Abstract

We propose the Context-Aware Browser (CAB), a new approach to context-aware Web content perusal by means of mobile devices. The CAB is a Web browser running on mobile devices, exploiting artificial intelligence techniques and mixing several ingredients.

The novelty of the proposed approach is threefold. First, our proposal allows a different perspective in the development of Web contents, which should be conceived in a context-dependent way. Our framework makes it possible to develop information and web applications specifically for a given environment, which will become capable to transmit context information, and ad hoc, context-based applications to whom will enter that context. Second, we exploit several informations provided by the surrounding environment and not only the location in order to carry out a more refined search of web contents and applications. Third, the CAB approach involves an innovative two stage retrieval and a filtering process, that minimizes privacy issues.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval
Categories and Subject Descriptors: J.9.a [Mobile Applications]: Location-dependent and sensitive

Keywords: Context-aware information processing, Location-based search for mobile devices, Evaluation.

1 Introduction

The typical scenario of a user seeking information on the Web requires a somehow significant effort in order to get the desired information (web pages, applications, resources, etc.). The user must provide a query using a web search engine (e.g., Google) and she also has to check the results, in order to see if they really provide the desired information. Often the query must be refined, re-submitted
and so on, giving rise to an iterative process. Thus, this information seeking activity is rather cumbersome for the end user both from a cognitive viewpoint (thinking of the right query to submit, checking the obtained results) and from a practical one (lots of keystrokes, screen scrolls: in general many interactions with the user interface).

If this can be acceptable in the everyday use of a desktop system, it becomes a serious issue when the same task has to be carried out on a PDA or a mobile phone. Indeed, the graphical user interfaces and the input/output peripherals of such devices are rather limited when compared to their counterparts on a classical PC. Moreover, the user can be in a hurry (for example, being late for the check-in at the airport) or absent-minded by focusing elsewhere in the environment; whence she cannot be engaged in a very complex searching task, where the interaction with the device interface is fundamental.

Nevertheless, in a world where information plays an essential role, it can be useful or even crucial to get the desired information quickly even when one is away from a classical computer. For instance, in an airport it could be useful for a passenger to receive on her own cellular phone the details of her flight (location of the check-in desk, gate number, etc.) without having her to put too much effort in retrieving such information. To this respect, in our vision the best solution would be to develop an infrastructure capable of automatically push such data as soon as the passenger enters the airport.

In addition, another important, and much discussed, limitation of current search engines is to neglect or to limit to the geographical position the context of the user submitting a query. This “unawareness” implies that the answer generated according to the query itself will not take into account many data (potentially suggested by the surrounding environment) that could refine the set of results in a useful way, providing the user with a more suitable and exploitable information. Going back to the previous example, a simple event like the arrival of a passenger (which can be spotted-out, e.g., by means of the Bluetooth device on her cellular phone) could be used to determine the context “the user is in the airport”. Then, if the passenger had updated her agenda with a note about the flight number, such data could be communicated (if the user agrees, of course) by a suitable background smart application to the airport server (for example exploiting again the Bluetooth connection), in order to receive back the details of the flight. In this case the event of receiving the flight number would enrich the previously inferred context, determining that “the user is at the airport in order to take that particular flight”. This in turn could automatically generate further information “pushing” towards her mobile phone. Notice that, in the whole process, no effort is required on behalf of the user, besides enabling the Bluetooth antenna of her mobile device and activating the above mentioned “smart application”.

With these considerations in mind, we propose a new approach in the production and fruition of web contents, i.e., what we call Context-Aware Browser (CAB for short), which will play the primary role of the “smart application” briefly sketched in the previous scenarios. More in detail, the CAB will be able to collect the data from the surrounding environment (e.g., interfacing with physical sensors or information systems), to infer a description of the context by exploiting several non-trivial artificial intelligence techniques (e.g., inferential networks), to search contents and applications suitable for the current context, to automatically load and display the selected web contents and to automatically
offer web pages and web applications.

With respect to other approaches in the literature there are some significant differences.

