such as MOPSI, which is available at cs.uef.fi/mopsi/, where simple thumbnail with title is the minimum information a user needs. It's also used in social applications such as Facebook and Google+ when users share a link of a web page on their wall.

Existing works have been mostly focused on extracting several useful images from a web page (Fauzi et al. 2009) or a collection of web pages (Park et al. 2006) and on selecting an image for a particular category of web pages such as news article. Less attention has been paid to how to select an image that represents the entire web page.

The method in (Joshi and Liu, 2009) focuses on news articles. It assumes that the relevant image is

embedded in the article block and has a caption and that non-article images have no caption. It considers only images with captions as candidate images and may therefore misses potentially useful web images that do not have captions.

The image extraction method in (Adam et al., 2010) focuses on web pages that are written in article-style (title and body). The method locates the border of the article and selects an image from this region based on its size and aspect ratio. It provides image annotation by identifying the captions assigned to them. This method considers only images that accompany the article, which is a section of the web page and may falsely select advertisement images if they have acceptable size and aspect ratio. Our task is wider because we consider all images in the web page and we select an image that represents the entire web page.

Google+ *share preview snippet* (Google+, 2014) summarizes a post made to Google+. It includes a link, a page title, a brief description of the page, and a thumbnail image. The image is selected based on its size and aspect ratio. The image height must be at least 120 pixels, and if the width of the image is less than 100 pixels, then the aspect ratio must be ≤ 3. Although the explicit framework for the snippet has not been published in any scientific forum, the method is described in technical document, and it is used in real application.

The method in (Tsymbalenko and Munson, 2001) focuses on finding relevant images to specific query without downloading or analyzing images. It examines only the text that surrounds the image tag in the source code of the web page and then decides whether the image is relevant or not. However, many web pages do not have text surrounding their useful images which lead to exclude them from being candidate images.

A functional categorization of images is studied in (Hu and Bagga, 2003). The images are classified into categories based on their usage in the web page by defining eight categories: *story*, *preview*, *commercial*, *host*, *heading*, *icons and logos*, *formatting*, *miscellaneous*. These are further grouped in two super-classes: one for useful images (story, preview and host) and one for the images that are not associated with the content (the other categories). We also use image categorization, but we use it directly in the method for helping to choose the best image.

The method in (Gupta et al., 2003) navigates Document Object Model (DOM) tree that is created by parsing the Hypertext Mark Up Language (HTML) code recursively and uses it to extract relevant information, including images. It filters out irrelevant data such as advertisement images by examining the values of the *src* and *href* attributes to determine the servers which the links refer to. If an address matches against a list of common advertisement servers, the node of the DOM tree that contains the link is deleted. We also use the DOM tree of a web page, but we use more image attributes and we define more categories.

The method in (Parmar and Gadge, 2011) removes advertisement images by using a rule-based classifier. Seven rules are defined to decide whether the image is an advertisement or not: *domain name difference*, *dimension*, *well-known advertisement provider*, *advertisement related keywords*, *advertisement by scripting*, *dynamic advertisement*, *flash plug-in removal*. This method eliminates most advertisement images in the web page.

Despite of several researchers have been working in related areas, none of the existing methods is directly applicable to our problem as such. To our knowledge, the only existing methods are the commercial ones implemented in Google+ and Facebook but, according to our experiments, neither of them is working perfectly.

In this paper, we propose a method that parses the source code of a web page, detects all the images and selects one that best represents the content. Instead of analyzing the content of the images or examining the text surrounding them, we rely on the functional purpose of the images within the web page and on the features such as the size, the aspect ratio, the format of the image and the attributes of HTML tags. Similarly to (Hu and Bagga, 2003) we classify the images into categories. We define the following categories based on image functionality: *representative*, *logos*, *banners*, *advertisements*, and *formatting including icons*. We rank the categories in this order, based on how important they are with respect to the content of the web page. The images in each category are ranked based on their features.

The main contribution of our method is that it does not rely on the surrounding text, on certain template or web page categories. Instead, it is targeted to work with all types of web pages. It is therefore general and not limited by the writing style or the layout of the web page. Besides the selection of the threshold values, the method does not require any training data. It is designed to work in real time, without the need to store the results in a database or to query a set of pre downloaded web pages. Since we consider prior classification of images, our method is useful in several applications such as automatic identifying adverts, saving bandwidth by web crawlers by downloading carefully only most relevant

media objects, and automatic converting web page for consumption on mobile small screen devices.

The proposed method is implemented in two places in a mobile location-based application called *Mopsi* (Fränti et al., 2011). The first one is to show search results to the mobile user, and the second one is the interactive tool for adding new services to the database using data from web pages.

2 EXTRACTING REPRESENTATIVE IMAGES

The workflow of the algorithm is shown in Figure 2. We start by downloading the source code of the web page and analyzing it using a DOM parser. The DOM representation is a platform- and language-independent interface that allows programs and scripts to dynamically access and update the content, structure and style of documents (www.w3.org/DOM).

We navigate through the DOM tree of the web page to identify links to images by locating <*img*> tags and to Cascading Style Sheets (CSS) files by locating <*link*> tags (*type*=text/css or *rel*=stylesheet), and JavaScript (JS) files by locating <*script*> tags. After analyzing the HTML source code, we use regular expressions to extract the images in CSS and JS files. If the considered web page does not contain any images and is not the root page of the domain, we also analyze the root page in the same way.

