Journal of Location Based Services

ISSN: 1748-9725 (Print) 1748-9733 (Online) Journal homepage: http://www.tandfonline.com/loi/tlbs20

Framework for location-aware search engine

Andrei Tabarcea, Najlah Gali & Pasi Fränti

To cite this article: Andrei Tabarcea, Najlah Gali & Pasi Fränti (2017) Framework for location-aware search engine, Journal of Location Based Services, 11:1, 50-74, DOI: 10.1080/17489725.2017.1407001

To link to this article: https://doi.org/10.1080/17489725.2017.1407001

Published online: 27 Nov 2017.

JOURNAL OF LOCATION BASED SERVICES, 2017
VOL. 11, NO. 1, 50–74
https://doi.org/10.1080/17489725.2017.1407001

Taylor & Francis
Taylor & Francis Group

Framework for location-aware search engine

Andrei Tabarcea, Najlah Gali and Pasi Fränti

Machine Learning Group, School of Computing, University of Eastern Finland, Joensuu, Finland

ABSTRACT

Nowadays, a large part of the multimedia data on the Internet is generated with devices that automatically annotate them with location information However, free-form content on 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 summarising the search results are titles and representative images, which we mine from the content using simple rule-based approaches utilising the structure of web pages. This information can be used to personalise search results for mobile users so that the results are relevant to their location.

ARTICLE HISTORY
Received 16 May 2017
Accepted 8 November 2017

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 (Patterson, Muntz, and Pancake 2003). A large part of these multimedia data is nowadays generated with devices that automatically annotate them with location information, but free-form content, such as that found on websites, does not implicitly contain any geographical information. Mobile search engines allow users to find information anytime and anywhere. However, search performance is influenced by the type of mobile device and the user location, profile, previous activity, time of year and social network (Liu, Rau, and Gao 2010).

Location is an important factor in personalising web search because the content of a website often has relevance only to a limited area (Bennett et al. 2011). *Location-based search* aims at finding a business or place around a specific geographical location. This is implemented by search engines that support geographical preferences (Markowetz et al. 2005). The relevance of a search result depends on the distance between the user-specified location and the location of the service (Yokoji, Takahashi, and Miura 2001). Location-based search changes the search from web-oriented to service-oriented, which makes it a more challenging task for 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

CONTACT Najlah Gali najlaa@cs.uef.fi

is relevant to. Second, we need to extract a summary of content from the web page, such as the title and images, but, in the case of service directories, we also need to detect the part belonging to that particular service. This requires both identification of geographical data and automatic information extraction from web pages.

Locations can be embedded explicitly into the HyperText Mark-up Language (HTML) as geographical coordinates (latitude and longitude) usually in three different ways: as meta tags named *geo-tags*,[1] as <*address*> tags. or as plain text. According to previous studies (Vänskä 2004; Ahlers and Boll 2008a) and our own experiments, very few web pages contain explicit locations. Instead, 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*. Particular cases of geo-referencing are *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.

A typical workflow of a location-based search application 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 results consisting of the following: short text summary (title), link, image thumbnails, address and distance.

In this paper, we present a location-based framework for a search engine, summarise our choices for its components, and discuss their advantages and disadvantages. We extract locations by analysing the text content of the web pages. To summarise the detected entities, we extract a title and a representative image for each search result. The framework integrates 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 should be close to the user's location, relate to the keywords provided by the user and extracted

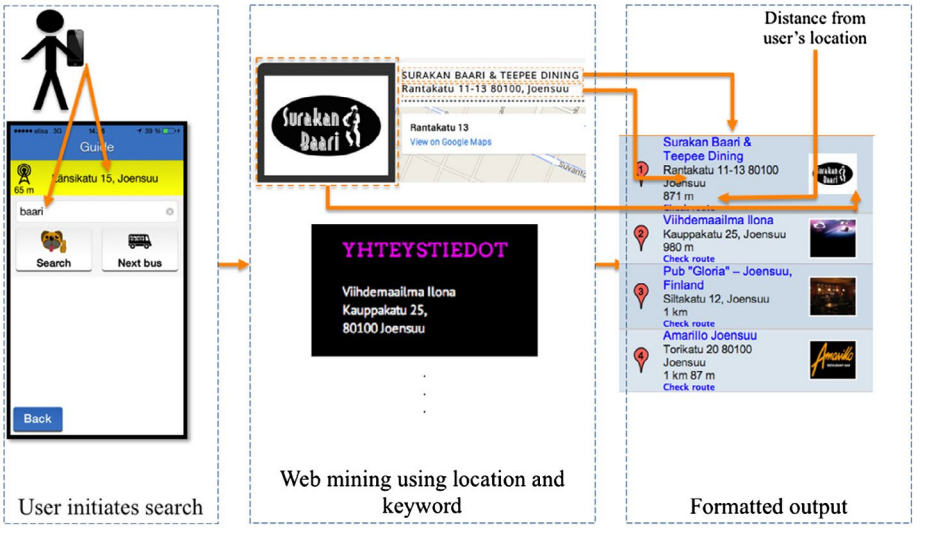

Figure 1. Web mining using location and keyword.

from the content of the websites found by standard search engines. The prototype of the proposed framework is implemented in the Mopsi platform, see (Fränti, Kuittinen et al. 2010; Fränti, Tabarcea et al. 2010). Besides location-based search, Mopsi offers tools for collecting, processing and displaying location-based data, such as photos and GPS trajectories, along with social media integration.

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 (Tsai 2011) that finds services 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 Wang et al. (2005), 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. Content 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 (Ding, Gravano, and Shivakumar 2000), 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. We focus 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 (Shi and Barker 2011). 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 locations from websites by several different strategies, as outlined in Hu, Lim, and Rizos (2006): 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. According to Mikheev, Moens, and Grover (1999), detecting addresses is more accurate when using a gazetteer.

