IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Enhancing Search Engines Based On Context Awareness

Varadharajan S[1], Varalakshmi S[2], Vidhiya R[3], Ranjitha G[4]

[1] **Assistant Professor,Computer Science Department,Anna university**
Surya Group Of Institution,
Villupuram, Tamilnadu, India

[2,3&4] **UG Student,Computer Science Department,Anna university**
Surya Group Of Institution,
Villupuram, Tamilnadu, India

Abstract

Surfing is the regular task to be considered, when any information needs. To provide Satisfactory usability, Surfing must consider the nature of request and context. Being both Pervasive and Person-centric, System continuously capture information about the user Queries and their context. Context Awareness can enhance the Surfing experience by augmenting user queries with Context Information captured through a system, The authors unified Architecture Supports context Awareness in mobility. Describe various design approaches and the benefits of making Surfing more Context Aware.

1. Introduction

SURFING to the Internet has grown rapidly in recent years, leading key industry specialists to predict that it will soon be the most common method of Web access. Worldwide, the number of users connecting to the Internet from devices rather than desktop computers.

To keep up with this expansion, search engines — the primary gateway to the Internet for more than half of all users[1] — must adapt to environments. In particular, search engines must take into account system, which are pervasive and person-centric, continuously capturing user-related information. Such data can provide more meaningful search results by augmenting searches with real-world information related to users' profiles and behavioral patterns.
Information about users such as location, how they interact with the device, or what's occurring in the surrounding physical world is called *contextual data*. Augmenting user queries with context awareness can improve both a search engine's understanding of intent and a user's interaction with the search interface. of intent and a user's interaction with the search interface.

Our proposed model is continuously capture information about the user Queries. We developed an application architecture that supports context-aware surfing, and use real context data from the Reality Mining project.[5] Web browser continuously capture information about the user Queries , the Queries will be change automatically based on our time and location, according to user needs.

2. Context and Context-Aware Surfing

Context has many definitions and inherent concepts. Our context model uses various context variables to augment and enhance surfing.

Context Awareness

Since the 1960s, debate has been ongoing about how we define "context" in relation to computer systems. The term context, or context awareness, usually refers to a general class of systems that can sense a continuously changing physical environment and provide relevant services to users on this basis.[6] Based on this core definition, many authors[2,3,7,8] focus on different aspects of context awareness, such as its nature, how to model contextual data, and interactions between users and context.

Building on this work, we define context on the basis of three parameters:

1.1 Concept

Context is any relevant information that can help characterize an entity's situation when that entity is interacting with a user at a given point in time. In this user-centric world, an entity can be a person, device, software application, or almost anything else. Here, we focus primarily on a surfing application.

1.2 Acquisition

We obtain a view of the user's context by combining signals from a mix of sensors available in mobile devices: GPS, camera, accelerometer, compass, touch, and so on. Our context model analyzes and groups the data into context variables.

1.3 Usage

We use context information to enrich the applications that users interact with. For our surfing scenario, we consider all sensors available in a system as potentially informative.

3. Context Modeling

Several common variables accessible in system can provide relevant information about the context in which a user performs a surfing (these can also affect other system interactions in general):

- signal coverage strength;
- active connectivity methods;
- environmental information, such as light;
- current location;
- activity time and time frame;
- internal device information, such as power; and
- the device's motion and angle.

These variables provide different and complementary views on user interactions. If, for example, the network coverage signal is low, a user could lose Internet connectivity. Depending on the connection type (Wi-Fi, 3G, and so on), the device's speed might vary. Other active connection methods such as Bluetooth or applications running in the background can affect power consumption. If the battery level is low, then the system might turn off, causing the user to lose his or her surfing history.

