

# Artificial Intelligence Approach of Context-Awareness Architecture for Mobile Computing

Feng Gui, Nuannuan Zong and Malek Adjouadi

Florida International University, Department of Electrical & Computer Engineering

10555 W Flagler Street, Miami, FL, 33174, USA

[gui\\_feng@yahoo.com](mailto:gui_feng@yahoo.com), [flzn527@yahoo.com](mailto:flzn527@yahoo.com), [adjouadi@fiu.com](mailto:adjouadi@fiu.com)

## Abstract

*Wireless network carriers are expanding their data services to combat the shrinking average revenue per user in the face of challenges from competitive markets and emerging technologies such as VOIP. Context-aware applications, built on the notions of human computer interfaces and interaction, extract user data, analyze user patterns, and infer user intentions in a given environment. The objective of this study is to design an intelligent architecture that is based on an integrated platform that provides a context-aware solution for mobile users. The major components of the architecture are smart proxy on the user devices and context-aware search engine in the carrier's network. Context weights and profiles are automatically generated to reflect the mobile user's contextual change. The smart proxy computes context weights by utilizing a dynamic neural network based algorithm and adapts to user situation. At the server side, the context-aware search engine returns query results based on the context profiles sent by client proxy. As such, a systematic approach can be used to handle context awareness for mobile users.*

**Key words:** context-awareness, data cache, context cache weight, context profiles, virtual frame, reference frame, and search engine.

## 1. Introduction

The pervasive nature of portable wireless devices has played an increasingly important role in our daily lives. Wireless communication carriers start to roll out value-added services such as local map and mobile wallet so as to improve mobile user's experience. Context awareness becomes one of the focused research areas in ubiquitous computing [1]. Context awareness in general refers to analysis and interpretation of the given situation based on context entities like mobile device configuration, mobile user characteristics, user activities, time of the day, and surrounding environment. In the past most carriers provided primitive context-aware services through the network which carried out all the computation. This outcome is mainly due to the hardware limitation of the cellular phones. As cellular phones are becoming more sophisticated, the client devices could take more tasks such as collecting user data, analyzing user situation, and delivering context-aware content to mobile users. To attain such specific aims a new

architecture that uses the client and server paradigm is proposed.

The context-aware computing at client site includes the acquisition of context entities, updating the context weights, compiling the context profile, and delivering the context content. On-board sensors and user applications [2] are the practical means to collect user's preferences. The trend in the industry clearly shows that manufacturers continually integrate new functions into a small hand-held cellular phone [3]. Nowadays, applications can easily log and collect user preferences, characteristics, activities, and profiles as mobile users make use of them on a daily basis. Clearly, mobile devices increasingly expand the role of extracting and analyzing user context information.

Because of the variable-and-Instable nature of user context, the difficulty in context awareness lies in the extraction of useful feature/context from changeable user situations. It is therefore important to design new algorithms at the client side to undertake the preliminary context analysis of user situation. The algorithms or client proxy should relieve the networks (i.e., servers) of the computing burden and reduce the need for uplink bandwidth [4]. The purposes of the user context profiles managed by the client proxy are 1) filter out irrelevant user information and describe the user's current context; 2) update server of current user context; 3) assist in predicting user intention at both server and client side. Upon receiving the context profiles from the client devices, the server could provide context-aware content/services based on the context profiles.

In the past, Artificial Neural Networks (ANN) based algorithms have elicited models in context-aware mobile computing application [5]. A few models have been proposed as exemplified by Clarkson and Pentland who carried out experiments with unsupervised clustering method, Hidden Markov Models (HMMs), to verify the audio and video contextual events [6]. Researchers at Nokia research center utilized a lattice-based structure, Symbol Clustering Map (SCM), to update the weights which are provided for context symbols. The updated weights quantify the context symbols so as to recognize the particular context [7]. In addition, other methods such as Self-Organizing Map [8] and Markov Models [9] are applied or combined to overcome the mentioned difficulties in assessing context.

The new architecture as proposed exploits dynamic neural network based on a learning of a single layer net. Weights in context profile at current epoch can be updated with

respect to the previous epoch. The merit of a learning rule is that it is simple and easy to implement.