Early methods of assigning locations to web resources in studies by (Buyukokkten et al. 1999) and (Watters and Amoudi 2003) relied on identifying the host location, which is the location of the owner or the administrator of the website. These researchers assigned a single location for each website by querying its *Whois*[2] records for the address and telephone number of the network administrator. This method was expanded on by (McCurley 2001)

who additionally used 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 (Goldberg, Wilson, and Knoblock 2007). They can be converted into locations by geo-coding services. We use two freely available geo-coding services: OpenStreetMap and a publicly available *geo-coded* database for Finland. Despite the occasional unavailability of the service and data inaccuracies, free geo-coding services such as OpenStreetMap or Geonames[3] are best suited for applications that require near-accurate (but not necessarily exact) geo-tagging because they have extensive coverage (Florczyk et al. 2010). 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*) (McCurley 2001). Both types of ambiguity can be resolved using heuristic rules. The algorithm in the work of (Zhang et al. 2011) attempts to resolve the geo/geo ambiguity using an algorithm similar to Google's Page Rank (Page et al. 1999). 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 to detecting addresses, such as training a classifier (Viola and Narasimhan 2005) or syntactic pattern recognition (Can et al. 2005), but most of the address detection methods, such as (Cai, Wang, and Jiang 2005; Silva et al. 2006; Borges et al. 2007; Lee, Liu, and Miller 2007; Ahlers and Boll 2008b; Leung, Lee, and Lee 2013) and ours roughly follow the same process. Firstly, address elements are identified using regular expressions and ontologies, which, in most cases, are gazetteers with hierarchical relationships between elements. Second, address candidates are built. 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 can be built either by training or using a set of rules, relationships and heuristics.

There have been several works in the literature on location-aware search. A personalised mobile search engine enhanced by capturing users' preferences in the form of click-through data is proposed in the work of (Leung, Lee, and Lee 2013). The users' preferences are captured in the form of concepts. These are modelled 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 between the content and the location concepts.

(Hess, Magagna, and Sutanto 2014) assigned location tags to websites with a precision up to street level. They extracted words from the websites and checked them against free gazetteers (Geonames and OpenStreetMap) using the Aho–Corasick string matching algorithm (Aho and Corasick 1975). Validation and disambiguation of the locations were done using all the geographical references found from the text content to create a *context*. The method was reported to outperform a commercial solution (Yahoo! Placemaker) despite detecting only 60% of the locations correctly. The authors derived three practical applications from their work: location-aware web surfing through a mobile device, browsing using nearby

tags and location tagging through social networks. Our address detection approach is similar to theirs because we also used free gazetteers and a fast string matching algorithm, which in our case is a prefix tree search.

A system that is capable of handling geographical queries of the triplet of <theme><spatial relationship><location> is described in the work of (Purves et al. 2007). It handles spatial relationships such as *inside, near, north-of, south-of* and *geo-references*. The system 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 both to conceptual and to 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. Locations are detected using a gazetteer, which is enhanced with context rules. Additional name lists are used for filtering.

Another system that can be used in the named entity recognition and disambiguation steps, was described by (Qin et al. 2010). It builds and updates a set of locations, and uses them to provide evidence of geographical contextual.

The method by (Schmidt et al. 2013) identifies companies from websites by detecting their address data using patterns and gazetteers from free sources such as OpenStreetMap. The websites are pre-processed by removing HTML tags, extracting text, line splitting, tokenising 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 use a similar method to aggregate address elements, but we decided on a different method to identify service and company names. We use scoring based on visual appearance and distance from the detected address. Furthermore, our goal is more general, as we do not restrict to company websites, although most of our results still came from businesses and companies.

A knowledge-based web-mining tool is described in the work of (Li et al. 2012). The method adopts a geospatial ontology, a rule-based screening algorithm, and inductive learning for automated location retrieval. They customised address detection to discover the locations of emergency service facilities; other detected addresses were discarded. Our method is simpler, as it does not require any training or learning. Compared to (Li et al. 2012), 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 labelling algorithm was introduced by (Chang and Li 2010). Unlike most existing methods, they do not use gazetteers to detect postal addresses. Instead, they detect addresses 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 was 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 the work of (Dou and Hu 2012) extracts product data from company websites, but it applies to our application as well. The leaf nodes of the DOM tree are analysed 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 to detect locations and associated information, but it is limited to 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 by (Liu, Meng, and Meng 2010). The visual block tree (Cai et al. 2003) 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 colour. The data records are extracted using the structure of the visual block tree and the visual features of its elements.

3. Location-aware framework

We next present the overall framework of our location-aware search engine, present our choices for the components, and discuss their advantages and disadvantages. The framework is summarised in Figure 2, and it contains the following components:

- Website provider
- Web page parser
- Address detector and validator
- Title and image extraction
- Distance-based ranking
- Summarisation (form output)

3.1. Website provider

The proposed framework is based on a meta-search approach similar to that of (Leung, Lee, and Lee 2013). The *Website provider* uses external search engines such as Google and Yahoo to perform the actual search. It takes the search keyword provided by the user and submits it to external search engines for results. We gather the results that are not location-aware and post-process them by extracting locations from the provided websites. The advantage of the meta-search engine is that it does not require web pages to be crawled and indexed; instead it combines results from multiple search engines, which allows us to find relevant websites and allows users to access more information. The drawback of the meta-search is that it depends on the relevance of other search engines and is vulnerable to changes in the external search engines we use.

3.2. Web page parser

The web pages are downloaded and converted by our *Webpage parser* into tree representations based on the approach of (Tabarcea, Hautamäki, and Fränti 2010). Contrary to existing