We can typically retrieve context data using APIs available in programming SDKs. Using common Java API sensor specifications, we list a number of context signals, which sensor captures these signals, and the context variables they affect:

• ***Movement*** **(accelerometer)** shows the device's inclination and motion.
• ***Location*** **(GPS)** shows location and proximity to places.
• ***Audio*** **(microphone)** captures auditory environmental information such as noise level, music, and voice commands.
• ***Light*** **(camera)** captures visual environmental information such as light intensity or spectrum.
• ***Battery level*** **(battery charge)** shows information about the device's remaining battery power.
• ***Time*** **(clock)** shows information related to the time frame and time gauge.
• ***Connection*** **(communications)** shows active connectivity methods and the network to which they correspond.

Next, we explain how we can improve surfing by adding contextual information.

4 . Including Context in Surfing

Next examine surfing and look at how to enhance them with contextual information. It also define our conceptual architecture for context awareness.

4.1 Surfing

Surfing is the practice of querying a search engine from Internet-connected devices (for example, modem), typically via search engine web pages displayed in a browser, or through dedicated surfing applications often native to the device. Most major players largely adapt their Web interfaces and APIs to meet system characteristics.[11] In the academic domain.

However, researchers have proposed specialized search engine designs in three main areas:

• ***Interaction.*** New methodologies such as Yahoo's oneSearch enable processing queries and categorizing results.[9]

• ***User interface*****.** One example, the Find browser, categorizes search results in the display.[9]

• ***Context usage*****.** Some applications use context information such as location, time, and social data together with query text, as in Yahoo one Search and Google's search application.[11]

]In a search data flow, users input a query using text (keyboard) or voice (microphone). The mobile user interface (an application or browser) captures this query and passes it to the search API, which is key to the search data flow because it defines the supported I/O parameters for a search query and its response.

Apart from the query string, most commercial search engines and browser technology APIs, such as the geolocation API included in HTML [5], support certain parameters.