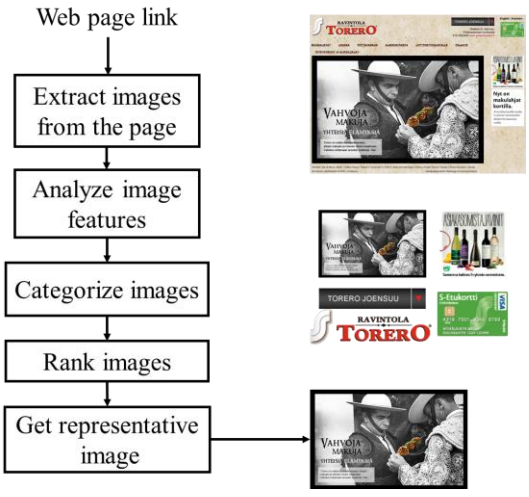


Figure 2: Extracting the representative image.

We notice that most of the images found in CSS are formatting or background images, although sometimes they have good features. Therefore, we chose to use images from CSS only in the case where no images are detected from HTML, even though best image is sometimes found from CSS (see Figure 3).

We then extract a list of features for each image, which are *src*, *alt*, *title*, *from*, *format*, *width*, *height*, *size*, and *aspect ratio* (see Figure 4).

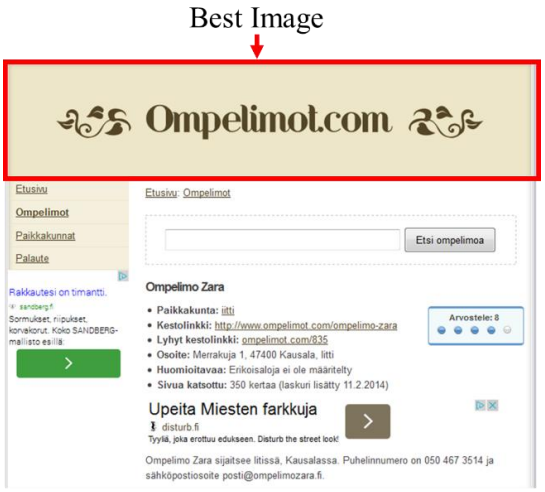


Figure 3: Best image is found from CSS source.

src	http://www.martina.fi/sites/martina.fi/files/styles/fiiliskuva/public/Valitse%20alikansio/Ravintolat/ravintola-martina-paakuva-pasta.jpg?itok=z8DMqAu2
alt	Ravintola Martina Joensuu
title	--
from	html
format	jpg
width	920
height	313
size	287.96 px
Aspect ratio	2.94
Parent tag	
Class	header_fiilis
Class of parent	content clearfix

Figure 4: A sample banner image and its features.

Because we aim at a real-time application, we do not download the images but we calculate width and height either by using the attributes of the <*img*> tag, retrieving them from CSS or by downloading just the image header (the first kilobytes of the file, which contain the image meta-information).

2.1 Categorization

We define five image categories based on its usage within the web page (see Figure 5) and rank them in the following priority order:

- *Representative:* images that are directly related to the content or the topic of the web page (see Figure 6);
- *Logos:* recognizable images that are used to identify the company or institution that owns the website (see Figure 7);
- *Banners:* images placed on a web page above, below or on the sides of the content. They are generally used for decoration. Headers and footers are classified in this category (see Figure 8);
- *Advertisements:* images that promote products or services that are irrelevant to the topic or the content of the web page (see Figure 9);
- *Formatting and Icons:* images that are used to enhance the visual appearance such as spacers, bullets, borders, backgrounds, or pictures used purely for decoration. We also include the small images which are not classified as logos and serve a functional purpose, such as icons which link to the home page or icons which are used for changing language (see Figure 10).

All images are first assigned into a proper category, and the images in the same category are ranked according to a secondary criteria. The image is chosen from the highest priority category that contains any image.

Figure 5: A sample web page which contains images from all the 5 categories we defined.

Figure 6: Examples of representative images.

Figure 7: Examples of logos.

Figure 8: Example of banners.

Figure 9: Examples of advertisement images.

Figure 10: Examples of formatting images and icons.

We categorize the images using the rules in Table 1. In all categories, a predefined set of keywords is used. If any of these are found in the image URL, in the class name of the <*img*> tag, or of the parent element, then the image is assigned to that category. Banners and Formatting are also categorized according to image size and aspect ratio.

Table 1: Rules used for image categorization.

Category	**Features**	**Keywords**
Representative	Not in other category	
Logo		logo
Banner	Ratio>1.8	banner, header, footer, button
Advertisement		free, adserver, now, buy, join, click, affiliate, adv, hits, counter
Formatting and Icons	Width<100 px Height<100 px	background, bg, spirit, templates

Note that the categories are overlapping, so the same image may meet the conditions of multiple categories. In this case, we use a decision tree to assign the image to a single category (see Figure 11). We categorize logo images first because their size and aspect ratio might satisfy the conditions of Banner and Formatting categorizes. We categorize advertisement images next because their aspect ratio or their HTML assigned keywords might satisfy the conditions of Formatting or Banner categories. Formatting category is followed because its image aspect ratio might satisfy the condition of Banner category. An image can belong to the class of Representatives only if it does not belong to any other category. The same prioritization is applied for all HTML, CSS and JS, and the images in these file types are considered equal.