As portable devices become more capable of delivering multimedia content, mobile users are likely to increase their usages of these devices for digital data such as emails, video, and m-commerce, to name a few. The surrounding environment may change from time to time due to the user's mobility. One of the challenges in the information retrieval for mobile users is in establishing context sensitive retrieval process. Furthermore, the mobile search for information on portable devices should match the web searches. Currently, most web search algorithms we experience are limited solutions for context sensitive retrieval and mobile search. This is so because most existing algorithms do not take into account mobile user context inputs such as surrounding environment. For this reason, a context-aware search scheme at server side (i.e., carrier's network) is devised to provide content based on the compiled context profiles.

## 2. Architecture Design

The traditional client/server paradigm fits well into the context-awareness applications. The carrier's network is the server which provides data and voice services to subscribers. Mobile devices request services from the network. What makes our approach different to other models is that the mobile devices play an extensive role of collecting, analyzing, and extracting context entities. Context profiles compiled at client side greatly reduce computing burden at network/server side.

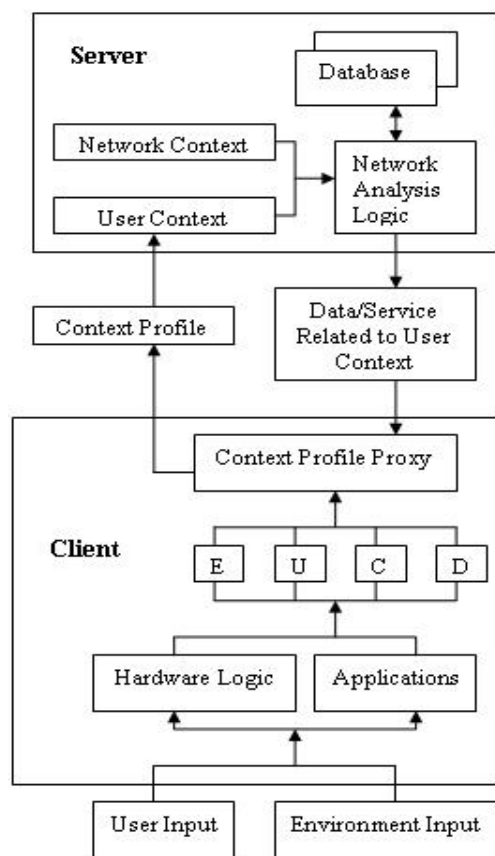


Figure 1. System architecture

E: environment profile      U: user profile  
C: cache profile              D: device profile

Figure 1 illustrates the structure of the paradigm for the client/server model. The user inputs (voice or data) and surrounding environment inputs (temperature, position, altitude, etc.) are collected by the hardware logic or the applications such as the operating system. The context-aware proxy further inspects inputs and extracts context entities from the inputs. Finally, the proxy compiles the context profile and sends it to the network/server. The network/server learns the user situation using the context profile in addition to the data collected at the network. Server application provides services and data to client based on the learning of the user situation.

The context profiles such as environment profile and user profile are explained in detail next.

## 3. Client Proxy

The client proxy that is running in the user devices monitors and collects mobile user information through sensors and applications. In addition, the proxy further compiles the context profiles which adapt to and reflect mobile user's changing situation. Cache schemes, virtual frame, and reference frame are thus proposed to improve the efficiency of the proxy's operation at the client side.

### A. Context Profiles

The context profile is a collection of context entities extracted from the on-board sensors, client applications, user activities, and so on. Ideally, context profiles should be a good reference of subscriber's current situation from which the carrier's network could derive subscribers' context or intention. Thus, the proxy on mobile devices frequently updates context profiles and uploads them to network for reference if necessary. Due to the limited uplink bandwidth [10], the context profiles should be concise. As user changes activities/context, context entities are added to or dropped from the profiles based on the algorithms we specify in the later section. Whenever addition or deletion of context entities occurs, the proxy notifies the network of the changes. If the user's situation remains the same or change a little, then only updated weights in the context profiles are sent to the network. There are four context profiles managed by the client proxy;

1. *User profile* describes the user's characteristics, preferences, activities, emotions, and so on. For example, user's age, gender, height, weight, health conditions, past incidents, and planned events can be sorted into this category.
2. *Device profile* describes the configuration and functions of the hardware. In addition, this profile includes the software applications and user interfaces. Examples of context entities in this profile are processor, memory, display type, device size, and operating system.
3. *Environment profile* describes surrounding area of the mobile user. This profile stores user location, weather, noise level, temperature, etc.
4. *Data profile* caches user's data in the local memory.