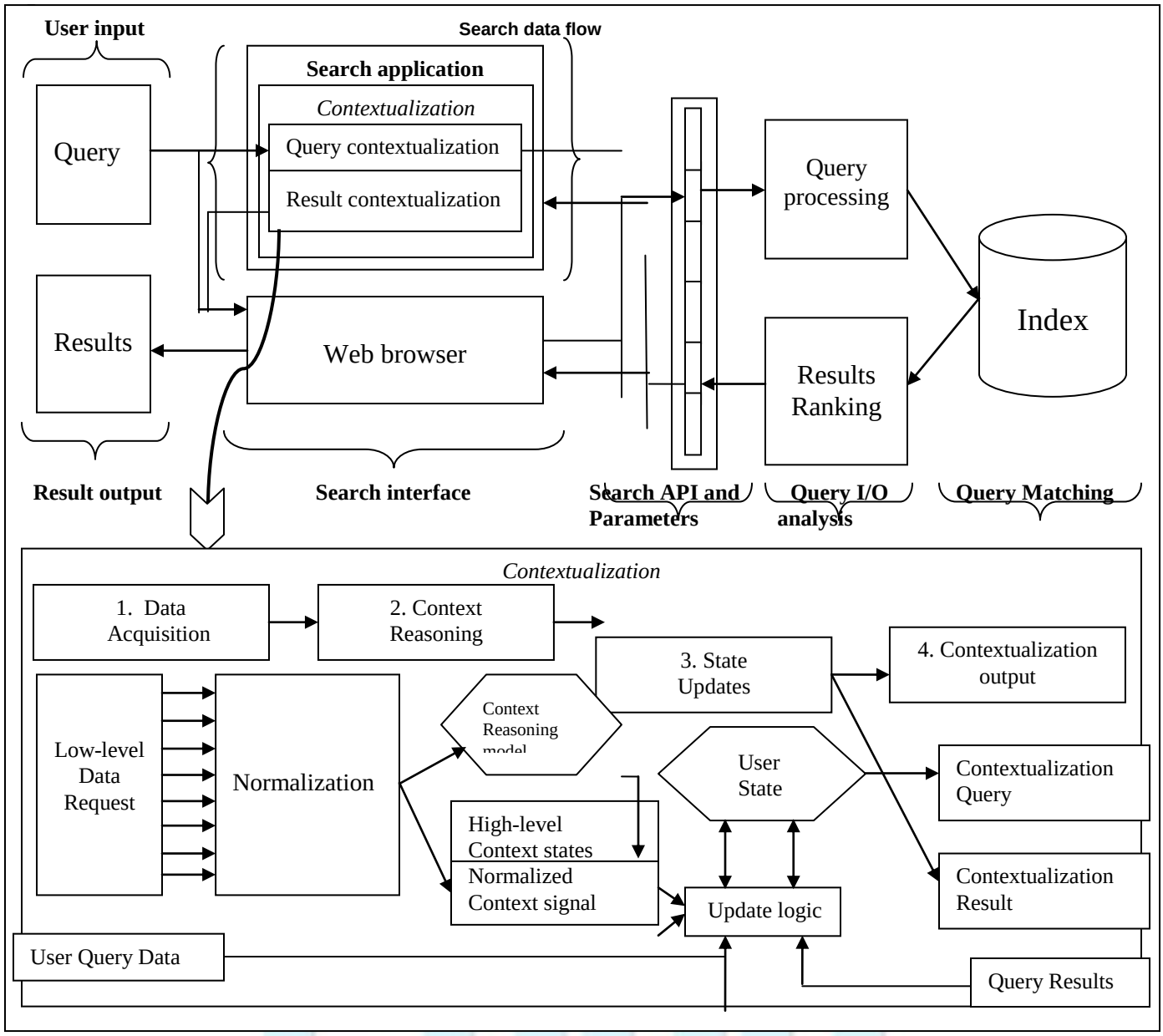

Figure 1. Surfing and Context-embedding module.

However, additional work is needed so that we can add more contextual variables to the surfing workflow.Richer contextual data could improve a mobile search application's utility by, for example, enabling it to adapt user interface and interaction paradigms..

It can define which kinds of documents they should return. Functionality includes determining the document language and how many documents to return, filtering the results for adult content, searching vertically (for example, multimedia, news,or local data), and determining whether to perform certain kinds of query processing.

4.2 Embedding Context in Surfing

We propose a generic architecture that processes context signals and makes them available to downstream applications such as search engines through a contextualization module. This module comprises two components: *query contextualization* and *result contextualization*. Figure 1 shows a conceptual diagram of a search application with our contextualization design. The contextualization module transforms the query text and context signals into the appropriate search API parameters. Once a user has queried the search engine via its API, the engine performs various internal query-processing steps —such as spelling alterations, synonym expansion, and query classification — before sending the query to its index to retrieve relevant documents.

The search engine then ranks the recalled documents by relevance to the query and returns them to the search application. The contextualization module processes context signals in four stages: stages 1 and 2 are application-independent and solely concern acquiring context signals and modeling higher level context states; stages 3 and 4 use application specific logic and I/O routines.

4.2.1.Data acquisition: The module samples the different hardware sensors and applies signal processing techniques to produce a context vector of clean signals with well-defined ranges. Typical operations at this stage include signal smoothing, scaling, thresholding, and applying stochastic filters such as Kalman filters. Raw context sensor signals are the inputs, whereas outputs are the normalized and cleaned context vectors.

4.2.2 Context reasoning: We apply a context model to infer context states from the underlying signal vector. The states we use are application dependent and need not be mutually exclusive. We can use both the activity contexts a user is in as well as various independent device states such as "low battery" and location. Context vectors are the inputs in this stage, while outputs are high-level context states.

4.2.3. State updates: After gaining a detailed and robust understanding of the context, the application can update its internal state. Aside from context, it can also use explicit user inputs and other application-state variables. With search, we can combine a user's query with the context states to produce a contextualized search engine API query. In this stage, inputs are context states and user inputs, while output is the application state.