First of all, in our approach we pay particular attention on the retrieval process and the related models and techniques. Because of the huge amount of contextual information which needs to be processed by the system and its heterogeneity and uncertainty, and because of the huge amount of web contents (applications, pages, etc.) which have to be managed and filtered, the performances of the retrieval process play a key role in our context-aware framework.

Secondly, the approaches in the literature are limited in the contextual values adopted, being based mainly on location, time, and user identity. We agree that these kinds of information are those carrying the largest amount of knowledge, but there are also other relevant contextual values which can be useful to accomplish the users’ needs, and we believe it is not correct to discard them a priori. Location is the main contextual dimension exploited, since it is used to retrieve the right inferential network; however the CAB is not a simple location-aware browser. Once the inferential network is loaded on the device, detailed contextual values are taken into consideration and high level information like activities, needs, etc. can be inferred.

Finally, while for the end user the CAB is a new way to access to web pages and applications, for the developers it represents a different approach to the design and development of contents. As the diffusion of the browser Mosaic in early nineties encouraged people to organize the information in the form of hypertext, in the same way the CAB might drive people to organize contents in the new form of Web applications and related inferential networks. This means that we are not going to write directly the functions for managing each daily life aspect (going to airport, renting a car, arriving at a hotel, etc.); rather, we provide just the common framework and delegate the implementation of specific services to existing service providers wanting to exploit the CAB approach. The aim of this paper is to present CAB architecture (Section 3), to show some scenarios of its usage (Section 4), to describe the evaluation of the retrieval approach (Section 5), and to sketch future developments (Section 6). We start by describing related work in the next section.

2 Related work: context-awareness and mobile computing

Several research fields are relevant to our approach. We briefly survey the work that has been done on context-aware computing and on context-aware retrieval.

2.1 Context-aware computing

Context-aware computing can be defined as the use of context in software applications, where the applications adapt to discovered contexts by changing their behavior.

Although a lot of research has been carried out within context-aware systems, the core term context is not yet a well defined concept: often context is ad-hoc defined and represented from project to project. In particular through the years several different definitions have been proposed. Theoretical definitions like the
one proposed in [2] come together with more application-related definitions, where the context is defined extensionally, through a list of associated values (like for example time, location, personal information, activities, etc.). Extensional definitions are not complete, as the context cannot be outlined just by some of its aspects, however they seem useful in practical applications, where the abstract concept of context has to be made concrete for dealing with real-world context-aware information. Thus, in this work we adopted this perspective.

In most of earlier context-aware applications, the notion of context included just a small amount of data. While most researches, for example, focused on time and/or location and other few data, more complex approaches tended to combine several contextual values to generate new contextual information, see, e.g., Abowd et al. [1] and the TEA Project [10]. Following these researchers, we intend to combine contexts to determine new, more abstract contexts. Differently from them, however, we do not want to limit the inferences only to an a priori defined set of contextual dimensions; rather, we aim to develop an inferential infrastructure able to work in a general and collaborative way.

The system we propose is based on the convergence of Web and context-aware computing. The idea is not new, since some researchers have already begun exploring the possibilities related to this convergence. The very first attempt is represented by the Stick e-note framework [3], that is based on the idea of “virtual Post-It”. More recent and advanced examples, as [7], on the contrary, concern with the association of Web contents and services with physical entities (objects, places, and other people). Starting from these ideas we want to improve the Web presence in the real world, aiming at a “physical browsing”, that is browsing the digital world based on situations in the real world.

2.2 Context-aware retrieval

Context-aware retrieval (CAR) is a rather new field of interest whose importance has increased with the rise of context-aware applications and the increasing amount of information managed by them. According to [8], it can be described as an extension of classical information retrieval that incorporates the contextual information into the retrieval process, with the aim of delivering information relevant to the users within their current context. CAR is based on an automatic use of the context for augmenting, refining, and improving a user’s search query in order to obtain more relevant results, or to define new queries. Thus, according to [4], the CAR model includes, among the classical Information Retrieval (IR) model elements, the user’s context, that is both used in the query formulation process and associated with the documents that are candidate for retrieval.