Because the device hardware configuration is quite fixed with respect to user activities, the weights for entities in device profiles are relatively lower and remain unchanged. They are included mainly for server to deliver content in a way that fits best for the device.

### B. Cache Scheme for Client Proxy

Least Recently Used (LRU) replacement algorithm [11], due to its simplicity, is widely adopted to manage the data cache in the traditional computing such as database system, file systems, or CPU design. The frequency-based replacement algorithm [12] factors out the locality from the reference count. Reference count remains 0 until the data block ages out of the “new” sector. The frequency-based replacement algorithm is effective to filter out bursts of reference counts. These data cache schemes, somehow, are designed for wired systems. Furthermore, these algorithms do not take mobile user’s context into account. We propose a new data cache scheme based on the user context profiles, virtual frame, and reference frame simultaneously.

When cache misses happen, the mobile devices do not have the information mobile users require. The downloaded data from the server must be cached in the local memory. With the traditional data cache scheme, the data is stored in the unit of memory block. In our cache scheme, the client proxy examines the data and extracts the incoming data into individual context entity based on context profiles defined early. The recognized context information is stored specifically in a context cache. Each cached context entity is coupled with a weight pointer. The weight pointer points to their historical values recorded in the virtual frames. Figure 2 shows the structure of the context cache. Each context entity,  $C_i$ , has its own weight pointer,  $W_i$ , which points to the history value in virtual frames recorded at different times.

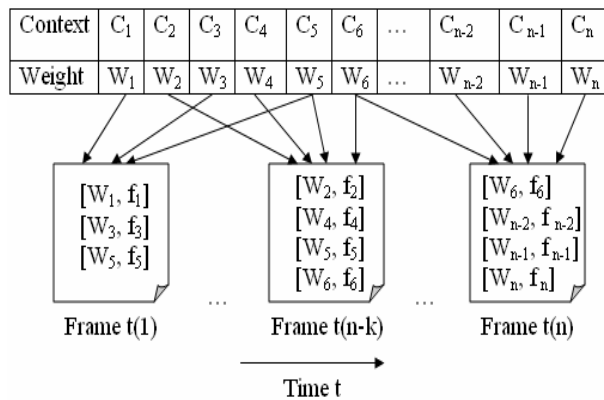


Figure 2. Structure of the Virtual frame

In this paper we introduce the concept of virtual frame. As most of us know, movies consist of individual frames that are played at a constant rate, for instance 24 frames per seconds. Each frame captures the physical scene of the context that leads to the understanding of the situation at the particular moment. Likewise, the virtual frame expresses the meanings of the mobile user’s context by featuring weights that are most significant at a particular moment. In other words, the virtual frame does not store any graphical components but weights of the cached data that captures the context of the mobile user at specific time frames. Figure 2 illustrates this concept. For example, virtual frame  $f_1$  consists of weights that are recorded to capture the context of the mobile user at time  $t_1$ . Likewise, virtual frame  $f_n$  contains all meaningful weights that describe the mobile user characteristics and environment at time  $t_n$ . Time plays a great role in predicting the mobile user’s context or intention. Intuitively, the user’s context is likely to remain

unchanged or consistent over a short time interval in most scenarios.

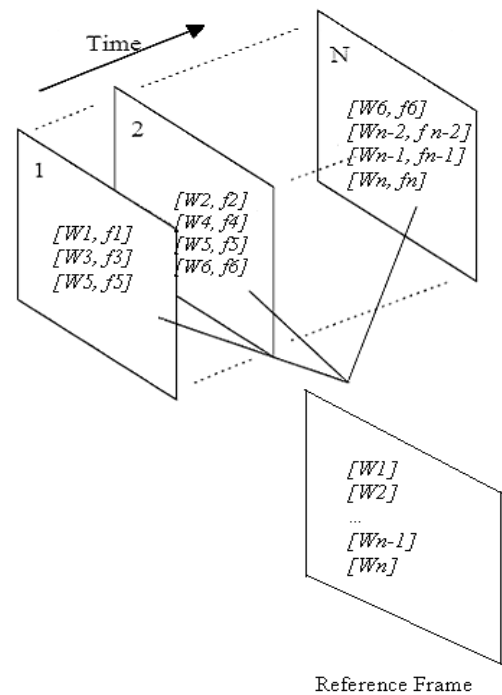


Figure 3. Structure of the reference frame

Having explained the virtual frame, we further introduce the notion of reference frame. The reference frame only packages data cache weights as virtual frames. However, reference frame is not quite like a virtual frame in that the reference frame is a moving average of the virtual frames. The reference frame averages the most-recent  $N$  virtual frames depending on the configuration of individual mobile devices as shown in Figure 3. Weights included in the reference frame are calculated as:

$$W_k = \frac{1}{N} \sum_{f=1}^N W_{fk} \quad (1)$$

where  $f$  indicates virtual frame number as  $f = 1, 2, 3, 4, \dots, N$ ; while  $k$  refers to all individual weights contained in  $N$  virtual frames as  $k = 1, 2, 3, 4, \dots$