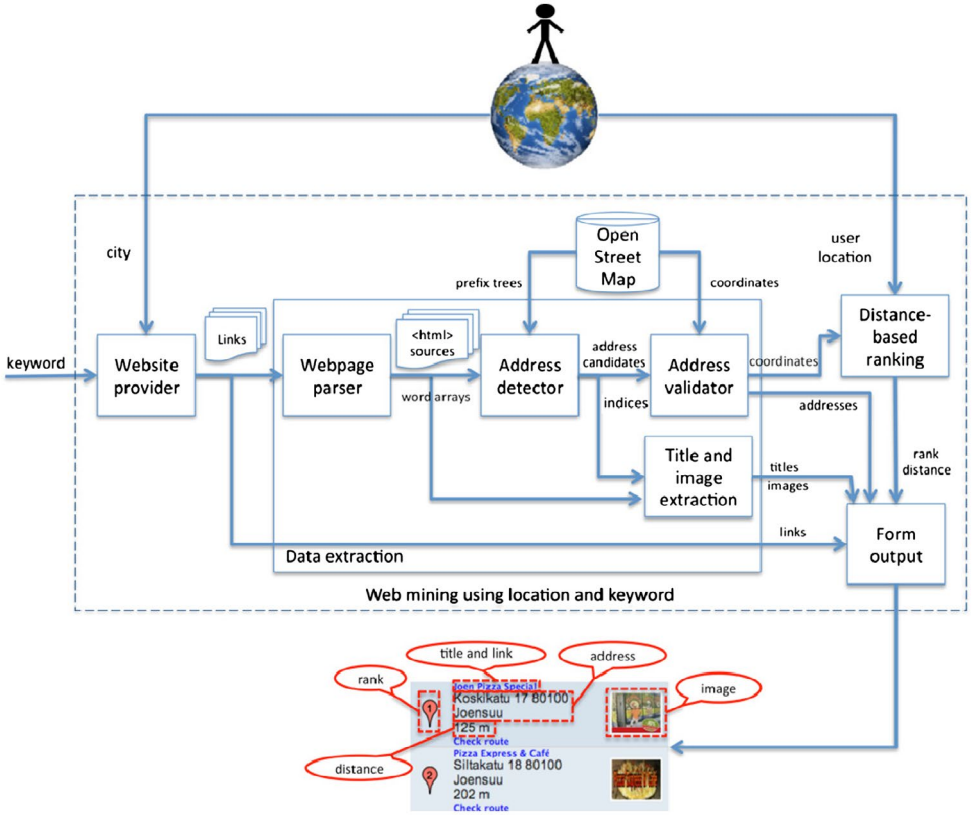

Figure 2. Location-based search framework.

methods, which concentrate merely on extracting plain text, we also extract titles and representative images using the same DOM tree representation of the provided websites. This allows better control of where to search for the additional information related to the detected locations, especially in websites that contain more than one location (see Figure 3).

3.3. Address detector

We find locations by detecting postal addresses. Our approach is similar to that of (Qin et al. 2010) 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, similar to the method of (Schmidt et al. 2013). 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 using the gazetteer to find their respective coordinates.

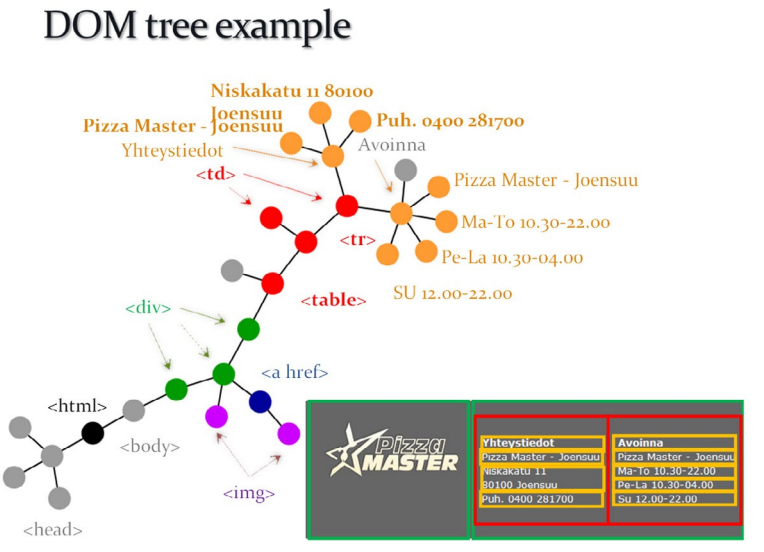

Figure 3. Example of the DOM tree representation of a part of a web page. All components can be found in this part: address, title and representative image.

3.4. Validation with Gazetteer

We optimise the gazetteer search using a fast prefix-tree based on text search, which was tested against heuristic and brute-force approaches, and our search provided better results (Tabarcea, Hautamäki, and Fränti 2010). This 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. In this paper, we further extend the previous solution, which was limited to only two locations (Finland and Singapore) by interfacing it with the largest available open-source mapping and gazetteer service, OpenStreetMaps.[4] In this way, our application can detect addresses in most countries, which improves its practical applicability. The drawback is that the use of gazetteers does not allow spelling mistakes, name variations or abbreviations.

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,[5] Bing Local,[6] or Yahoo Local,[7] our goal is the same: to 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 they 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 to utilise the location of the user but without restriction to any fixed location-based service.

In summary, we integrate the processes of geo-referencing, geo-coding and geo-tagging into a unified location-based system that is able to provide relevant information close to the user's location. The implementation is not limited to specific geographical areas, although it is dependent on the accuracy of the gazetteer we use. For this purpose, we use a gazetteer

built using OpenStreetMap data, which is available for most countries. Our system is flexible as it can detect locations from any set of web pages that contain postal addresses. It can be easily customised to work with web crawlers or custom search engines, not just the results of an external search engine.

3.5. Title and image extraction

We extract a title and representative images for each detected location to provide a brief summary of the search results in a way similar to the work of (Chang and Li 2010). The *Title and image extractor* uses the detected addresses and the DOM tree representation to identify candidate titles and images related to the address. For the image extraction, we use the solution presented by (Gali, Tabarcea, and Fränti 2015) as such. For the title extraction, we use the solution presented by (Gali, Mariescu-Istodor, and Fränti 2017a) with a few improvements. We do not associate a single location to a single website but allow the same page to contain multiple locations. This is typical for service directory web sites. For this purpose, we classify the web pages either as single, brand or service directories. A modified title selection is used in case of service directories. The scoring of the candidate titles is presented in the work by (Gali and Fränti 2016).

3.6. Ranking and summarisation

All result data are aggregated to compose the search results, which are sorted using *distance-based ranking* and displayed as a list. We rank these results by the 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. Adequate data are then displayed to the end user, as shown in Figure 4. In the mobile applications, the summarisation is simplified due to space limitations.

To sum up, most of the building blocks of our system have been studied as their own papers (Tabarcea, Fränti, and Manta 2009; Tabarcea, Hautamäki, and Fränti 2010; Gali, Tabarcea, and Fränti 2015; Gali, Mariescu-Istodor, and Fränti 2017a, 2017b). The main focus