CAR applications can be either interactive, if the user directly issues a request to retrieve relevant information, or proactive, if the documents are presented to the user automatically [4]. Typically, CAR applications present the following characteristics [8]: user mobility, i.e., a user whose context is changing; interactive or with automatic actions if there is no need to consult the user; time dependent, since the context may change; appropriate and safe to disturb the user.
3 The Context-Aware Browser

3.1 Description

In a previous work [5], we described MoBe, a general architecture for context-aware distributed applications on mobile devices. MoBe is based on the dynamic and automatic download, configuration, execution of applications on the basis of the user’s context. In this way, a device is not limited to a set of predetermined functionalities, but allows to adopt those which are (likely) more useful for the user at a given time in a given place.

With the emergence of AJAX technology on desktop web browsers, and on mobile devices browsers as well, it has been natural to think of adapting MoBe architecture to a Web oriented paradigm. Moreover this shift allows the simplification of the effort required for using such a kind of system, that was one of our most crucial aims. The web browser is now a daily use tool for most people, while mobile software applications like midlets are still somewhat unconventional. Moreover, with a web oriented approach, the currently existing web contents could be easily re-used (whereas previous applications have to be re-developed from scratch). Finally, we believe that also the developers, in these days, are more familiar with web applications.

In the process of redesigning the MoBe architecture in a web-oriented fashion, we came up with a new and more general kind of system that we named Context-Aware Browser. The main idea behind CAB is to empower a generic mobile device with a browser being able to automatically and dynamically load web pages, services and applications selected according to the current context the user is in. Despite the apparent simplicity of this approach, a more thorough definition has to take into account several features; whence, we can say that the Context-Aware Browser is best described by the sum of the following parts:

- a Web browser: CAB is able to interpret and render in a sound way (X)HTML code, to interpret client side (e.g., JavaScript) code, and to fully exploit AJAX-based applications;
- a software application for mobile devices: a device tied to a fixed location cannot fully exploit the context-awareness of our browser, since a fixed location entails a more static context;
- a context-aware application: CAB is able to automatically retrieve and constantly update the contextual information gathered from the surrounding environment (this feature allows CAB to provide contents varying in dependence of the current context of the user);
- a search engine: CAB is able to search both for “traditional” web pages and applications on the Web and for specifically tailored applications as we will see in the following;
- a proactive application, able to automatically search for and download contents: the resources retrieved by the search engine are automatically filtered against user’s preferences and several other parameters, in order to reduce the cognitive load imposed on the user by automatically selecting the most appropriate contents;
• an application working on any kind of Web content, i.e., both Web pages and applications: CAB is able to manage both static resources (e.g., plain (X)HTML pages) and dynamic ones requiring user interaction (e.g., web applications).

3.2 The CAB architecture

3.2.1 Modules

We provide some more details on CAB implementation. Figure 1 shows CAB overall three-tiered architecture, where the topmost layer manages the interaction with the user, the medium layer acts as a bridge between the topmost and the inner layer, which in turn is responsible for sensors, inferential network, and filtering engine management.

![CAB overall architecture](image)

The main modules of CAB architecture are:

• **Context server** and **sensors**. They are responsible for collecting and sending to the CAB Core the contextual information gathered from the surrounding environment, in the form of an inferential network that represents (part of) the knowledge base of the **Context manager**. Then, all the
sensors specified are instantiated by the Context manager. Such sensors communicate with the Context server in order to receive the contextual information (location, date/time, temperature, etc.), in either push or pull mode. If the user moves to a new environment, new context servers will become available, the inferential network will change, and some sensors will be no more meaningful, so they will be dropped by the Context manager.

• The Context manager, and its internal inferential system is the core component responsible for the synthesis of the current context, starting from the contextual information gathered by sensors.