It is likely that a specific weight,  $W_k$ , might not be captured in every virtual frame,  $f_i$ , as mobile user’s context changes over the time. Clearly data cache weights that appear most frequently in virtual frames would be the dominating weights appearing in the reference frame. Thus, the reference frame is a much more balanced frame that describes the current mobile user’s context with respect to both current and past. Definitely, there would be a trade off between the history context and the current context. As  $N$  increases, more virtual frames in the past would be included for consideration. So the latest virtual frame has less bearing on the reference frame. Therefore, the reference frame derived from a large set of virtual frames would account for more consistent context for mobile users. On the contrary, if  $N$  decreases, then the most recent virtual frame weighs more on the reference frame which better fits mobile users changing their context dramatically in a short period of time.

Obviously, it is desirable to make  $N$  adaptive to the context change. If the client proxy figures out that the mobile user’s context remains relatively consistent mea

no new context entities are brought into the most recent virtual frame, then  $N$  is incremented up to the total virtual frames in memory as;

$$N = N_{previous} + 1 \quad (2)$$

where  $N_{previous}$  refers to the total number of virtual frames used to derive the reference frame in the last iteration. By increasing  $N$ , the reference frame would take more history virtual frames into consideration if context changes were little.

If the client proxy detects that the most recent virtual frame will bring in new weights that do not exist in the current reference frame, then  $N$  changes according to the following equation:

$$N = \left\lceil N_{previous} \left[ 1 - \frac{k(w_{new})}{k(w_{reference}) + k(w_{new})} \right] \right\rceil_{ceiling} \quad (3)$$

where  $k(w_{new})$  counts new weights that do not exist in the last reference frame;  $k(w_{reference})$  sums the number of total weights recorded in the last reference frame. We round  $N$  by taking the ceiling value. As  $N$  decreases, the last virtual frame could weigh more in the new reference frame. Thus, the cache system adapts much faster to the mobile user's context which changes dramatically over a short time.

When mobile user queries information that is already stored in the local memory, it is called a cache hit. The client proxy further inspects to see if this is a context-awareness hit [4], meaning that one or more context entities stored in the context cache is(are) required by the user again. In a context-awareness hit, the client proxy would increment the reference count accordingly, generate a new virtual frame and update the reference frame. Similarly, if the client proxy finds that it is a context-awareness miss, then new context entities and their respective weights would be stored in the context cache. Sequentially, these new weights would be reflected in a virtual frame and the updated reference frame.

If the client proxy detects that the local memory is full, then cache prune must be carried out. With frequency-based replacement, the cache block with the least reference count could be chosen for deletion. LRU simply replaces the least recently used memory blocks. In our approach, we pick the virtual frame as the candidate for cache replacement. The client proxy calculates the context distance for each virtual frame. The context distance,  $d_i$ , for virtual frame,  $f_i$ , is calculated as:

$$d_i = \sum W_{reference,p} + \sum |W_{reference,k} - W_{i,k}| + \sum W_{i,q} \quad (4)$$

where  $\sum W_{reference,p}$  sums all the weights that exist in reference but not in the virtual frame  $f_i$ ;  $\sum |W_{reference,k} - W_{i,k}|$  totals the difference of weights contained in both virtual frame, and reference frame;  $\sum W_{i,q}$  calculates the total weights that exist in the virtual frame, but not in the reference frame.

Context distance should indicate how far away a particular virtual frame is to the current reference frame. If  $d_i$  increases, then we say that the context reflected by virtual frame  $f_i$  at time  $i$  has shifted away from the user's current context. On the contrary, a smaller value in  $d_i$  signals that

virtual frame  $f_i$  is still a close assessment of the current user context. The virtual frame that has the largest context distance would thus be the first candidate for replacement. Next the weights stored in the candidate frame but not in the reference frame should be deleted from the context cache, because weights in the reference frame are within the current user context. After deletion of the virtual frame with the largest context distance, if the mobile device is still low in memory, then same logic would apply to the virtual frame with the second largest context distance.