The criterion for Logos category is that at least one of the HTML tag attributes (URL, the detected classes, the IDs of the element, or the IDs of the parent element) contains the keyword ”*logo*”. The criterion for Banners category is that at least one of the HTML tag attributes contains any of the keywords: ”*banner*”, ”*header*”, ”*footer*”, ”*button*”, or that the aspect ratio of the image is higher than a threshold 1.8, which was experimentally obtained using small training set of 50 web pages.

The criterion for Advertisements category is that at least one of the HTML tag attributes contains any of the advertising keywords: ”*free*”, ”*now*”, ”*buy*”, ”*join*”, ”*adserver*”, ”*click*”, ”*affiliate*”, ”*adv*”, ”*hits*”, ”*counter*”.

Advertisement images can be hosted either on the same domain as the web page or on a separate server. It is also common that useful images are stored on a different domain, as more and more websites are using a separate server to store images on a cloud storage server. Therefore, the domain where the image is hosted is not a consistent rule that could be used to determine if the image is an advertisement.

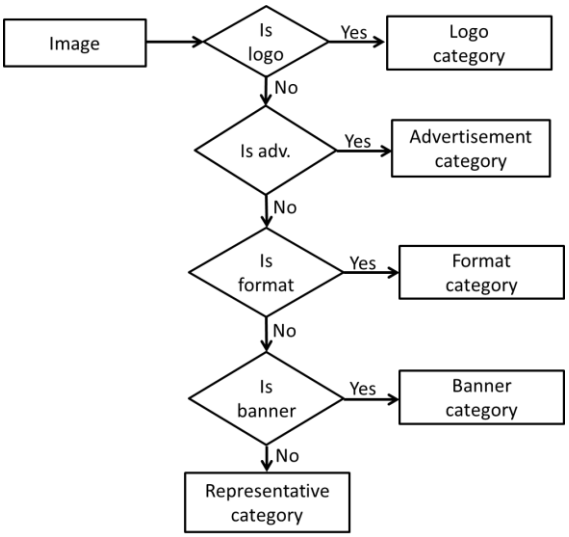


Figure 11: Decision tree used in assigning image categories.

The criterion for Formatting and icons category is that at least one of the HTML tag attributes contains any of the keywords: “*background*”, “*bg*”, “*sprite*”, “*templates*” or if the height or width are smaller than an experimentally selected threshold of 100 pixels.

2.2 Scoring

We analyze the features of the image according to a set of rules as shown in Table 2. We score the image based on the following criteria:

- *Image size*: we consider the image has a good size if it meets the following condition:

$$Size = width \times height \geq 10.000\,px \qquad (1)$$

- *Aspect ratio*: we consider the image that has aspect ratio ≤1.8 is more important than other images, which tend to be either wide and short, or narrow and long, which are usually features of banners, formatting or advertisements. We calculate the aspect ratio as follows:

$$Ratio = \frac{\max(width, height)}{\min(width, height)} \leq 1.8 \qquad (2)$$

- *Image alt and title*: the alt and title attributes describe the content of the image. Images that have alt or title attributes are more important than other images in the web page. Therefore, we extract the keywords of image alt and title and compare them with the keywords of the web page <*title*> and <*h1*> contents. We firstly extract the content of the web page <*title*> and <*h1*> by xpath. Secondly, we use a predefined set of patterns which consists of space and delimiters such as ‘,’, ‘;’, ‘/’, ‘|’, ‘>’, ‘¦’, ‘«’, ‘–’, ‘.’, ‘:’, ‘?’, ‘::’ to separate the words and the phrases of the web page <*title*> and <*h1*> tags. Thirdly, we remove any special characters such as ‘[‘, ‘{‘, ‘?’, ‘!’ from the text. Finally, we use string comparison to match the keywords of image *alt* and *title* with the keywords of the web page *title* and *h1s*;
- *Image path and URL*: we parse the image path, extract its keywords, and match the keywords with the web page <*title*> and <*h1*> keywords. If a match is found then we consider the image is more related to the content of the web page;
- *In the sub-tree of <h1> or <h2> tags*: we consider an image that is a child of <*h1*> or <*h2*> in the DOM tree is more related to the content of the web page because <*h1*> and <*h2*> contain the main topics of the page, therefore, we consider the images located in these sub-trees are important;
- *Image format*: we analyze four types of formats, which are Joint Photographic Experts Group (jpg), Scalable Vector Graphics (svg), Portable Network Graphics (png), and Graphics Interchange Format (gif). We consider jpg format is more important because it is used for photographs. Although png format is suitable and is increasingly used for compressing photos, most of the web pages use it just for logos and icons. Therefore, we consider it less important than jpg. Less importance is also given to image of format svg and gif because these types of formats are used for graphics and they are more often used for formatting images.

All rules are considered equally important and therefore assigned the same weight of 1 except for some types of image formats as mentioned earlier.

The scores are calculated only for images in the highest priority category, in our case Representative category. If no image is assigned to this category then the scores will be calculated for the images of next highest priority category and so on.

The scores are summed up and the images in the category are ranked based on their scores, except for logo category where the images are sorted based on size, because we consider the web page logo has biggest size among other logos that exist in the page.

Table 2: Rules used for image scoring.