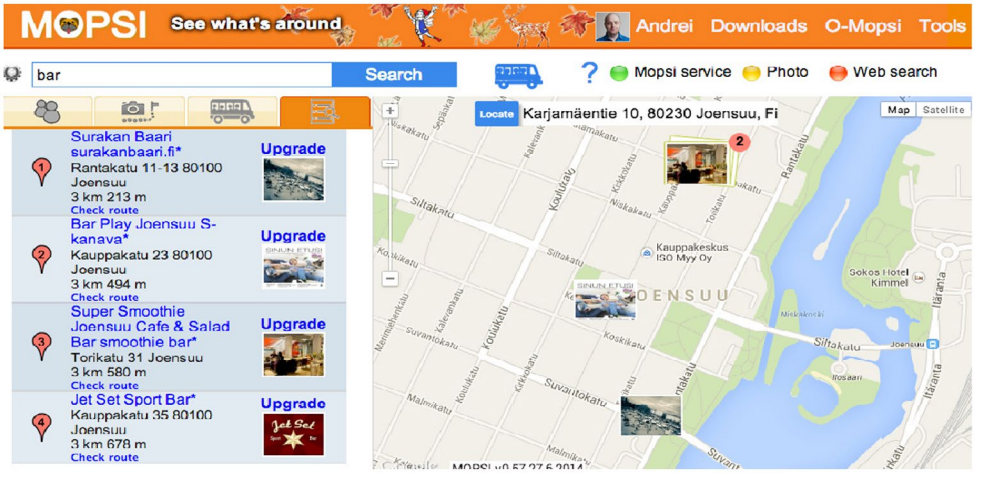

Figure 4. Web interface of Mopsi search.

of this paper is to describe the overall system, study how each of the components are constructed, and discuss their practical applicability. Our methods are implemented in the real prototype system Mopsi (Figure 4). There are also independent technical contributions, including

- an added classifier for detecting the web page functionality;
- improvements to the method for title extraction from service directories;
- a more detailed study of the prefix tree solution; and
- new results on location inclusion in web pages and address detection.

4. Extracting location-aware information

4.1. 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 (Fränti, Kuittinen et al. 2010). 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 images and distance to the location (see Figure 1). In the next section, we define all components needed to implement this. The search workflow is detailed in Figure 2.

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![8] and Custom Search by Google,[9] as they allow third parties to build search products using the infrastructure of their search engines. The website provider relies on the relevance of the results of the external search engine and 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.

The websites are processed by the *data extraction* module. This module consists of mostly self-built components based on the 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 address candidates found and the associated information.

First, the *website parser* downloads all the detected websites and converts their HTML sources into a DOM tree representation. The *address detector* then searches for postal addresses using a text matching algorithm based on street name prefix trees (Tabarcea, Hautamäki, and Fränti 2010). 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 (Tabarcea, Fränti, and Manta 2009), 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.

The *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 based on the lowest common ancestor of the nodes that contain location information, splitting the DOM tree so that each sub-tree contains a single address. Inside each of those sub-trees, the module searches for text and images associated with the detected address. The output is a list of geo-referenced entities that contains the information described in Figure 2. 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 and sorted using *distance-based ranking*. All the information is assembled by the *form output* as a list of search results. The coordinates are also used to display the results on the map and to compute the distance from the user's location. The postal address candidates that are not validated by the geo-coding services are discarded.

4.2. Parsing web pages

HTML documents are considered to be semi-structured data, which are neither raw nor strictly typed (Abiteboul 1997). HTML documents do not conform to a formal data model or 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.

An HTML document has a DOM representation, which is a platform- and language-neutral interface that allows programmes and scripts to dynamically access and update the content, structure and style of documents.[10] A DOM tree is composed from HTML elements and their parent–child relationship, having the <html> as the root of the tree. Figure 5 shows a simplified example of an HTML page, where we display just the sub-tree that contains the location and the result for this particular case. This DOM tree is used for address, title and image extraction. The three components are detailed separately in Sections 4.3, 4.4, and 4.5.

4.3. Address detection and validation

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

We use a rule-based text-matching algorithm to separately identify each type of 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-trees. This allows us to find addresses that are spreading through several nodes (e.g. if one element is bold) or through tables. The text is segmented into words, and each word is verified according to whether or not 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

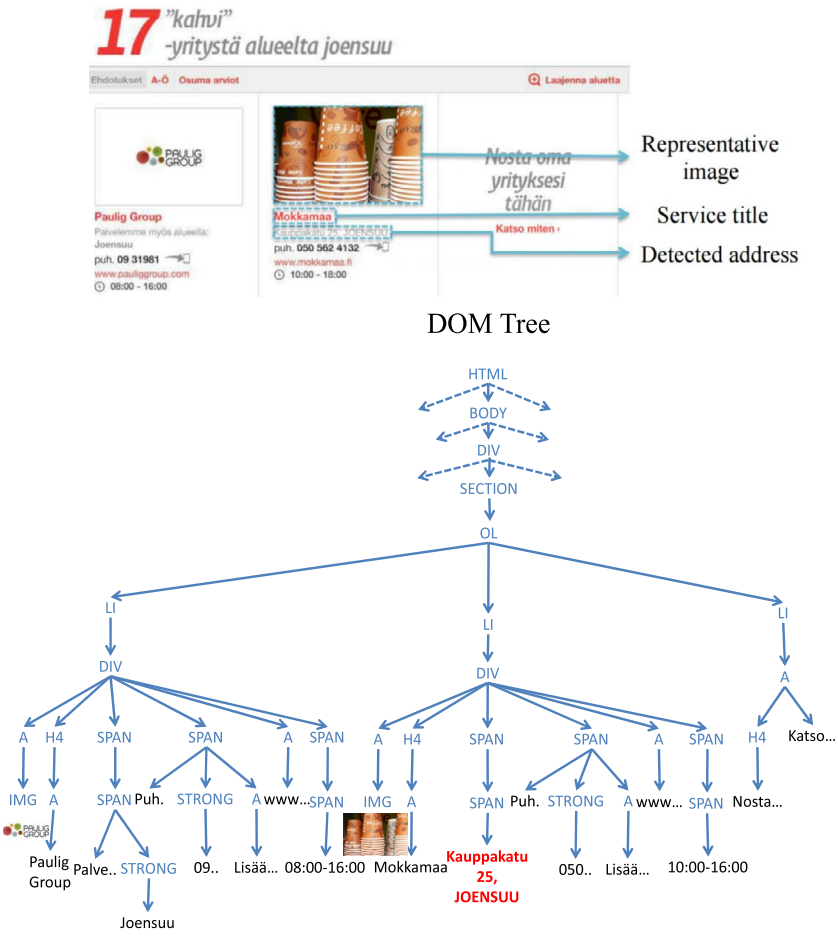

Figure 5. Part of a web page that contains location information: visual appearance (top) and DOM tree (bottom).

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.