4.2.4. Contextualized output: Based on its updated state, an application can either produce contextualized outputs that are user-facing, as in the contextualized presentation of search results, or call external services in a contextualized fashion, as with augmented search engine API calls. The application state is the input in this stage, while outputs include user directed interface actions and communication with other processes and applications.

We implemented our contextualization module using real-world data from the Reality Mining project. Based on this data, we can infer five main context states:

- ***leisure*** (entertainment activity);
- ***working*** (university-related work);
- ***errand*** (running an errand);
- ***commuting*** (traveling to or from work); and
- ***sleeping*** (resting).

Students (users) can be in any of these states, which represent a combination of activities done with the system at a particular location and time.

5. Proposed Algorithm

Build Context Based Query

For each request r **do**

$$t_r = time(r)$$

$$s_r(t) = Visiter\ Loc(r)$$

$$r_{st}(t) = OPT(\prod_{i=1}^{n} status_i(r,t))$$

$$cs_r(t) = \bigcap_{i=0,..,n-1}(t_r, s_r(t), r_{st}(t), r)$$

$$cq_r(t) = \mathbb{Q}(cs_r(t))$$

For every context query cq **do**

$$\coprod_{i=1,\dots,n} cr_i(t) = RET(cq_r(t))$$

End for

$$v_r(t) = \{cr_1(t), cr_2(t), \dots\dots, cr_n(t)\}$$

End for

Where,

t_r = time of the request

$s_r(t)$ = source of the request

$r_{st}(t)$ = request status (sleeping, working)

$cs_r(t)$ = context status of the request

$cq_r(t)$ = context query

$cr_i(t)$ = i^{th} context response from the server for the query

$v_r(t)$ = value of the response

We have to built Context based query, initially client request to server for context based information, then check time is t_r. The source of the visitor location , checking the status working or sleeping mode. The working mode is all user query information will be applicable and sleeping mode is not applicable.

After checking the status the query will be change in optimal according to user needs so our search engine filtering the data based on timing. Our search engine mainly reduce a server hit. Finally we retrieve optimal context response.

6. Context Model

The context model we describe next is based on the generic context architecture defined earlier, but uses only variables and knowledge from the Reality Mining project.

One challenge we face is that sensor data is highly heterogeneous, which makes it hard to form robust context inferences. We thus model the context signals as a unified context vector **V**(*t*), which is parameterized by time *t* (see Equation 1). All context variables are synchronized with respect to time and bucketed into discrete time slices; the bucket duration determines how much context data is available for each context reasoning step.

During real-time use, the bucket size also determines how quickly an application can respond to a change in context variables.

$$\mathbf{V}(t) = \{CR1(t), CR2(t),.. .CRn(t)\} \quad (1)$$

where *CS* is the context signal value for time slice *t*. We calculate some context variables as the number of times we observed the event associated with a signal in the time slice, which we measured as day of the week/time — that is, Monday through Sunday, morning, afternoon, or evening.

Other context variables come from the application logs; we calculate these as the percentage of time the signal is on over the time slice.

These variables include

- ***communication*** (the duration of the call, texts sent, calls made, or missed calls);
- ***Internet*** (whether an Internet connection is active);
- ***location*** (whether the cell ID is active); and
- ***application*** (which phone applications the student was using).

We use the context vector as the input to derive our higher-level context states. Our model design has two steps in which we define a set of rules.

During **step 1**, we use the location cell ID to infer a location group, such as "Home" or "Work." In **step 2**, we use the derived location signal together with the remaining signals to infer the higher-level context states defined earlier.

For this example, we use some assumptions derived from the student survey given during the Reality Mining project.[5]

Step 1: Location modeling**:** We know that students are at a specific location during a given time frame. So, when capturing a cell ID, if its time stamp is in the time frame, we match it to a location and store it in a table, which we use in our rules to define location. We specify the cell ID and location table values on the left side of the rule and the location group on the right. We can build this location table for new users by prompting them to input their location or by inferring it based on recurring patterns and the time of day.