### C. Dynamic Artificial Neural Network

The learning rule is based on the following general equation:

$$\Delta W_{ij}(t) = x_i(t)y_j(t) \quad (5)$$

In our approach, we define  $x_i(t)$  as a function of the context changes  $\Delta C$  and frequency count  $\Delta F$  of the context entities in the context cache managed by the client proxy between the current epoch and the previous epoch. In addition,  $y_j(t)$  is defined as the weight values,  $W_{reference}$ , in the current reference frame.

Thus, the derived equation of weight change for our approach at epoch  $t$  follows:

$$\Delta W_k(t) = |\Delta C(t) + \mu \Delta F(t)|_k W_{k,reference}(t) \quad (6)$$

where  $\Delta C(t)$  is calculated as  $\Delta C(t) = \frac{|C(t) - C(t-1)|}{C(t-1)}$ .  $C(t)$

is the current context value, and  $C(t-1)$  is the previous context value. If  $\Delta C(t)$  is zero, then  $\Delta C(t) = -|C(t) - C_{average}|$ .  $C_{average}$  is the mean of  $C(1)$  through  $C(t-1)$ .  $\Delta F(t)$  is the difference in frequency counter for context entity between the virtual frame at current epoch and the virtual frame from previous epoch,  $\Delta F(t) = F(t) - F(t-1)$ .  $\mu$  is the coefficient that is machine specific.

Finally, the weights for context entity  $k$  at epoch  $t$  are updated as

$$W_k(t) = W_k(t-1) + \Delta W_k(t) \quad (7)$$

If the virtual frame brings in new context weights that do not exist in the current reference frame, then the initial value of  $W_k(t)$  is set to 1.

## 4. Context-aware search engine

As the information exponentially increases on the World Wide Web and beyond, people rely more on the search engines to find the content of the digital age. The wide spread of mobile portable devices expedites the trend for mobile devices to gradually overtake the wired computers to provide users access to digital information. However, most search engines existing today provide query results based on the user string inputs and popular common queries. Search techniques evolve from simple string match algorithms [13] to sophisticated search on multimedia content such as image and video [14]. Major search engines from Google and Yahoo attempt to increase local content in the query results in a hope that users are provided more information in a local

context. In this paper we propose a new algorithm that takes into account user context profiles compiled at the client site.

### A. Search query expansion

The context profiles compiled by proxy at client devices reflect the hardware configuration, user applications, usage history, and derived user situation. The majority of current search engines only take in query string that typically consists of one key word only. By contrast, our approach sorts the context entities based on their weights. The higher context entity's weight gets, the more consideration is given to that context entity. Of course, not all context entities in the profiles have changing weights. The hardware profile has fixed context weights due to the stable hardware configuration at the client device.

Before a query is submitted to the search engine, it is expanded with context profiles sent by device proxy. The context profiles include the weights of the context entities calculated at proxy,  $p = [CW_{i+1}, CW_{i+2}, CW_{i+3}, \dots, CW_{i+n}]$ , where  $i = 0, 1, 2, \dots$ . Let the query forms the set,  $q = [T_1, T_2, T_3, \dots, ]$ . After query expansion, the augmented context profile is:

$$AP = \sum_{i=1}^m CW_i + \sum_{j=1}^n K(T_j) \quad (8)$$

where  $K(T_j)$  is the offset function. An example of augmented context profile could be  $AP = [CW_{i+1}, CW_{i+2} + K(T_2), CW_{i+3}, \dots, CW_{i+n}, K(T_{i+n+1}), \dots]$ . If the terms/words from the query set are matched in the context profile, then the offset weight of the term/word,  $K(T_j)$ , is added to the context weight,  $CW_{i+n}$ . Furthermore, if the terms/words from the query set are not found in the context profile, then the offset weights of these terms,  $K(T_{i+n+1})$ , are merged to the context profile. It is obvious that the augmented context profile is likely to have more weights on the query words.