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

```
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.

and mark all the nodes that satisfy the criterion that they contain validated locations, and will use them later in the content extraction stage.

The first step of the address detection algorithm is to identify the city names from the content of the web page. This is done 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 city considered, we build a prefix tree of street names (Tabarcea, Hautamäki, and Fränti 2010). 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.

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[11] 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.

Table 1 shows the prefix tree statistics of a small city (Joensuu) compared to three other bigger cities (Paris, Singapore, and Stockholm). The size of the prefix tree varies from 420 kB (Joensuu) to 3.5 (Stockholm). Storing it for the whole of Finland would take about 74 MB, which is negligible compared to the size of the corresponding gazetteer, which takes roughly 3 GB (Tabarcea, Hautamäki, and Fränti 2010).

Every word on a web page is checked as to whether it is a street name by searching for it in the prefix tree. The complexity is the same as string search in general. It depends linearly on the size of the web page (number of words). Searching from the prefix tree is fast: in case of a positive match, we need to retrieve down to the level corresponding to the length of the word. In case of a mismatch, each search terminates usually within a few steps. Building the tree is more time-consuming, around 10–15s, but since it can be made off-line beforehand, it does not affect the search.

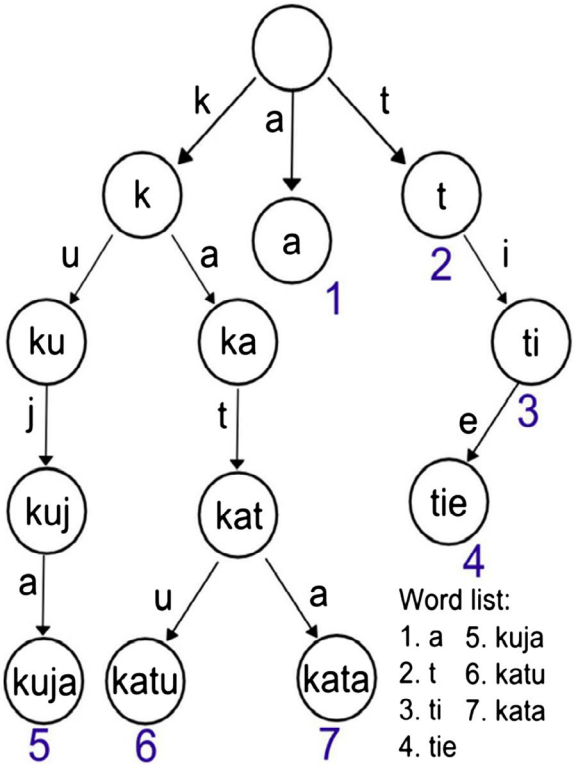

Figure 8. Prefix tree example.

Table 1. Prefix tree statistics.

	Joensuu	Paris	Singapore	Stockholm
Total number of street names	1145	6472	5316	4584
Total number of tree nodes	5430	38 323	24 062	40 189
Total number of leaf nodes	1131	4836	3274	3940
Maximum depth	20	57	63	51
Average depth for each leaf node	12.5	18.5	15.5	15.1
Average width for each level	271	672	382	788
Size (MB)	0.42	3.33	2.09	3.50

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 a maximum of 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 5). 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 as

follows: *Kauppa-* (market), *Alexanterin-* (Alexander's) or *Kaisla-* (Reed). Examples of descriptions are as follows: *-katu* (street), *-tie* (road), *-polku* (way), or *-kuja* (alley). Finnish addresses have a fixed order: street name, street number, neighbourhood, postal code and city, but neighbourhood and postal code are optional. Although the websites we use to test our methods are Finnish, our address detection algorithm is not customised for Finnish addresses and does not use a predefined order of address elements.

The proposed address detection is based on the method described in the work of (Tabarcea, Hautamäki, and Fränti 2010) 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 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 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 normalisation (Ahlers and Boll 2008a).

4.4. 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 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 also contain numerous 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 by (Gali, Mariescu-Istodor, and Fränti 2017a) 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. Such sites 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 decision-tree classifier to detect to which of these two types a given website belongs. It analyses the web link and the content of possible menu lists on the page. The page is classified as a service directory if it fulfils 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 (see Section 5).

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 dealing with very long title strings. The second rule analyses how often the same phrases are used in header tags, and the third rule determines whether the phrase also appears in the web link. The candidate phrase with the highest score is selected (Gali and Fränti 2016).

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 analyse 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 it is that the candidate relates to this address. Visual cues, such as emphasis (Table 2) and colour (Table 3), are also used. The perceptual colour difference CIE 2000 (Luo, Cui, and Rigg 2001) between the *colour* and *background-colour* 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 3, 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, analysing the layout of the page. All candidates are scored based on the distance and the visual factors; the highest scored candidate phrase is chosen.

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 in Figure 9.

4.5. 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

Table 2. 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

Table 3. Scoring CSS attributes.

CSS attributes	Score
Colour, background-colour	+ perceptual colour difference (0–10)
Font-size	+ (node font size − address node font size)
Font-weight	+3 if bold or >500
Text-transform	+5 if uppercase

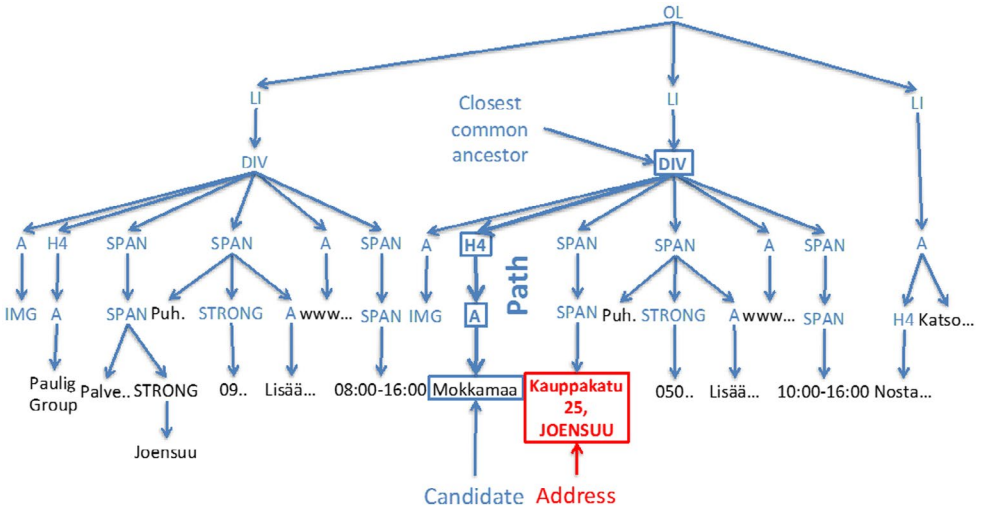

Figure 9. DOM sub-tree that contains the address of a service.

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. In the next section, we briefly review the method that was presented in the work of (Gali, Tabarcea, and Fränti 2015).

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 described 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*: recognisable images that identify the service provider (company or institution);
- *Banners*: images placed usually either above or below the website, or beside the content. They are generally used for decoration. Headers and footers are classified in this category;

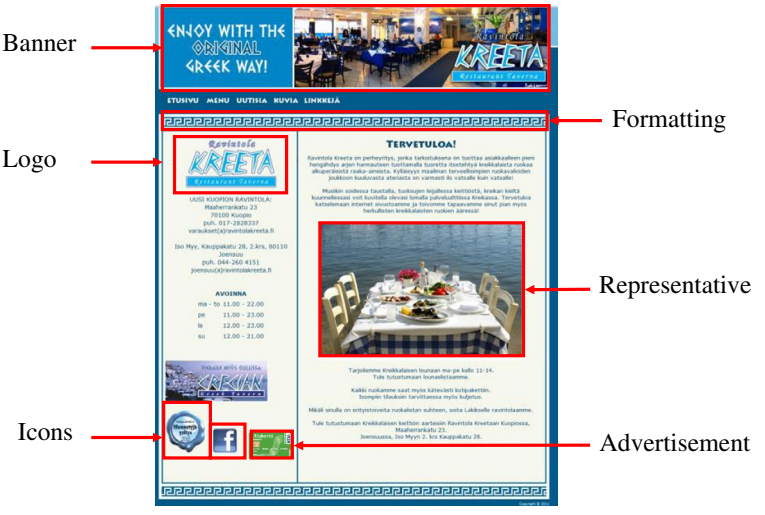

Figure 10. Image categories.

- *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.

For classifying the images, we use the rules shown in Table 4. 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 categorised 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.

5. Experiments