• The Filter module and the Descriptor search engine are in charge to search for and filter the most suitable contents according to the inferred context. The contextual information, obtained from the Context manager, is represented through a Context Descriptor that holds the most important information related to the user’s current context. CAB allows the user to specify which filtering information is public (may be used to formulate the query to be sent to the Descriptor search engine out of the device) and which one is private (and may be used only in “internal” filtering activities), in order to achieve a reasonable privacy level. The alternative of sending a more detailed query to the Descriptor search engine would be simpler but less privacy-aware and more open to malicious eavesdroppers attacks.

• The Connector is an infrastructure component used to separate the user interface from the context management part. This abstraction allows to easily change the way users interact with the system, or to modify the techniques exploited for the current context inference and management. Currently, as the user interface is Web-browser based, the Connector is implemented as a local web server, being able to open HTTP connections and to manage browser’s requests.

• The Browser module manages the presentation of the web contents to the user and it also automatically starts the most relevant ones. Such features are implemented in two sub-modules: the AJAX engine and the User interface.

3.2.2 Context inference

The current user’s context is composed by an undefined number of contextual values. Each value is described by two elements: an unambiguous ID, categorized using a simple ontology, and a probability value. We divide contextual values in Concrete contexts, that represent the information obtained by a set of sensors, e.g., “temperature: 20°C” or “12:30”, and abstract contexts that represent everything that can be inferred from concrete contexts like, for example, “user at home”. The aim of the inferential system is to combine concrete contexts to determine abstract contexts and to combine abstract contexts to obtain new, more abstract contexts.

We designed a two stage inferential mechanism, where both rules and Bayesian networks are used. We define the combination of rules and Bayesian networks
as the inferential infrastructure (Figure 2). The input to the inferential infrastructure is represented by concrete contexts. Concrete contexts are processed by a rule-based system in order to simplify the information and map them to the starting node of the Bayesian network, that represents the second and main stage of the inferential infrastructure, where concrete contexts are transformed into abstract ones. This represents all the knowledge machinery needed for context inferences, combining concrete and inferred abstract contexts into the current user’s context. We decided to adopt the combination of rules and Bayesian networks, since they are well-known and consolidated models in AI. A possible alternative model, that we are currently considering, could be Conditional Random Fields; the latter can be seen as a probabilistic framework where one can build all rules or prior knowledge as a feature function and let the framework learn weights for all these rules.

The following is a simple example. The mobile device senses the data: “proximity-bluetooth=001122334455, time=07:15, light-lux=50, temperature-celsius=22, movement=0”. Through the first rule-based stage, these data are mapped into the Bayesian network starting nodes. For example “proximity-bluetooth=001122334455” is mapped on the node “location=kitchen”, while “light-lux=50” is mapped on the node “light=medium”. Starting from these nodes the Bayesian network can infer the abstract context “having breakfast” with a high probability level. With different concrete contextual data, the Bayesian network would have inferred the same abstract contexts with a low probability value, or it would have inferred different abstract contexts.
It is important to remark that both the rules and the Bayesian networks are not fixed in the Context manager, but they are dynamically obtained on the basis of the environment the user is in. When the user enters her home, the device will obtain the inferential network specialized on the contexts related to the home; when the user goes to work, the device discards the inferential network related to home and loads the office related one. In this way, instead of having a single huge inferential network, we combine smaller subnetworks, where each of them meets the possible demands one might encounter in the environment the network has been designed for.

Therefore, each inferential network is predefined and developed specifically for an environment. Once the inferential network is on a mobile device, user preferences and habits can be exploited to adapt it to each single user. This personalization step acts like a layer put on the inferential network that can remove or add contexts, change the probability distribution, remove or add arcs between contexts.

4 Usage scenarios

In order to show how the CAB can be adopted in realistic environments, we propose two different scenarios describing a typical day of a person and a visit to a museum using a device equipped with the CAB. The CAB prototype has been implemented for different software platforms (Symbian™, Windows Mobile™, iPhone™, and Android™).