### B. Context search

A number of popular search engines, like Google, apply *tf-idf* weight (term frequency - inverse document frequency) in their search algorithm [15]. The term frequency for any given document refers to the sum of the occurrences of the considered word in the query. This frequency serves as a measurement that indicates how relevant the given document is to the searched word. If the term frequency in a given document is higher for a word in the query, then this document is more likely to be included in the returned search result. The document frequency basically calculates how many documents that contain a given word against the total number of the documents in a collection. If the document frequency is higher, then the searched word appears in a lot of documents in the collection. The *tf-idf* weight is calculated by dividing the term frequency of a given word by the document frequency of the given word.

The weights of augmented context profile are sorted in the descending order. Instead of calculating *tf-idf* weight of each term in document collection, our method takes the augmented context profile, AP, calculated and sorted in the previous section and derives the *tf-idf* weights,  $d = [CTF_1, CTF_2, CTF_3, \dots, CTF_n]$ , from the document collection for all the context entities and query terms/words.

At this stage, a number of optimization strategies could take place to expedite the search process. Techniques such as automatic text categorization could be used for document

pre-processing [16]. Documents are classified into categories. Database servers are assigned to store individual categories as online information explodes. Thus, query search is carried out in parallel fashion. In addition, the database selector classifies the context profiles and select category-specific databases to find documents.

### C. Result Merging

The search engines return all the documents related to the query based on the augmented context profile. The returned documents are grouped into categories and ranked in descending weights in the context profile. Categories that match more context weights in augmented context profile have higher rank. Because most mobile devices have small displays, the maximum documents in each category are limited to a number. Obviously, there is a trade off between more categories displayed with fewer documents in each category and fewer categories displayed with more documents in each category. This method relies on the user selection to further narrow down user current context.

### D. User feedback

When the user selects the data or services downloaded from the server, his/her selection would greatly help the system understand user's intention. Hence, it is important to utilize the user's selection, as feedback, to adjust context weight calculation at both client and server sides. The server embeds the calculated context weights in data to the user device. The embedded context weights are transparent to users. However, the user selection on the data allows the related context weights to feed back into the proxy. This feedback mechanism significantly improves the accuracy of the server response.

## 5. Context Simulation

In this section, we provide two user scenarios that are close to the daily activities of mobile users. The value of  $\mu$  is set to 0.05 in our simulation. To simplify the context computing, we quantify some context entities as real numbers. *Weather*: sunny = 1, cloudy = 2, raining = 3; *Location* is expressed by (x, y) such as (1, 1), (2, 2), ...; We arbitrarily assign numbers to *store type*: department store 10, Woman Cloth = 20, foot wear = 30; *food store* has a type in 40, specifically Chinese food counter = 41, Japanese food counter = 42, Subway = 43, Pizza Hut = 44. To simplify the simulation, we only present a few context entities in each simulation.

### A. Scenario One

In scenario one, the user carrying mobile device walks at varying speeds in a fixed direction. During his trip it rains. User runs and stops at a gas station, waits for the rain to stop. Seven steps describe the first scenario:

- 1) The user walks by location (1, 1) at the speed of 3 mile/h when the weather is sunny (1), and the temperature is 77
- 2) While the user keeps walking at the same speed passing location (2, 2), the weather changes to cloudy (2) and the temperature drops to 73.
- 3) When it starts raining (3) and temperature continues to drop to 69, the user is running at the speed of 6 mile/h towards the same direction and passing location (4,

- 4) User is still running, passing location (6, 6). Temperature changes slightly to 68. It is raining (3).
- 5) User keeps running at the same speed reaching location (8, 8) while it is raining (3). Temperature is unchanged.
- 6) User stops at location (8, 8) while temperature drops slightly to 67 and it is raining (3).
- 7) User stays at location (8, 8) while temperature rises slightly to 68 and it is raining (3).

The above 7 steps are summarized in Table 1.

Table 1: Context entities of scenario 1

Step	S	W	T	x	y
S(1)	3mile/h	1(sunny)	77	1	1
S(2)	3mile/h	2(cloudy)	73	2	2
S(3)	6mile/h	3(raining)	69	4	4
S(4)	6mile/h	3(raining)	68	6	6
S(5)	6mile/h	3(raining)	68	8	8
S(6)	0mile/h	3(raining)	67	8	8
S(7)	0mile/h	3(raining)	68	8	8

Simulation results for scenario 1 are given in Table 2.