We tested the proposed approach for address and data extraction using the *Mopsi Services* data-set (cs.uef.fi/mopsi/data/). It contains services submitted by users of the application and confirmed by administrators. Each service has the following fields: title, web link,

Table 4. Rules used for image categorisation.

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 < 100 px Height < 100 px	Background, bg, sprite

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

Figure 11. Image features we use.

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 web link. The titles have been manually entered by the users, and the geo-location is obtained either by GPS positioning (in the case of mobile input) or manually tuned on the map (in the case of web input). Although there has been no common practice in terms of how to input the data, the most relevant data (title and address) exist for all services and can be used as a ground truth against which the results of the data extraction results can be compared.

The data-set contains 364 services from Finland; it includes a wide variety of web links such as *home, brand, business directory, Wikipedia, Facebook, blog* and information pages, which cover various domains, such as *education, health, shop, bank, entertainment, sports*, *food & drink*, *hotel & accommodation*, *travel & leisure* and *news*. Our experiments were conducted on all these data types without exception.

The services have been stored using both Finnish and English language. Some users have systematically used both languages (*kirkko*, church), but not all. The 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 and 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 in the period from 29 April to 15 May 2015, and we analysed their content. We observed that relevant information about the services, such as the title, description, address and opening hours, is commonly static because users are expected to rely on this information. However, if the target pages are expected to be fully dynamic, we recommend using a WebKit browser such as *PhantomJS*[12] to render the web page prior to the application of our method.

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 or no addresses.

First, we checked for geo-tags and <address> tags. The results in Table 5 show that just 10 (3%) of the websites have a geo-tagged 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 we 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 number might appear low, but it is high enough for our needs. First, we do not need to retrieve all information relevant to the keywords, but only those related to the location. Second, the existence of addresses is much higher in commercially oriented web pages such as those of cafes and shops, which are usually the most relevant search results. What we cannot retrieve are web pages that might be relevant to the location but do not contain addresses or other location-specific information, such as blogs and Wikipedia pages.

Most errors are results that are relevant by their content but for which no location information was detected. Thus, such results were simply omitted. Some relevant search results could be missed, but this has only limited effect on practical applicability, as users never see these. The inaccuracies cause some content-relevant results from different location to also be included. According to our experiments, such cases appear relatively rarely – 26/145 = 20% of the results. Thus, one result out of five is from different locations, although the content is probably still relevant. This level of error can be accepted already in practical applications.

The high number of websites containing postal addresses was expected because we considered service-based websites as needed for entertainment, sports, restaurants, and websites that advertise businesses, local services, or tourist locations. The small number of geo-tags and useful addresses tags emphasises 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.

For each correctly detected address, we tested the title extraction algorithm described in Section 4.4. 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 was considered correct if the similarity of the strings (extracted and the ground truth) is higher than a threshold of 0.5. The results in Table 6 show that the correct title was found in 29% of the cases if we exclude the websites in which we did not detect any address. This is significantly lower than that of the baseline method (72%). (Gali, Mariescu-Istodor and Fränti

Table 5. Results of the address detection.

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

Table 6. Results of the title and image extraction.

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

2017a) conducted a similar study as a normal web-mining task without considering the location aspect. The method described in Section 4.4 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 using website classification and the methods described by (Gali, Mariescu-Istodor and Fränti 2017b), in which we first classify a website as a single service, brand or service directory. In Table 7, we summarise the classification accuracies for the decision-tree and the clustering models using precision, recall and F-measure. We observed that both models provide good precision (83%) and recall (91%) values for detecting single service websites. The values for detecting service directory websites (88 and 89%) and (76 and 97%), respectively, are also good, considering that this is a challenging task due to their heterogeneous structures. For brand websites, our classifiers provided less satisfactory results, as only (38%) of them are detected by the decision-tree and (19%) of them are detected by the clustering. These results are due to the fact that brand websites can be as simple as single websites, such as the *Cocoa Bar* website,[13] or as difficult as service directories, such as the *Best Western Hotels* website.[14] Our classifiers misclassified more than half of them. The clustering-based model has an advantage in that it classifies service directory webpages with high accuracy (97%) in comparison to the decision-tree model (89%). This would be beneficial in applications that need a central place of information rather than searching every website individually.