4.1 A typical day

It is seven o’clock in the morning and the user just woke up: this context causes the download and execution of a WakeUp Content allowing the user to manage the lights, the television and stereo sets, the heater, etc. When the user is having breakfast, see also Section 3.2.2, the CAB provides another functionality to support the user’s daily routine: it presents, on the basis of user’s preferences, a web content with the news of the day and the weather forecast (Figure 3(a)).
Figure 4: CAB as tourist guide in a museum.

Our example goes on with the user entering her own car to drive to work. The CAB detects the new contextual information from the external environment (e.g., the presence of the Bluetooth installed in the car) and downloads new specific web contents providing information about the car and allowing the user to personalize and adapt the car settings to her desire.

The user drives into a highway; when she is near the tollbooth, the CAB figures out the new context and retrieves the relevant web content (Figure 3(b)) providing information about the specific highway route. While driving, the CAB detects the presence of a highway stop, i.e., a petrol station with food and beverage services, and retrieves a specific web content (Figure 3(c)), which informs the user about the estimated distance and the services available.

4.2 At the museum

A tourist decided to visit a museum. As she enters the museum main hall with her own mobile device, a Web page with the list of the collections available in the museum is presented to her (e.g., the official museum home page, or an adapted version).

Moving around the museum the tourist is continuously guided by the system that ubiquitously and contextually retrieves the most interesting information (Figure 4): moving towards an artwork the tourist will automatically obtain the detailed description of that artwork. The localization can be performed both using RFID sensors, and exploiting wireless connections (e.g., Wi-fi, Bluetooth, Zigbee) and triangulation/trilateration on them. Because of the huge amount of information that is relevant to each artwork, the ubiquitous search has to be precise and has to provide only the information interesting and important for the tourist, based on her preferences, activities, interests and based on the community behavior and on the accuracy of the same information.

During her stay in the museum, the tourist is not only a visitor, but she can
become a content producer (as foreseen in [6]). The tourist can capture content at the point of inspiration and upload it in real-time. Tourists can create, add, modify, and delete contents about artworks; content can be of different kinds: photos, videos, audios, text (comments or posts), drawings, etc. These new contents become part of the artwork description and improve it. Thus an artwork is represented by a lot of information from both museum curators and the visitors. All contents can be judged by users: in this way, the visitors’ community is the reviewer of its own contents.

5 Evaluation

The CAB development is currently at an intermediary stage: although the components are available, and part of them, as the Context manager, have been already evaluated, the retrieval mechanism needs a more accurate implementation. In particular an early evaluation of different strategies to automatically build queries starting from the user’s context is needed. To study the different strategies a user testing evaluation could have been premature, so we adopted a TREC-like benchmark evaluation (http://trec.nist.gov/), named CREC (CAB Retrieval Evaluation Collection) [9].

5.1 The CREC benchmark

CREC is constituted by three components: topics, document collection, and relevance judgments. Topics (information need descriptions) are represented by context descriptors, which define different user’s contexts in different domains. Context descriptors have been designed similarly to the TREC topics; CREC includes 10 context descriptors which differ for user activities, location, time, etc. The following is an example of context descriptor:

<contextDescriptor>
<title> Heathrow airport </title>
<description>
The user has just landed at London Heathrow international airport. He is looking at a flight timetable and at a timetable for connections to London. It is lunch time.
</description>
<narrative> ... </narrative>
<relevance>
A Web page is relevant: it contains information about a flight, about the means of transport to reach town, about bars and fastfoods in the airport, or it allows to book a flight. A web page that contains only one of these aspects is relevant; if it contains some links to relevant pages is partially relevant. If the judge is not able to judge the page, its value is ‘‘I don’t know’’.
</relevance>
...
</contextDescriptor>

The context descriptors have been provided in textual form in order to simplify the evaluation of the retrieval system. However, the same, or rather an equivalent, contextual information can be automatically synthesized by the in-
ferential engine starting from basic sensor values. For instance, as the user lands at the Heathrow airport, her device downloads the inferential network provided by the airport system. Combining this information with the user’s current position (given, e.g., by GPS coordinates or trilateration through Wi-Fi or Bluetooth fields), the CAB can infer that the user has just landed at Heathrow. Moreover, when the mobile device will detect the presence of a short-range Bluetooth field near, e.g., the flight timetables and a steady RSSI (Received Signal Strength Indication) value, the inferential engine will derive that there is a high probability that the user is standing still, looking at the timetables.

