

Web Searching for Daily Living

Takuya Maekawa

Yutaka Yanagisawa

Yasushi Sakurai

Yasue Kishino

Koji Kamei

Takeshi Okadome

NTT Communication Science Laboratories, 2-4 Hikaridai Seika-cho, Souraku-gun, Kyoto, Japan
{maekawa,yutaka,yasushi,yasue,kamei}@cslab.kecl.ntt.co.jp, tokadome@acm.org

ABSTRACT

The new concept proposed in this paper is a query free web search that automatically retrieves a web page including information related to the daily activity that we are currently engaged in for automatically displaying the page on Internet-connected domestic appliances around us such as televisions. When we are washing a coffee maker, for example, a web page is retrieved that includes tips such as ‘cleaning a coffee maker with vinegar removes stains well.’ A method designed on the basis of this concept automatically searches for a web page by using a query constructed from the use of ordinary household objects that is detected by sensors attached to the objects. An in-situ experiment tests a variety of IR techniques and the experiment confirmed that our daily activities can produce related web pages with high accuracy.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval.

General Terms: Algorithms, Experimentation

Keywords: Sensors, Web search, Daily living, Experiment.

1. INTRODUCTION

Web pages are conventionally displayed on personal computers. However, they will also be displayed on various domestic appliances and pieces of furniture that are connected to the Internet. Many televisions have already been equipped with LAN ports, and some of them can display web pages. Internet-connected kitchen appliances such as refrigerators and microwaves can show a web page on their screens [13]. Also, Internet-enabled music players, which are increasingly a part of modern life, will be able to read web page text. Furthermore, an always-on gadget with a web browser, which is installed in living environments and constantly displays web content, has become commercially available [3]. Therefore, we will soon be able to enjoy web pages anywhere and at any time from devices around us to obtain useful advice. We can all benefit from information related to the real world activities in which we are currently involved. Showing the information on the domestic appliances around us in real time may enrich our daily lives. However, domestic appliances such as televisions may not have an interface such as a keyboard to enable us to in-

put queries quickly. Query inputs may also interrupt the real world activities. Many studies try to search for information about real world activities without explicit queries for appliances (devices) without rich interface. [7] retrieves web pages related to the TV broadcast news that a user is watching. The Remembrance Agent [23] retrieves documents related to the people around a user and shows related documents on a head mounted display (HMD). [8] proposes a framework for retrieving web pages related to a user’s context, e.g., the latitude and longitude of the user’s location, by manually attaching metadata that also represent context, e.g., latitude and longitude, to web pages in advance.

This paper describes our attempt to search for web pages related to the activities of daily living (ADLs) without any metadata. When shaving, for example, we search for such tips as ‘the best time to shave is about ten minutes after you wake up.’ We try to generate queries automatically about ADLs currently being undertaken, to search the web, and to show a web page that matches the query. This kind of web search is called a *query free search* and web pages provided as search results are responses to automatically generated queries. To realize query free searches in our daily lives, we monitor our ADLs by using sensors. Advances in sensing technologies have led to the mass and low-cost production of small sensors such as accelerometers and RFID tags that are used to monitor environments and detect human activities. (See section 3.) In the present work, we attach these sensors to daily objects such as cups, toothbrushes, and shavers and monitor their movements during daily life. To search the web by using objects, we assume that a set of names of objects that are used (moved) by a user in a given period of time corresponds to user’s context. For example, when a cup, milk, and cocoa are moved in a given period of time, “cup milk cocoa” becomes the user’s context. Then, we search for a web page that matches the context. That is, we make a query/queries from the context to search the web. The strategy of the method reflects our idea that ‘a web page including the names of objects that are used in an ADL may relate to the ADL.’ In addition to tips about ADLs, we can also obtain web pages that include information about the object itself such as a page about the health benefits of cocoa.

One advantage of our approach, which translates sensor readings (object usage) into terms (object names), is that we can incorporate existing techniques of IR studies, e.g., term weighting, query disambiguation, query generation, and ranking algorithms, with some modifications into the daily life web search. Another advantage is its very simple settings. That is, the method simply requires the names of objects and the time period in which the objects are used. Of course, installing sensors (tags) into home only for this web search is not practical. However, in the near future, cheap tags will be embedded into objects (products) during the manufacturing process in place of bar-code.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR ’09, July 19–23, 2009, Boston, Massachusetts, USA.