We defined the following location derivation rules:

If (*tower networkID in* [*Home cellID range*]) →
Location = HLR

If (*tower networkID in* [*Work cellID range*]) →
Location = Work

If (*tower networkID in* [*Elsewhere cellID range*] →
Location = VLR

Step 2: context-state modeling**:** In this step, we define rules for identifying context states (see Figure 2). Some context variables become more useful when they're combined — for example, time with location, which can indicate different activities (sleeping, leisure, and so on). The rule to the left of the arrow specifies all the context signals and the values they take; text to the right specifies the associated context states.

7. Applying Context

Given the set of context states and the set of normalized context signals, we contextualize the Web search application (stages 3 and 4 in Figure 1). Following our methodology, we distinguish between contextualizing the search API query and displaying the search results.

As with our prior prototypical definition of a context model, we can define an appropriate state update model for our search application in many ways. For this article's purposes, we mention several heuristic rules to illustrate the benefits that the available context signals have in different search scenarios.

Although rules such as these are practical and effective in some cases, a more automatic and learned approach might be more suitable for other applications; we plan to explore this idea further in future work. The search engine then ranks the recalled documents by relevance to the query and returns them to the search application.

Figure 3 lists several example scenarios, together with the state updates and example search engine outputs produced for contextualized and non-contextualized versions of the query. We used Microsoft's Bing search engine for all queries.

Often, the appropriate state updates we must perform can benefit from a deeper understanding of individual user characteristics.

Query	Context Signal	State Update	Example
Hotel	**State = = abroad Location = = London**	**Add Location string to query words "hotel" → "hotel London"**	**Hotel-Wikipedia the free encyclopedia → en.Wikipedia.org/wiki/hotel** **London Hotel | save up to 55% Search discount price….. www.hotels-london.co.uk**
Restaurant	**State = = errand Location = = London Time = = 9pm-9am**	**Based on query time, activity, and location add extra qualifiers to query keywords "restaurant"→ "restaurant London lunch"**	**Restaurant-Wikipedia the free → en.Wikipedia.org/wiki/ Restaurant** **London Restaurant | Save up to 55% Search discount price on over 600 handpicked.**
Hospital	**State = = Tamilnadu Location = = Chennai**	**Add Location string to query words "hospital" → "hospital chennai"**	**Hospital-Wikipedia the free encyclopedia en.Wikipedia.org/wiki/ Hospital**
	State = = Working Location = = Chennai Time = = 9pm-9am	**Based on query time, activity, and location add extra qualifiers to query keywords " hospital"**	**Hospital-Wikipedia the free encyclopedia It Shows only applicable hospital**

Figure2. Example Scenarios.

((*location*[*Home*] = 1) AND (*dateOfWeek*[*Sunday*] = 0) AND (*dayTimeSlice* [*Night*] = 1)) → ***Sleeping***
(*location*[*Work or Home*] = 1) AND *dayTimeSlice* [*Night*] = 0) AND (*Phone*[*Active*] = 1) AND (*Application*[*PhoneUsage*] = 1) AND (*communication*[*out*] = 1) → ***Leisure***
(*location*[*Work*] = 1) AND (*dayTimeSlice*[*Night*] = 0) AND (*Phone*[*Active*] = 0) AND (*Application*[*PhoneUsage*] = 1) AND (*communication*[*Missed*] = 1) → ***Working***
(*location*[*Work or Home*] = 1) AND (*dayTimeSlice*[*Night*] = 0) AND (*Phone*[*Charge*] = 0) AND (*communication*[*Out*] = 0) → ***Working***
(*location*[*Elsewhere*] = 1) AND (*dayTimeSlice* [*Morning*] = 1 OR *dayTimeSlice*[*Elsewhere*] = 1) AND (*Phone*[*Active*] = 1) AND (*Application*[*ANY*] = 1) AND (*communication*[*out*] = 1) → ***Commuting***