The documents collection consists of web pages. The relevance judgments have been made by a unique judge using a four level relevance scale. Due to the evolving behavior of the Web, we opted for a dynamic collection, that evolves during the tests. Therefore our benchmark is not static: if a new implementation of the CAB external search engine needs to be evaluated, CREC will not contain, in general, all the retrieved pages. Since this would make the evaluation not reliable, our approach is that the collection will be extended by including the newly retrieved documents, and judging them.

In particular we built two CREC versions so far. The first version has been implemented performing 5 manual queries for each context descriptor, and judging the first 150 single retrieved documents for each context descriptor. Starting from this version of the collection, we adopted an “interactive search and judge” approach, obtaining its second version (3634 total pages: 494 relevant, 596 partially relevant, 34 not classified, and 2510 not relevant). The high number of relevant documents (contrary to real settings where the number of relevant documents could be much less with respect to the size of the collection) is explained by the fact that we assume that the documents not judged are not relevant.

5.2 Strategies and experiment

We used CREC to compare four automatic query construction strategies. All of them work on term lists extracted automatically from the <description> field of the context descriptor: for instance, the context descriptor in Section 5.1 is seen by the strategies as “user just landed london heathrow international airport looking flight timetable timetable connections london lunch time” (conversely, the human relevance judge uses the whole context descriptor).

The strategies are based on the combination of two main indexes: tf.idf and geoterms (i.e., terms that refer to geographical information — a term is a geoterm if its Wikipedia page contains geographical coordinates). We chose tf.idf as it is a classical and largely used IR technique, and geoterms because location is probably the contextual dimension that is more informative of user’s current context. In particular the four strategies are:

- **tf.idf**: all the significant terms in a context descriptor are taken into account for query formulation based on the evaluation of their tf.idf value, in descending order.

- **inverse tf.idf**: the previous strategy with terms taken in ascending order. This strategy represents a lower bound to use in the comparisons.
geoterms + tf.idf: in the query first the geoterms are exploited and then the remaining terms ordered by their tf.idf value, in descending order.

tf.idf + geoterms: like the previous strategy, but introducing the first two terms with the highest tf.idf, then the geographical related terms, and then again the following tf.idf ordered terms.

For each strategy and context descriptor, 10 queries of different lengths (from 1 to 10 terms) are automatically formulated, incrementally selecting the first 10 terms of the ranked lists. Thus, query construction is incremental: once a term is in a query, it will remain in longer queries as well. We also use, as an upper reference strategy, the manual approach, where a mobile user directly chooses terms and defines her query (we used the queries generated to build the first version of the document collection).

We measure strategies effectiveness by means of a standard IR metric, nDCG@10 (normalized Discounted Cumulative Gain). Using a graded relevance scale of documents in a search engine result set, nDCG measures the usefulness of a document based on its position in the result list. This measure has been chosen as it emphasizes quality at the top of the ranked list. Moreover, nDCG@10 takes into consideration only the first 10 retrieved items, which is reasonable for CAB, since the user is unlikely to scroll long lists of retrieved items.

5.3 Results

Fig. 5(a) compares the four strategies, and the manual one, showing their effectiveness (nDCG@10, on the Y axis) averaged on all 10 contexts, for different query lengths (X axis). Apart from the manual one, the most effective strategy is the geo+tf.idf. Fig. 5(b) shows, besides geo+tf.idf average, also min, max, and variance. In this strategy, first all the geoterms are added to the query, then the other terms, ranked by decreasing tf.idf, follow. Further analysis of the data, not reported here, shows that the maximum performance is obtained when just one tf.idf term is added after all the geoterms (each context contains 1, 2, or 3 geoterms), then nDCG@10 decreases. The performance of long queries is very low.