Copyright 2009 ACM 978-1-60558-483-6/09/07 ...\$5.00.

Providing ADL related web pages has the following benefits. (1) As with the above example of shaving, we can access information that enriches our daily life and that we may not be aware of the existence of it in the *world wide* web. (2) As shown by the popularity of books and TV shows about trivia, many of us feel satisfied when we obtain background knowledge about items and phenomena in the world. Our method can satisfy this desire for knowledge about daily living and make ADLs that occur routinely fun. (3) The method can provide us with information about a (new) ADL product. Obtaining the information may convince us to buy the product.

This paper contributes to the creation of the concept of a daily life web search that uses daily objects and the large-scale test of the web search methods in a real environment by using as many as 50 nodes. Our method consists of several components such as query generation and web page re-ranking components. In the test, we test various kinds of query generation and re-ranking algorithms to evaluate them in the real daily life environment. To our knowledge, this is the first work to search for ADL related web pages by using the object-based real world information. We can connect real world activities with the web by translating object usage into words. Note that the aim is to search automatically for information related to ADLs, *not* to provide answers to a user initiated search.

We have already implemented web browsers on domestic appliances such as televisions, refrigerators, and stationary music players that are embedded in living environments, and experimentally confirmed the usefulness of the daily life web search. The browsers automatically recommend (present) a web page that matches the user's context or profile, e.g., web pages are presented that are related to a TV program currently being watched, a web page including tips and trivia about a user's current ADL, and a newly arriving news page that matches the profile. As regards the ADL related pages and news pages, the browsers inform the user of the web page presentation by using a sound effect notification and visual signals. The browsers' behavior is designed not to disturb the user's daily life. For example, the browsers can control the notification level according to the user's estimated interruptability. Fig. 1 shows a prototype web browser installed on a refrigerator. The browser presents a web page that is related to an ADL such as cooking, which is performed near the browser. The browser also supports the user's browsing by automatically scrolling the presented page only when the user is looking at the browser screen by using a face detection technique. In a pilot usability study, we confirmed the usefulness of the three web page presentation methods. In the study, eight evaluators gave the usefulness of the TV program related page, the ADL related page, and the news page presentations average ratings of 6.00, 6.25, and 6.50 (7 is best).

The rest of this paper is organized as follows. After describing related work and the background to our study, we explain our approach, provide an experiment of our proposed methods, and conclude the paper.

2. RELATED WORK

2.1 Context aware search

There are many studies that try to (automatically) retrieve documents related to user's current context. We can roughly divide approaches of the studies into two categories. The first approach detects/infers user's current context and then retrieves a document whose metadata match the context. For example, [8] achieves context aware web page retrieval by manually attaching a specific context such as location information as metadata to web documents in advance. The metadata enable a user to automatically retrieve a web document that relates to the user's location obtained from a



Figure 1: Prototype web browser installed on refrigerator.

GPS sensor. Also, an information system at Oslo airport in AmbieSense project [19] uses detected user's location, user's preference, status of user's flight, etc. to search for a web page. That is, a web page has metadata including location of where the page should be made accessible to users and keywords of the page.

The second approach expresses user's context in words, e.g., sentences or bag-of-words, and then retrieves a document whose content matches the context. For example, [7] retrieves web pages related to TV broadcast news that a user is watching by generating the user's context from the closed captions of the news. [11] reinforces a user's query by employing the user's context, i.e., terms that appear in documents that the user is writing. [2] generates user's context from sentences on which a user focuses in documents by using an eye tracker. We adopt this second approach because it requires no metadata and we can incorporate various IR techniques into the daily life web search as described in section 1. To achieve context search, studies in this category basically (1) generate a query from context, (2) expand the query, (3) send the query to a standard search engine (SE), and (4) re-rank the search result. We also employ this procedure on our method, which is described in section 4. Note that all studies in this category do not exactly follow the procedure. Also, [9] mentions a special search engine to incorporate context (sensor data) into its ranking algorithm by using machine learning techniques. In this case, context may not have to be words.