Rule	**Score**
Image size ≥ 10.000 px	1
Aspect ratio ≤ 1.8	1
Image alt or image title has a value	1
Keywords of alt or title are in web page <*title*>	1
Keywords of alt or title are in web page <*h1*>	1
Keywords of path are in web page <*title*> or <*h1*>	1
The image is in the sub-tree of <*h1*> or <*h2*> tags	1
Format : jpg	1
Format: svg	0.5
Format: png	0.5
Format: gif	0.5

3 EXPERIMENTS

To collect a reasonable size of ground truth database we asked volunteers to select at most three images from websites of their own choice, or Mopsi search result. The interface of the data collection can be found here: cs.uef.fi/mopsi/img/. It works as follows:

Search: The user can copy/paste the link of any website he/she visits often, like, or at least knows about. Website selection is therefore not limited to a country, category, specific domain or size of website. Alternatively, the user should give a keyword such as a favorite hobby and a city in Finland. Mopsi search will then provide resulting pages for him/her to evaluate. If the search results returned are service directory, the content is unclear, or the user just cannot decide, then he/she can skip the page and try another keyword/city combination. The user should select maximum three images that best describe the web page.

Evaluation: We have collected a dataset of 1002 websites and 2363 ground truth images (2.36 images per webpages, on average) by 117 volunteers during September 2014. The number of images in each website varies between 1 to over 154. Although the selected images serve as useful ground truth for our purpose, users’ choices can sometime be subjective,

which makes it impossible for any realistic algorithm to get 100% accuracy with this data, even if the algorithm was trained for this particular website and knew the user who made the selection. Nevertheless, the ground truth collection is still useful for evaluating the performance of our method, on average.

We compare the performance of our method (WebIma) with Google+ algorithm because it also looks for an image to represent the entire webpage and it uses the same heuristics as in (Adam et. al., 2010), which are the size and the aspect ratio of the image. In addition, we also compared our method with the method used in Facebook when user shares web link on his/her wall. We evaluated these three algorithms by counting how many times they select one of the ground truth images as the output. The results were obtained by input the web link to the algorithms (ours by script, the others manually one by one) and comparing their first choice against the ground truth. These were done using their public web pages during 17-25 September 2014. The results are summarized in Table 3. It shows that our method finds correct image 642 times out of 1002 (64%), which outperforms the comparative results of Google+ (48%) and Facebook (39%). The results also show that our parser extracted the images from 99% of the tested websites.

To find out why the methods perform differently, we have selected two sets of samples from the collected dataset. The first set contains 30 websites where our algorithm performed 100% accuracy while both Google+ and Facebook failed. The second set contains 30 websites where our algorithm failed.

According to our categorization, for the first set the analysis shows that 63% of the ground truth images are selected from Representative category and the rest 37% of the images are from logos category. Less importance was given to the images of banner category, and no images were selected from advertisements and Formatting categories.

Comparing these results with the selections made by Google+ and Facebook, we can observe that both Google+ and Facebook preferred banner images because of their big size. About (57-60) % of images selected by them belong to banner category, which reduces the performance of both algorithms (see Table 4).

Further analysis of the images features shows that the users did not rely mainly on the images that are big in size, long or wide. About 73% of the ground truth images in set 1 are relatively smaller in size, height and width in comparison with those selected by Google+ and Facebook.

Table 3: Performance results for Mopsi WebIma dataset.

	Accuracy	**Extracted Images**
WebIma	64%	99%
Google+	48%	92%
Facebook	39%	90%

The aspect ratio of the images recorded in the first set lies between 1 and 6.2 and about 50% of images selected by our algorithm have aspect ratio lower than the ratio of the images selected by Google+ and Facebook. This indicates that users prefer more square images than images like banner. The statistics also show that most of the ground truth images in this set are of jpg format.

In the second set, as shown in Table 4, the ground truth images are distributed among three categories, which are Representative, Logos and Banners. Our algorithm did not select the correct images because the websites in this set contain many images that meet the criteria of Representative category, but are not selected by the users. The statistics shows that logo images are important and should be given bigger weight in the scoring.

The analysis of the images features in set 2 shows that both Google+ and Facebook selected more small images compared to those selected by WebIma.
This means that the users do not necessarily prefer biggest images in the web and therefore, bigger size should not be considered as the only threshold to select the representative image. Both Google+ and Facebook ignore the logo images if they do not meet the thresholds of the size and the aspect ratio, which affects their accuracy.

Jpg is still the most popular format for the representative images and the statics of the whole data set of 2363 images shows that 63% of the images are of jpg format. Png and gif are preferred for the logo and formatting images. These results make our early assumption of giving extra score to jpg images, is correct. Added to that, both Google+ and Facebook do not select CSS images even though some websites use only CSS and JS images in their design (see Figure 12).

We conclude that some of image features such as the aspect ratio and the jpg format are more important than other features such as the size of the image and therefore should have been given extra weights. Better optimization of the weights, however, is left for future work. Image categorization is also important because it identifies advertisements and formatting images as being harmful and excluded them from being candidates for image representation.

Table 4: Number of images selected from each category.