For example, Karen Church and her colleagues highlight some diverse user characteristics exhibited during their diary study[13] related to mobile search behavior. Ideally, we should consider individual query habits and history when contextualizing a search application, especially when targeting a platform as user-centric as a system. In this sense, we can view context information as the input to a more general inference engine (as a specific case of stage 3 in our design, which also models user preferences and profiles). A representative example of such an inference engine that models user profiles (although not other contextual information) for query reformulation purposes is available elsewhere.[14]

8. Conclusion

We've presented a context-aware system that can be used with search engines to enhance queries. we're investigating machine learning approaches to replace the heuristic rules presented in this article. the future, enhancing search engines themselves must become more context-aware. Besides user location and time, and other context information.

REFERENCES

[1] J.L. Gómez-Barroso et al., *Prospects of Mobile Search*, tech. report EUR 24148 EN, Inst. for Prospective Technological Studies (IPTS), European Commission, 2010.

[2] J. Coutaz and J.L. Crowley, "Context Is Key," *Comm. ACM*, vol. 48, no. 3, 2005, pp. 49–53.

[3] H.W. Gellersen, A. Schmidt, and M. Beigl, "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts," *Mobile Networks and Applications*, vol. 7, no. 5, 2002, pp. 341–351.

[4] P. Korpipää and J. Mäntyjärvi, "An Ontology for Mobile Device Sensor-Based Context Awareness," *Proc. 4th Int'l and Interdisciplinary Conf. Modeling and Using Context* (CONTEXT 03), Springer, 2003, pp. 451–458.

[5] N. Eagle, A. Pentland, and D. Lazer, "Inferring Friendship Network Structure by Using Mobile Phone Data," *Proc. Nat'l Academy of Sciences*, vol. 106, no. 36, 2009, pp. 15274–15278.

[6] A.K. Dey, "Understanding and Using Context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, Springer, 2001, pp. 4–7.

[7] L. Barkhuus and A. Dey, "Is Context-Aware Computing Taking Control Away from the User? Three Levels of Interactivity Examined," *Proc. 5th Int'l Conf. Ubiquitous Computing* (Ubicomp 03), vol. 2864, 2003, pp. 149–156

[8] O. Lassila and D. Khushraj, "Contextualizing Applications via Semantic Middleware," *Proc. 2nd Ann. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services* (MobiQuitous 05), IEEE Press, 2005, pp. 183–191.

[9] Yi, F. Maghoul, and J. Pede, "Deciphering Mobile Search Patterns: A Study of Yahoo! Mobile Search Queries," *Proc. 17th Int'l World Wide Web Conf. — Refereed Track: Mobility* (WWW 08), ACM Press, 2008, pp. 257–266.

[10] T. Heimonen and M. Käki, "Mobile Findex — Supporting Mobile Web Search," *Proc. Int'l Conf. Human Computer Interaction with Mobile Devices and Services*, ACM Press, 2007, pp. 397–404.

[11] K. Church and B. Smith, "Who, What, Where and When: A New Approach to Mobile Search," *Proc. 13th Int'l Conf. Intelligent User Interfaces* (IUI 08), ACM Press, 2008, pp. 309–312.

[12] H. Daume and E. Brill, "Web Search Intent Induction via Automatic Query Reformulation," *Proc. Human Language Technology Conf. North Am. Chapter of the Assoc. for Computational Linguistics* (NAACL-Short 04), 2004, pp. 49–52.

[13] K. Church and B. Smyth, "Understanding the Intent Behind Mobile Information Needs," *Proc. 14th Int'l Conf. Intelligent User Interfaces* (IUI 09), ACM Press, 2009, pp. 247–256.

[14] V.C. Storey, V. Sugumaran, and A. Burton-Jones, "The Role of User Profiles in Context-Aware Query Processing for the Semantic Web," *Natural Language Processing and Information Systems*, LNCS, Springer, 2004, pp. 45–62; doi:10.1007/978-3-540-27779-8_5.