Depending on the type of the website, we used either the method described in Section 4.4, the method used by (Gali and Fränti 2016), or the method described by (Gali, Mariescu-Istodor and Fränti 2017b). The method tested in this paper shows clear limitations for websites that are not service directories, but our experiments in the work of (Gali, Mariescu-Istodor and Fränti 2017a) show more promising results.

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 (Gali, Tabarcea and Fränti 2015).

Table 7. Classification accuracy for decision-tree and clustering-model website classifiers.

	Decision-tree (%)			Clustering (%)		
Type of website	Precision	Recall	F-measure	Precision	Recall	F-measure
Single	83	91	87	83	91	87
Brand	46	38	42	44	19	26
Service directory	88	89	91	76	97	88

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 the work of (Gali, Tabarcea, and Fränti 2015), where a more extensive set of 1032 websites was evaluated. The accuracy reported there was 64%, which was significantly higher than that of Google+ (48%) and Facebook (39%).

The relevance of the overall search results was not studied, but we briefly recall a previous study comparing between GoogleMaps and Yellow Pages in (Fränti, Tabarcea et al. 2010). We chose 10 test keywords consisting of five commercial (hotel, restaurant, pizzeria, cinema and car repair) and five non-commercial ones (hospital, museum, police station, swimming hall and a church). Mopsi provided more results (2352) than Google Maps (1405) or Yellow Pages (1597), and the results were slightly more relevant (1.59) than Google Maps (1.66), but always less relevant than Yellow Pages (1.28). Mopsi was better especially with rural non-commercial keywords.

6. 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 made 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, would have lower success rates.

Notes

1. http://www.w3.org/2003/01/geo/.
2. http://www.whois.net.
3. http://www.geonames.org.
4. http://www.openstreetmaps.org.
5. http://maps.google.com.
6. http://www.bing.com/local.
7. http://local.yahoo.com.
8. http://developer.yahoo.com/boss.
9. https://developers.google.com/custom-search.
10. http://www.w3.org/DOM.
11. http://nominatim.openstreetmap.org.
12. http://phantomjs.org/.
13. http://www.cocoabarnyc.com/.
14. http://www.bestwestern.fi/hotels/best-western-hotel-savonia-kuopio-91083.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Tekes [grant number 70010/12] and [grant number 70052/09].

ORCID

Najlah Gali http://orcid.org/0000-0001-6038-0875
Pasi Fränti http://orcid.org/0000-0002-9554-2827

References

Abiteboul, S. 1997. "Querying Semi-structured Data." In *Proceedings of the 6th International Conference on Database Theory*, 1–18. London: Springer-Verlag.

Ahlers, D., and S. Boll. 2008a. "Retrieving Address-based Locations from the Web." In *Proceedings of the 5th International Workshop on Geographic Information Retrieval*, 27–34. New York, NY: ACM.

Ahlers, D., and S. Boll. 2008b. "Urban Web Crawling." In *Proceedings of the First International Workshop on Location and the Web*, 25–35. New York, NY: ACM.

Aho, A. V., and M. J. Corasick. 1975. "Efficient String Matching: An Aid to Bibliographic Search." *Communications of the ACM* 18 (6): 333–340.

Bennett, P. N., F. Radlinski, R. W. White, and E. Yilmaz. 2011. "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*, 135–144. New York, NY: ACM.

Borges, K. A., A. H. Laender, C. B. Medeiros, and C. A. Davis Jr. 2007. "Discovering Geographic Locations in Web Pages Using Urban Addresses." In *Proceedings of the 4th ACM Workshop on Geographical Information Retrieval*, 31–36. New York, NY: ACM.

Buyukokkten, O., J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar. 1999. "Exploiting Geographical Location Information of Web Pages." *Proceedings of the ACM SIGMOD Workshop on the Web and Databases*, 1–6.

Cai, D., S. Yu, J. R. Wen, and W. Y. Ma. 2003. *VIPS: A Vision-Based Page Segmentation Algorithm*. Technical Report MSR-TR-2003-79. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2003-79.pdf

Cai, W., S. Wang, and Q. Jiang. 2005. "Address Extraction: Extraction of Location-Based Information from the Web." In *Proceedings of the 7th Asia-Pacific web conference on Web Technologies Research and Development*, 925–937. Berlin-Heidelberg: Springer-Verlag.

Can, L., Z. Qian, M. Xiaofeng, and L. Wenyin. 2005. "Postal Address Detection from Web Documents." In *Proceeding of the International Workshop on Challenges in Web Information Retrieval and Integration,* 40–45. Los Alamitos, CA: IEEE Computer Society.

Chang, C. H., and S. Y. Li. 2010. "MapMarker: Extraction of Postal Addresses and Associated Information for General Web Pages." In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, 105–111. Washington, DC: IEEE Computer Society.

Ding, J., L. Gravano, and N. Shivakumar. 2000. "Computing Geographical Scopes of Web Resources." In *Proceedings of the 26th International Conference Very Large Data Bases*, 545–556. San Francisco, CA: Morgan Kaufmann.

Dou, W., and J. Hu. 2012. "Automated Web Data Mining Using Semantic Analysis." In *International Conference on Advanced Data Mining and Applications*, 539–551. Berlin Heidelberg: Springer.

Florczyk, A. J., F. J. López-Pellicer, P. Muro-Medrano, J. Nogueras-Iso, and F. J. Zarazaga-Soria. 2010. "Semantic Selection of Georeferencing Services for Urban Management." *Journal of Information Technology in Construction (ITcon)* 15 (8): 111–121.

Fränti, P., J. Kuittinen, A. Tabarcea, and L. Sakala. 2010. "MOPSI Location-Based Search Engine: Concept, Architecture and Prototype." In *Proceedings of the 2010 ACM Symposium on Applied Computing*, 872–873. New York, NY: ACM.

Fränti, P., A. Tabarcea, J. Kuittinen, and V. Hautamäki. 2010. "Location-based Search Engine for Multimedia Phones." In *International Conference on Multimedia and Expo (ICME),* 558–563. IEEE.

Gali, N., and P. Fränti. 2016. "Content-based Title Extraction from Web Page." In *International Conference on Web Information Systems and Technologies (WEBIST 2016)- Volume 2,* edited by Karl-Heinz Krempels, Paolo Traverso, Tim A. Majchrzak, Valérie Monfort, 204–210. Setúbal: SCITEPRESS.