	Set 1				Set 2			
Image Category	Ground truth (%)	WebIma (%)	Google+ (%)	Facebook (%)	Ground truth (%)	WebIma (%)	Google+ (%)	Facebook (%)
Representative	63	63	13	20	33	83	27	40
Logo	37	37	13	10	30	7	30	7
Banner	0	0	57	60	37	3	27	33
Advertisement	0	0	0	0	0	0	3	3
Formatting	0	0	17	10	0	7	13	17

Figure 12: A website uses images from CSS source only.

No significant differences were concluded with respect to the other features from the selected sets.

4 CONCLUSIONS

We have introduced a method for extracting representative image from a web page. With the collected dataset, it finds correct image in 64% of the cases, which outperforms the results provided by Google+ and Facebook. Our method is implemented in the framework of MOPSI, which is a research project of location-based services developed at the University of Eastern Finland in two places: Search and Service upgrade.

The method works well especially for business oriented touristic places that have their own web page, whereas smaller enterprises in small towns or rural areas rely more on service directories. The service directories include two challenges: they include content of multiple services and it would be more difficult to detect an image that relates to the service in question. Many commercial service directories have also quite poor content, usually showing only name, contact info and map, followed by the service provider's own information. Image of the services themselves are often missing completely. Some web pages also have rather complicated structure where the visual appearance is not a single image, but more complicated product of several independent components. Future research should focus on these challenges.

Some improvement can also be done to the current method such as training the parameters for better results. This would require a large set of training data, which we are currently lacking. Nevertheless, the data we used has 1002 web sites, which makes the results statistically significant.

ACKNOWLEDGEMENTS

The work described in this paper was supported by MOPIS project, University of Eastern Finland.

REFERENCES

Adam, G., Bouras, C., & Poulopoulos, V., 2010. Image Extraction from Online Text Streams: A Straightforward Template Independent Approach without Training. In *Advanced Information Networking and Applications Workshops (WAINA), 24th International Conference*, pp. 609-614. IEEE.

Azad, H. K., Raj, R., Kumar, R., Ranjan, H., Abhishek, K., & Singh, M. P. 2014. Removal of Noisy Information in Web Pages. *In Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies.* ACM.

Fauzi, F., Hong, J. L., & Belkhatir, M. 2009. Webpage segmentation for extracting images and their surrounding contextual information. *In Proceedings of the 17th ACM international conference on Multimedia*, pp. 649-652. ACM.

Fränti, P., Chen, J., & Tabarcea, A. 2011. Four Aspects of Relevance in Sharing Location-based Media: Content, Time, Location and Network. In *WEBIST*, pp. 413-417.

Google+ platform, 2014, *https:/developers.google.com/+/web/snippet/?hl=no.*

Gupta, S., Kaiser, G., Neistadt, D., & Grimm, P., 2003. DOM-based content extraction of HTML documents. In *Proceedings of the 12th international conference on World Wide Web*, pp. 207-214. ACM.

Helfman, J. I., & Hollan, J. D., 2000. Image representations for accessing and organizing Web information. In *Photonics West 2001-Electronic Imaging*, pp. 91-101. International Society for Optics and Photonics.

Hu, J., & Bagga, A., 2003. Functionality-Based Web Image Categorization. *WWW (Posters)*, *2*(003).

Joshi, P. M., & Liu, S., 2009. Web document text and images extraction using DOM analysis and natural language processing. In *Proceedings of the 9th ACM symposium on Document engineering*, pp. 218-221. ACM.

Kherfi, M. L., Ziou, D., & Bernardi, A., 2004. Image retrieval from the world wide web: Issues, techniques, and systems. *ACM Computing Surveys (CSUR)*, *36*(1), pp. 35-67.

Kim, M., Kim, Y., Song, W., & Khil, A., 2013. Main Content Extraction from Web Documents Using Text.

Block Context. In *Database and Expert Systems Applications*, pp. 81-93. Springer Berlin Heidelberg.

Park, G., Baek, Y., & Lee, H. K. 2006. Web image retrieval using majority-based ranking approach. *Multimedia Tools and Applications*, *31(2)*, pp.195-219.

Parmar, H. R., & Gadge, J., 2011. Removal of Image Advertisement from Web Page. *International Journal of Computer Applications*, 27(7).

Tsymbalenko, Y., & Munson, E. V., 2001. Using HTML metadata to find relevant images on the world wide web. *Proceedings of internet computing*, *2*, pp.842-848.

Yu, S., Cai, D., Wen, J. R., & Ma, W. Y., 2003. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th international conference on World Wide Web*, pp. 11-18. ACM.

Paper P6

A. Tabarcea, Z. Wan, K. Waga and P. Fränti, "O-Mopsi: Mobile Orienteering Game Using Geotagged Photos", Int. Conf. on Web Information Systems and Technologies (WEBIST'13), Aachen, Germany, 8–10 May 2013

O-Mopsi: Mobile Orienteering Game using Geotagged Photos

Andrei Tabarcea, Zhentian Wan, Karol Waga and Pasi Fränti
Speech and Image Processing Unit, School of Computing, University of Eastern Finland, Joensuu, Finland
{tabarcea, zhentiw, kwaga, franti}@cs.uef.fi

Keywords: Location-aware Gaming, Mobile Gaming, GPS, Orienteering.

Abstract: Location-based mobile gaming combines gameplay with physical activity. We have developed a game, O-Mopsi, based on the concept of orienteering, which can be played on mobile phones with GPS receiver and Internet connection. In order to complete a game, a player must visit a set of targets that are photos chosen from a user-generated geotagged database. Game creation, management and live tracking can be done using a web interface. The game was presented at an annual international festival which is aimed at introducing science and technology to school children and the overall feedback received from the players was positive.