All the proposed strategies have lower performances than the manual one (higher curve in Fig. 5(a)), therefore they can be improved. Moreover, the effec-
tiveness of manual strategy tends to improve with longer queries: one reason is that automatic strategies are constrained by the incremental query construction, whereas manual strategy is not.

As relevance judgments have been made by a unique judge, to verify that subjectivity is not an issue for our benchmark, we performed an additional experiment, involving two more judges. Measuring inter-judge agreement on a pool of retrieved pages, judgments on average agreed on 65% pages (which became 92% after a discussion between judges). New relevant pages retrieved by the two new judges were also added to the collection, and the evaluation performed again for geo+tf.idf. The dashed line in Fig. 5(a) shows the effectiveness of geo+tf.idf computed considering the new pages and judgments: it does not change significantly. Thus CREC seems reliable at least to a reasonable extent, at least for its aims.

5.4 Discussion

The CREC benchmark helped the development process giving good insights (e.g., in the best strategy, the best performance is reached when adding a single term after the “geoterm”) and underlining weak points (e.g., adding more and more terms in the query does not necessarily increase performance). Moreover, a lesson learned is that early stage evaluations using a benchmark, followed by user studies, is an effective methodology for evaluating systems like CAB. The benchmark does not substitute the user testing evaluation. Rather, several early stage benchmark experiments could provide more solid basis for subsequent (more focused) user testings.

The CAB is an extremely novel and highly interactive context-aware retrieval application, thus its evaluation puts us in a paradoxical situation. On the one hand, users and tasks seem needed, thus a user study based on some task analysis seems mandatory. On the other hand, user studies would be far too expensive at an early development stage, and there are no realistic users and tasks until the system is launched on a large scale, and if a “wrong” system is launched to start the wave and have some users, then the system will fail, and there will be no users at all. Pushing this line of reasoning to its extreme consequences, one might wonder if evaluation somehow hinders development in the IR field. Indeed, when a paradigm shift is going to happen, it is quite unlikely that current evaluation methodologies can cope with the new scenario: they will evaluate the revolutionary system on the basis of the current evaluation techniques, that could be not appropriate for such a system and could lead to negative results.

6 Conclusions and future work

In this paper we have presented the Context-Aware Browser, whose strong point is represented by its generality: it is not a domain dependent application, but a new generic approach to web contents perusal by means of mobile devices. The novelty of the proposed approach is twofold. First of all, we take into account the information provided by the surrounding environment in order to carry out a proactive search of web contents. Secondly, information is not only automatically pushed towards the mobile device and then filtered: rather, the
CAB approach involves a two stage retrieval plus filtering process, that minimizes bandwidth usage and privacy and security risks. As an overall outcome, the effort required on behalf of the user is minimized as well.

As future work, besides aiming at a full user evaluation in a controlled environment, we continue the already started process of integrating CAB into the social networks and Web 2.0 worlds, to hopefully allow mobile users to easily produce personalized and contextualized web contents, comments, evaluations, etc. In addition we aim also at comparing our work with other systems already presented (e.g. Ubiquitous Web [7]), in order to find the weak and the strong points of our model. Actually we are working to deploy the CAB in the real world scenarios described in Section 1 and Section 4 (airport, home-automation, museum), as well as in other scenarios (shopping mall, trade fair, etc.). These deployed version of the CAB will be the basis for the comparison we aim at.

Moreover, current models for context-awareness are too limited for a very general approach like the CAB, as they rely on rigid categorizations built upon ontologies and strict terminologies. Thus we are studying new models that exploit the social dynamics underlying the Web 2.0, with the aim of involving the whole community of users: we believe that the collaborative effort of a community can provide the right tool for a comprehensive definition, management and use of context, in an open architecture as the CAB.

Acknowledgements

The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M888, and the region Friuli Venezia Giulia. This research has been partially supported by MoBe Ltd. (www.mobe.it), an academic spin-off company specializing in software for mobile devices.

References