Table 2: Virtual frames of scenario 1

Ref.	S	W	T	x	y	t
f(1)	1.0	1.0	1.0	1.00	1.00	0
f(2)	0.0	2.0	1.1	2.00	2.00	3
f(3)	1.1	2.6	1.2	3.05	3.05	5
f(4)	-0.1	1.9	1.3	3.65	3.65	5
f(5)	-1.9	1.5	-1.4	4.13	4.13	5
f(6)	-0.7	1.2	-1.3	1.17	1.17	10
f(7)	-3.8	1.0	-1.1	-1.30	-1.30	15

In Table 2, the first column from the left represents the virtual frames calculated at different time instances. The other columns store the weight values for the context entities. The first virtual frame has all context weights set to 1 initially as  $f(1) = \{1.00, 1.00, 1.00, 1.00, 1.00\}$ , because the client device has no prior knowledge of the user state. The proxy would update weights as it adapts to context input. The speed changes significantly from step 2 to step 3 and from step 5 to step 6. As we can see the speed weight values increase in reference frame 3 and 6. The weather changes dramatically in the first three steps. The weight rises to reflect weather changes. It drops as it continues to rain. Also we observe the weight in x and y drops as the user stops. The last column shows the accumulated network time of the mobile user at each specific location. It is obvious that this accumulated time is proportional to user's stay at any location. In Table 3, the network returns content pertaining to the current user context entities.

Table 3: Context Result for scenario 1

Source	Description	Class
weather.com	National Forecast, free local weather alerts	120
Yahoo! weather	Yahoo weather forecasts, resources, and categories.	120
Local.com	Free live weather forecast	120
Shell	Locate Shell gas station	150

BP Gas	Great deals on gas	150
Exxon	Speedpass simplifies your life	150
MapQuest	Maps, directions, and more	230
Yahoo! Maps	Street maps and driving directions	230

The context search engine organizes and merges data based on context weights and accumulated network time. Each category has 3 documents at most to guide the users. Weather context has higher weights in the context profile at the end. The context search rates weather information higher than any other category. The accumulated stay time at a near by gas station increase the location-dependent information such as service at gas station and driving directions. It is intuitive that user selection on this set of data would further feed back the device proxy and hikes context weights related to the selection.

### B. Scenario Two

In scenario two, a mobile user goes shopping in the mall. The user walks from store to store. We describe user activities through the following steps.

- 1) User is walking by department store which has store type 10 and location coordinate (1, 1) at the speed of 2 mile/h while the noise level is at 60 db.
- 2) User is still walking at the same speed passing Woman Cloth with store type 20 and location coordinate (1, 2). The noise level rise to 63 db.
- 3) The noise level remains the same. User walks by foot ware store with type 30, location (1, 3), and at the speed of 1 mile/h.
- 4) User walks at 0.3 mile/h and passes Pizza Hut with type (44). The noise level continues to rise to 68 db. Pizza Hut has location coordinate (2, 4).
- 5) User walks by Japanese food stand (type=42) at 0.2 mile/h. The noise level is at 67 db at this food stand with coordination (2, 6).
- 6) User arrives Subway which has store type = 43. User walks by at this food store whose location coordinator is (2, 7) with a noise level of 67db.
- 7) User reaches the Chinese food stand and is waiting in line. Noise level rises to 68 db.

The above 7 steps are summarized in Table 4

Table 4: Context entities of scenario 2

Step	Speed	Store Type	Noise	x	y
S(1)	2mile/h	10	60	1	1
S(2)	2mile/h	20	63	1	2
S(3)	1mile/h	30	63	1	3
S(4)	0.3mile/h	44	68	2	4
S(5)	0.2mile/h	42	67	2	5
S(6)	0.2mile/h	43	68	2	6
S(7)	0mile/h	41	68	2	7

The simulation results are listed in Table 5.

Table 5: Virtual frames of scenario 2

Ref.	Speed	Store Type	Noise	x	Y
f(1)	1.00	1.00	1.00	1.00	1.00
f(2)	0.67	2.00	1.05	1.00	2.00
f(3)	1.22	2.55	0.10	1.05	2.55
f(4)	2.02	3.17	0.28	2.15	2.98
f(5)	2.50	3.31	0.44	1.70	3.38
f(6)	2.08	3.54	0.66	1.40	3.78
f(7)	3.34	3.83	-1.80	1.22	4.20

In this scenario, the speed weights capture the significant speed drops from step 2 to step 7. The noise level in the background remains relatively same, the noise weights in virtual frames drops as expected. The location weights in y rise as user walks. However, the weights in x drop when the x value changes a little. The weight for store type changes significant as user goes by different stores, but its change is less dramatic once the user roams at food court.