1 INTRODUCTION

Whilst using mobile devices for playing games is very popular and long-established idea, mobile gaming which uses the player's location and involves physical activity is a recent idea, although it starts to gain popularity. With the fast development computing devices, mobile location-based gaming has evolved from using wearable computers and head-mounted displays (Piekarski and Thomas, 2002) to using mobile phones with access to Internet and GPS localization.

We introduce a mobile location-aware game, O-Mopsi, which is based on the classical concept of orienteering and exploits the multimedia data available in a geotagged user generated photo collection (cs.uef.fi/mopsi/photos). A game is created by defining a set of targets and a player can complete the game by visiting all the targets in any order. A target is represented by a photo along with location information.

O-Mopsi can be played using a mobile application for Symbian phones, which is available at cs.uef.fi/o-mopsi/ along with the web interface.

The mobile client has functionalities such as plotting targets on the map, displaying compass data and giving audio clue with varying pitch and sound frequency about the distance to a target. Photos are used as an additional aid for identifying a target

The web interface allows game management, real-time player tracking, post-game trail analysis and suggesting tours calculated by either greedy heuristics or an ant colony-based optimization.

2 RELATED WORK

One of the first examples of location-aware games is *Pirates!* (Björk et al., 2001), which demonstrates how proximity-triggered technology can be integrated into computer game design.

A large number of location-aware games are adaptations of traditional board games or computer games. including *Quake* (Piekarski and Thomas, 2002), *Pacman* (Cheok et al., 2004), *Tic-Tac-Toe* (Schlieder et al., 2006), *Monopoly* (Li et al., 2008) or *Chase and Catch* (Misund et al., 2009).

Mobile gaming is often combined with education. For example, *Savannah* (Facer et al., 2004) aimed at children, who use devices with GPS to navigate and study animal behavior in a virtual savannah. Education is also the goal of *Skattjakt* (*Treasure Hunt*), (Spikol and Milrad, 2008), which encourages players to get physically active by solving a mystery surrounding a castle. It considers that mobile outdoor games are well suited for novel learning activities that involve physical motion, problem solving, inquiry and collaboration.

A common approach is combining game reality with physical reality. For example, *Song of the North* (Lankoski et al., 2004) is game inspired by Nordic mythology, which combines a virtual spirit world with the physical world and in which the players interact by using mobile devices. Another example of mixed reality is *Can you see me now?* (Benford et al., 2006), where players are chased through a virtual model of a city by runners (professional performers equipped with Wi-Fi and

GPS). The rules are similar with the game of catch, but the "runners" have to traverse actual city streets in order to capture the online (virtual) players.

In a location-aware game, photos can be used in various ways, such as a means of interaction (Suomela and Koivisto, 2006) or to provide additional information about game targets, such as in *See It* (Neustaedter and Judge, 2012).

O-Mopsi is based on the concept of virtual orienteering, similarly to applications such as OrientGames (www.orientgames.com) or Virtual Orienteering Game (www.vorienteering.com). The difference between O-Mopsi and most of the other orienteering-based games are that the targets are shown as photos and the order of visiting the targets is freely chosen.

3 GAME RULES

The goal of the game is to visit all targets in the shortest time. A target is identified by its location, photo and a short description. A game is created by selecting photos from the Mopsi photo collection.

A game starts when the player visits the first target. The order of targets is not fixed and it can be freely chosen by the player. A game ends when the player visits all the targets. To visit a target, a player has to be closer than 20 meters from its location. This threshold was chosen taking into consideration GPS inaccuracies. Players are ranked in the order of completion time. The total distance, the starting target or the order of visiting targets do not affect ranking.

The game shares similarity with the concepts of orienteering and geocaching (Cameron and Ulmer, 2004), which both require identifying and visiting a number of targets.

Orienteering requires navigating from point to point in a predefined order using a map and a compass. Unlike orienteering, in O-Mopsi the targets can be visited in any order, encouraging players to develop different strategies and to choose the starting point. Orienteering is focused on identifying the location on map whereas O-Mopsi provides additional hint in the form of photo of the target.

Geocaching requires finding a hidden treasure (a collection of things placed in a container called geocache and placed in a location available to the public). The GPS location of the cache is published online and other people need to find the cache and replace an item from the collection by another. O-Mopsi also requires find a certain GPS location defined by another player, but the location is identified by a photo. Additionally, in Geocaching, the time does not matter and targets are not grouped into games or other entities.

O-Mopsi uses virtual targets, whereas orienteering and geocaching use physical objects.

4 O-Mopsi WEB

O-Mopsi web interface can be used for creating and managing a game, displaying the proposed shortest path of a game, displaying game results and viewing real-time player progress.