Gali, N., A. Tabarcea, and P. Fränti. 2015. "Extracting Representative Image from Web Page." In *International Conference on Web Information Systems and Technologies (WEBIST 2015)*, edited by Valérie Monfort, Karl-Heinz Krempels, Tim A. Majchrzak, and Ziga Turk, 411–419. Setúbal: SCITEPRESS.

Gali, N., R. Mariescu-Istodor, and P. Fränti. 2017a. "Using Linguistic Features to Automatically Extract Web Page Title." *Expert Systems with Applications* 79: 296–312.

Gali, N., R. Mariescu-Istodor, and P. Fränti. 2017b. "Functional Classification of Website." International Symposium on Information and Communication Technology (SoICT), December.

Goldberg, D. W., J. P. Wilson, and C. A. Knoblock. 2007. "From Text to Geographic Coordinates: The Current State of Geocoding." *URISA-WASHINGTON DC*- 19 (1): 33.

Hess, B., F. Magagna, and J. Sutanto. 2014. "Toward Location-aware Web: Extraction Method, Applications and Evaluation." *Personal and Ubiquitous Computing* 18 (5): 1047–1060.

Hu, Y. H., S. Lim, and C. Rizos. 2006. "Georeferencing of Web Pages Based on Context-Aware Conceptual Relationship Analysis". http://cs.uef.fi/pages/franti/lami/papers/Georeferencing%20of%20Web%20Pages%20based%20on%20Context-Aware_Conceptual%20Relationship%20Analysis.pdf

Lee, H. C., H. Liu, and R. J. Miller. 2007. "Geographically-sensitive Link Analysis." In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 628–634. Washington, DC: IEEE Computer Society.

Leung, K. W. T., D. L. Lee, and W. C. Lee. 2013. "PMSE: A Personalized Mobile Search Engine." *IEEE Transactions on Knowledge and Data Engineering* 25 (4): 820–834.

Li, W., M. F. Goodchild, R. L. Church, and B. Zhou. 2012. "Geospatial Data Mining on the Web: Discovering Locations of Emergency Service Facilities." In *International Conference on Advanced Data Mining and Applications*, edited by S. Zhou, S. Zhang, and G. Karypis, 552–563. Berlin Heidelberg: Springer.

Liu, C., P. L. P. Rau, and F. Gao. 2010. "Mobile Information Search for Location-based Information." *Computers in Industry* 61 (4): 364–371.

Liu, W., X. Meng, and W. Meng. 2010. "ViDE: A Vision-based Approach for Deep Web Data Extraction." *IEEE Transactions on Knowledge and Data Engineering* 22 (3): 447–460.

Luo, M. R., G. Cui, and B. Rigg. 2001. "The Development of the CIE 2000 Colour-difference Formula: CIEDE2000." *Color Research & Application* 26 (5): 340–350.

Markowetz, A., Y. Y. Chen, T. Suel, X. Long, and B. Seeger. 2005. "Design and Implementation of a Geographic Search Engine." *In Eighth International Workshop on the Web and Databases*, edited by AnHai Doan, Frank Neven, Robert McCann, and Geert Jan Bex, 19–24. http://pages.cs.wisc.edu/~anhai/papers/webdb05_eproceedings.pdf

McCurley, K. S. 2001. "Geospatial Mapping and Navigation of the Web." *Proceedings of the 10th International Conference on World Wide Web*, 221–229. New York, NY: ACM.

Mikheev, A., M. Moens, and C. Grover. 1999. "Named Entity Recognition without Gazetteers." In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics Linguistics*, 1–8. Stroudsburg, PA: Association for Computational Linguistics.

Page, L., S. Brin, R. Motwani, and T. Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report. Stanford InfoLab.

Patterson, C. A., R. R. Muntz, and C. M. Pancake. 2003. "Challenges in Location-Aware Computing." *IEEE Pervasive Computing* 2 (2): 80–89.

Purves, R. S., P. Clough, C. B. Jones, A. Arampatzis, B. Bucher, D. Finch, G. Fu, H. Joho, A. K. Syed, S. Vaid, and B. Yang. 2007. "The Design and Implementation of SPIRIT: A Spatially Aware Search Engine for Information Retrieval on the Internet." *International Journal of Geographical Information Science* 21 (7): 717–745.

Qin, T., R. Xiao, L. Fang, X. Xie, and L. Zhang. 2010. "An Efficient Location Extraction Algorithm by Leveraging Web Contextual Information." In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 53–760. New York, NY: ACM.

Schmidt, S., S. Manschitz, C. Rensing, and R. Steinmetz. 2013. "Extraction of Address Data from Unstructured Text Using Free Knowledge Resources." In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, Article No. 7. New York, NY: ACM.

Shi, G., and K. Barker. 2011. "Extraction of Geospatial Information on the Web for GIS Applications." In *10th IEEE International Conference on Cognitive Informatics & Cognitive Computing,* 41–48. IEEE.

Silva, M. J., B. Martins, M. Chaves, A. P. Afonso, and N. Cardoso. 2006. "Adding Geographic Scopes to Web Resources." *Computers, Environment and Urban Systems* 30 (4): 378–399.

Tabarcea, A., P. Fränti, and V. Manta. 2009. "Using a Spatial Database in a Location-based Search Application." *Buletinul Institutului Politehnic Iasi*, 55–62.

Tabarcea, A., V. Hautamäki, and P. Fränti. 2010. "Ad-Hoc Georeferencing of Web-pages Using Street-name Prefix Trees." In *International Conference on Web Information Systems and Technologies,* 259–271. Berlin Heidelberg: Springer.

Tsai, F. S. 2011. "Web-Based Geographic Search Engine for Location-Aware Search in Singapore." *Expert Systems with Applications* 38 (1): 1011–1016.

Vänskä, I. 2004. "Using Location Information in Web Documents." *University of Eastern Finland*.

Viola, P., and M. Narasimhan. 2005. "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*, 330–337. New York, NY: ACM.

Wang, C., X. Xie, L. Wang, Y. Lu, and W. Y. Ma. 2005. "Detecting Geographic Locations from Web Resources." In *Proceedings of the 2005 Workshop on Geographic Information Retrieval*, 17–24. New York, NY: ACM.

Watters, C., and G. Amoudi. 2003. "GeoSearcher: Location-Based Ranking of Search Engine Results." *Journal of the American Society for Information Science and Technology* 54 (2): 140–151.

Yokoji, S., K. Takahashi, and N. Miura. 2001. "Kokono Search: A Location Based Search Engine." In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*. http://www10.org/cdrom/posters/p1146/index.htm

Zhang, Q., P. Jin, S. Lin, and L. Yue. 2011. "Extracting Focused Locations for Web Pages." In *International Conference on Web-Age Information Management*, 76–89. Berlin: Springer.