In Table 6, the network returns content pertaining to the current user context entities.

Table 6: Context Result for scenario 2

Source	Description	Class
Buffer King	All you can eat for \$10.99	140
Checker	99c menu for you	140
Tai's Cuisine	Experience the Exotic Thai food.	140
Sears	All TVs on sale	130
JCPenny	Huge Savings for rooms	130
Wal-Mart	Speedpass simplifies your life	130
MapQuest	Maps, directions, and more	230
Yahoo! Maps	Street maps and driving directions	230

The context-awareness server is able to sense that the mobile user is roaming in the food court. It provides information pertinent to restaurants. In addition, it offers user information about close stores and help for directions.

## 6. Conclusion

As the portable wireless devices become more pervasive, wireless users expect more value-added services to improve the experiences. In this paper, context awareness is build into the proposed architecture which consists of the proxy at the client site and context search at the server side. A number of algorithms and schemes are applied at the proxy site to calculate the weights of the context entities. Particularly, the concepts of virtual frame and reference frame have been introduced to capture and analyze context entities from different sources. Server side takes the context profiles as the input and searches the results which are best fit into mobile users' current context. In addition, two scenarios were analyzed through simulation to test the performance of the proposed architecture. The simulation results are listed to assess the performance of the client proxy and context search engine. This new design does not

require too much processor time since the weight update calculations are simple and carried out at client site. The weight changes do reflect the mobile user's context change. Therefore, network servers are able to learn the user situation based on the compiled context profiles and provide users services in relation to their situation.

## Acknowledgements

This research was supported by National Science Foundation Grants EIA-9906600, HRD-0317692, CNS 042615, and the Office of Naval Research Grant N00014-99-1-0952.

## Reference

- [1] G. D. Abowd and E. D. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous computing", ACM Computer Human Interaction, 2000
- [2] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the market place", IEEE Personal Communications, 1997
- [3] A. Bhattacharya, A. Roy and S. K. Das, "Towards a Novel Architecture to Support Universal Location Awareness", Intl. MISIL Workshop, pp. 182-189, Vienna, 2001
- [4] F. Gui, C. Liu, X. Chen, and D. Wang, "A Context-Awareness Algorithm Based On Data Cache for Mobile Computing", World Wireless Congress, San Francisco, 2004
- [5] M. Adjouadi and M. Ayala, "Introducing Neural Studio: An Artificial Neural Networks simulation for Educational Purpose", Computers in Education Journal, Vol. 14, No. 3, pp. 33-40, July-September 2004.
- [6] B. Clarkson and A. Pentland, "Unsupervised Clustering of Ambulatory Audio and Video", In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1999, volume 6, pages 3037-3040, 1999.
- [7] J. Himberg, J. A. Flanagan, J. Mäntyjärvi, "Towards Context Awareness Using Symbol Clustering Map, Workshop on Self-Organising Maps", Kitakyushu, Japan, September 11-13, 2003.
- [8] T. Kohonen, "Self-Organizing Map", Springer, Berlin, Heidelberg, 2001. (Third Extended Edition)
- [9] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Vande Velde, "Advanced Interaction in Context. In Hand Held and Ubiquitous Computing", number 1707 in Lecture Notes in Computer Science, pages 89-101. Springer-Verlag, 1999.
- [10] I. Chlamtaca and J. Redi, "Mobile Computing: Challenges and Potential", Encyclopedia of Computer Science, 4th Edition, International Thomson Publishing, 1998
- [11] E. J. O'Neil, P. E. O'Neill, and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering", ACM 1993.
- [12] J. T. Robinson and M. V. Devarakonda, "Data Cache Management Using Frequency-Based Replacement", ACM, 1990
- [13] P.D. Michailidis and K.G. Margaritis, "On-line String Matching Algorithms: Survey and Experimental Results".
- [14] N. Sebe, "Multimedia Information Retrieval: Promises and Challenges".
- [15] M. Sahami and T. Heilman, "A Web-based Kernel Function for Matching Short Text Snippets", International Workshop located at the 22nd International Conference on Machine Learning, 2005
- [16] A. Aizawa, "Linguistic Techniques to Improve the Performance of Automatic Text Categorization", Symposium (NLPRS2001) pages 307-314, 2001.