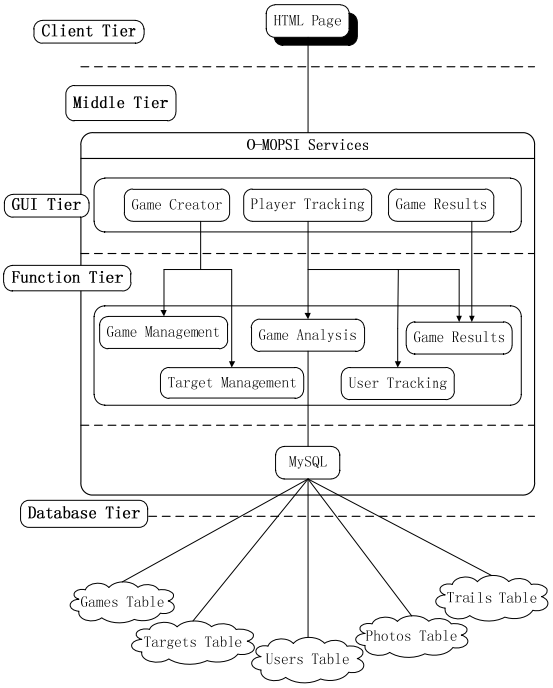


Figure 1: Architecture of the O-Mopsi website.

Fig. 1 shows the architecture of the website, along with the main functions. We use Google Maps for displaying targets, the interface is developed using PHP and JavaScript and the game data is stored in a MySQL spatial database.

The client tier contains the HTML page which is presented to the user. The middle tier is composed of the GUI tier and the function tier. The GUI tier includes interfaces for the creation of the game, for the live tracking of the players and for the game results. The function tier includes all the server logic operations: game and target management, game analysis, user tracking and game results. Finally, the database tier consists of the tables needed to store game data (games, targets, users, photos and trails).

4.1 Game Management

A game is created by providing a description for a set of targets. The targets are selected from the Mopsi photo collection with the help of keyword search or using the location-based recommendation documented in (Waga et al., 2012b).

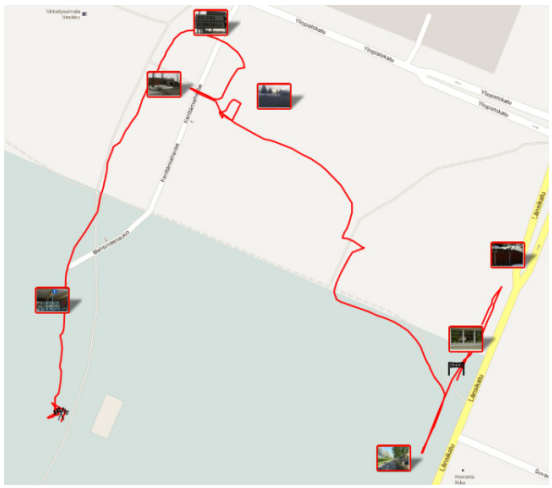


Figure 2: Live player tracking.

4.2 Game Tracking and Analysis

Player's location and the targets can be seen in real time on web in the Game Hall (see Fig. 2). This allows viewing the player's progress and analyzing the players' trails along with other characteristics such as speed or moving type using the algorithm in (Waga et al., 2012a). The user has also the option of viewing the shortest route calculated by ant colony optimization algorithm (see Fig. 3).

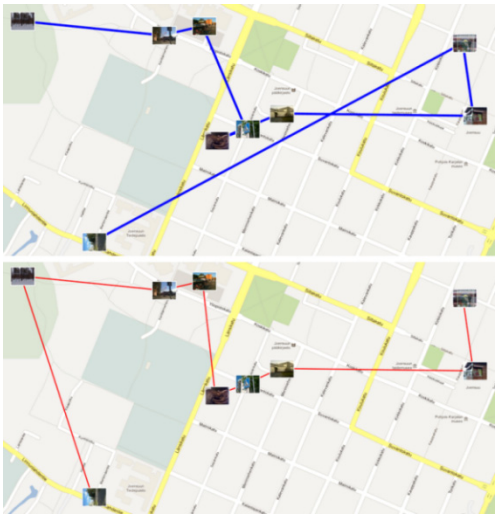


Figure 3: Ant colony optimized route.

5 GAME CLIENT

The game client is available for Symbian phones. It is developed using C++ and the Qt libraries. A Windows Phone version is in development and iOS and Android versions are considered in future. After logging in, the player can choose to join a new game or continue an existing active game.

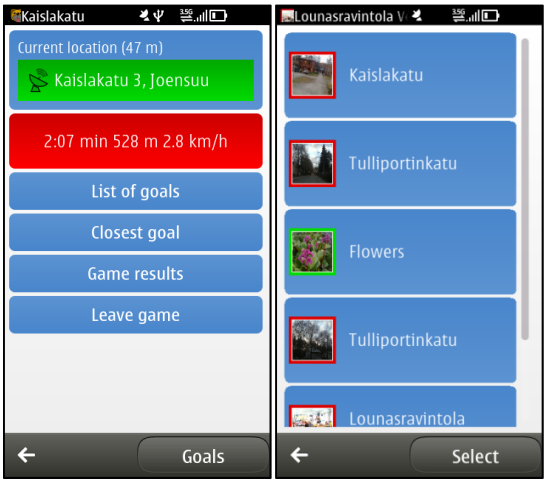


Figure 4: The main screen of the application (*left*) and a list of targets (*right*).

During gameplay, the player's location is tracked and stored on server. The main screen of the application (see Fig. 4) shows the current location, accuracy and statistics such as playing time, distance and speed. It also contains shortcuts to the map, option to highlight the closest target and to view the full list of targets.

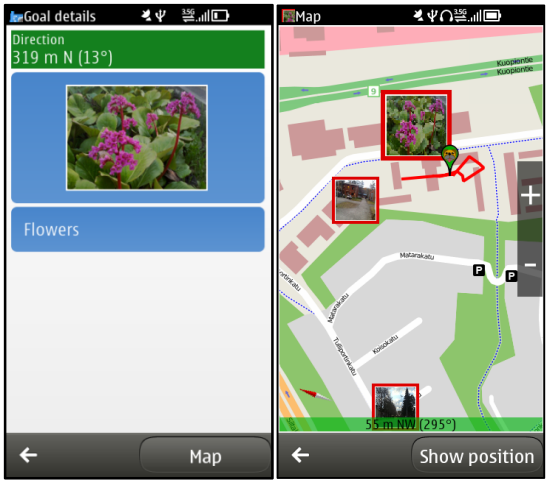


Figure 5. Viewing target details (left) and highlighting the chosen target on the map, along with the player's trail (right).

A target can be identified by its photo highlighted on the map (see Fig. 5). For aiding navigation, the application displays the distance and bearing to the selected target along with player's orientation taken from the phone's compass sensor.

Additionally, the player is guided by sounds while the map is open. When the player approaches a target and is closer than 500m, a beeping sound is played at fixed intervals. The interval between sounds is inversely proportional to the distance, starting from 5 seconds (500 m) and decreasing by 1 second every 100m. The sound frequency increases or decreases as the player becomes closer or further away from the target. Reaching the target turns off the sound guidance.

6 FEEDBACK

O-Mopsi was designed for the annual SciFest festival (www.scifest.fi), which brings together thousands of school kids, high school students, and teachers to learn about science and technology (Jormanainen and Korhonen, 2010). SciFest is organized in Joensuu, Finland in April.

O-Mopsi was presented during 2011 and 2012. Because limited availability of smart-phones, the players were organized into teams. After the game, a short feedback survey was filled by the teams.

Table 1: Players' ratings of the game.

Feedback	Very good	Good	Adequate	Bad
Scifest 2011	3	6	0	0
Scifest 2012	3	7	0	2

Feedback (see Table 1) shows that the game as mostly rated as good or very good. According to users, game rules are easy to understand and playing the game is enjoyable.

7 CONCLUSIONS

O-Mopsi is a location-aware game based on the concept of orienteering, in which targets can be visited in any chosen order. Testing O-Mopsi in a real-world situation during an international festival produced positive feedback.

REFERENCES

Benford S., Crabtree A., Flintham M., Drozd A., Anastasi R., Paxton M., Tandavanitj N., Adams R., Row-Farr J., 2006. Can You See Me Now? *ACM Trans. on Computer Human Interaction,* 13(1):100–133.

Björk S., Falk J., Hansson .R, Nakao K., Ljungstrand P., 2001. Pirates – using the physical world as a game board. *Interact 2001*, Tokyo, Japan, July 2001

Cameron, L., Ulmer, D., 2004. *The Geocaching Handbook*. Globe Pequot Press.

Cheok A., Goh K., Liu W., Farbiz F., Fong S., Teo S., Li Y., Yang X., 2004. Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. *Personal and Ubiquitous Computing*, 8(2):71-81.

Facer K., Joiner R., Stanton D., Reid J. Hull R., Kirk D., 2004. Savannah: mobile gaming and learning? *Journal of Computer Assisted Learning*, 20(6):399-409

Jormanainen I., Korhonen P., 2010. Science festivals on computer science recruitment. *Koli Calling Int. Conf. on Computing Education Research*

Lankoski P., Heliö S., Nummela J., Lahti J., Mäyrä F., Ermi L., 2004. A case study in pervasive game design: the songs of north. *NordiCHI '04*, pp. 413–416

Li M., O'Grady M. J., O. Gregory M.P., 2008. Geo-gaming: The mobile monopoly experience. *WEBIST 2008*, pp. 220-223.

Misund G., Holonen H., Karlsen J., Tolsby H., 2009. Chase and catch – simple as that? Old-fashioned fun of traditional playground games revitalized with location-aware mobile phones. *Int. Conf. on Advances in Computer Entertainment*.

Neustaedter C., Judge, T., 2012. See It: A Scalable Location-Based Game for Promoting Physical Activity. *CSCW'12*, Seattle, USA.

Piekarski W., Thomas B., 2002. ARQuake: The Outdoor Augmented Reality Gaming System. *Communications of the ACM*, 45(1), pp. 36-38.

Schlieder C., Kiefer P., Matyas S., 2006. Geogames – Designing Location-based Games from Classic Board Games. *IEEE Intelligent Systems*, 21(5):40-46.

Spikol D., Milrad M., 2008. Combining Physical Activities and Mobile Games to Promote Novel Learning Practices. In *WMUTE 2008*, pp. 31-38.

Suomela R., and Koivisto A., 2006. My photos are my bullets – using camera as the primary means of player-to-player interaction in a mobile multiplayer game. *ICEC 2006*.

Waga K., Tabarcea A., Chen M., Fränti P., 2012. Detecting Movement Type by Route Segmentation and Classification. *CollaborateCom '12*, Pittsburgh, USA, October 2012.

Waga K., Tabarcea A., Fränti P., 2012. Recommendation of Points of Interest from User Generated Data Collection. *CollaborateCom '12*, Pittsburgh, USA, October 2012.