Here, we can search for an ADL related web page after inferring the ADL. When we detect a brush teeth ADL, for example, we can retrieve a web page by using a "brush teeth" query. However, our approach is simple and does not require ADL estimation, which is not perfect. In addition, most ADL estimation methods [25] require teaching signals in an environment, and this places a burden on end users. In fact, in our another project, it took about one week for four persons to create teaching signals by using one month of sensor data and video recordings. Furthermore, ADL estimations require a predefined taxonomy for the ADLs. We consider there to be a wide variety of ADLs. We confirmed that there were unexpected ADLs (22%) in our experiment described below. (An example is that a user simply lifted and looked at a bottle of essential oil in which the user was interested.) Our method can retrieve web pages about such ADLs (activities).

2.2 Information presentation

Ambient displays and peripheral displays are abstract and aesthetic displays portraying non-critical information on the periphery of a user's attention [15]. Many kinds of real world information such as that related to the day to day activities of the elderly [18] have been shown on ambient displays and peripheral displays. The results of our daily life web search are also suitable for presentation on these displays. Our implemented web browsers described in section 1 are kinds of peripheral displays.

3. BACKGROUND

3.1 Sensor node

We assume that sensor nodes are attached to daily objects to

monitor their use. We use our developed sensor node that is equipped with a three-axis accelerometer and obtains its acceleration every 30 msec and sends it to a data server. We use the accelerometer for the following reasons: (1) We can detect the movement of objects independent of both its direction and the position of the accelerometers attached to the objects. (2) Accelerometers, which are frequently used for entertainment, security, and maintenance purposes, have been growing in importance. This means we can expect both their cost and size to decrease. (3) Our method employs the time period during which an object is used. Because the time period is simple and fundamental measure of ADLs related to objects, we use an accelerometer that detects the use of most objects.

Our method simply assumes that a time period during which acceleration data change greatly coincides with that during which an object is used (moved). Note that, instead of an accelerometer, we can adopt any sensor that can detect the time period for this method. [21], for example, employs a system that senses object use by using RFID tags attached to objects and an RFID reader belonging to the user. We consider that ubiquitous sensor environments in a home will start with inexpensive RFID tags attached to products in place of bar-code. We can also detect the time period by using capacitive touch sensors, two-axis accelerometers, and tilt sensors.

3.2 Definition of object use

We assume that a time period during which acceleration data change greatly coincides with the period during which an object is used. We call this time period an *activity*, which is a term used in the signal processing research field. Because we use a three-axis (x, y, and z) accelerometer, the combined activities extracted from each axis sensor data are defined as the time period during which an object is used. Some signal processing studies use the Gaussian Mixture Model to learn the Fourier components of the activity and the noise (non-activity) time periods [24]. We adopt a similar approach to detect activities. (We omit a detailed explanation here.)

4. APPROACH

Fig. 2 shows the architecture of the web search method for daily living. The method computes activities of objects in the past t minutes for each object every t minutes. It searches the web by using the activities detected in t minutes. The method consists of two main procedures: ‘*Cluster objects*’ and ‘*Web search*.’ In ‘*Cluster objects*,’ we generate clusters consisting of objects that are simultaneously used in the same ADL in t minutes. Sometimes users use (move) objects that are not directly related to the ADL they are performing. When making tea, for example, user may move not only tea but also cocoa or salt, which are unrelated to making tea, but in the cabinet. Furthermore, two or more people may simultaneously perform different ADLs in a given environment. For example, one person may be shaving with a razor and another person may be using a toothbrush and toothpaste. To cope with these issues, we cluster objects by types of ADL. For example, as regards brushing, we group such objects as a toothbrush and toothpaste as a cluster.

In ‘*Web search*,’ we search the web by using the names of objects in the same cluster to obtain a web page for each cluster. In this implementation, we set t at three. In a preliminary experiment, the average ADL time period during which participants used multiple objects was about two minutes, therefore we assume that object use in a three-minute period yields sufficient accuracy. We explain *Cluster objects* and *Web search* in detail.

4.1 Cluster objects

‘*Cluster objects*’ consists of three procedures in Fig. 2. We

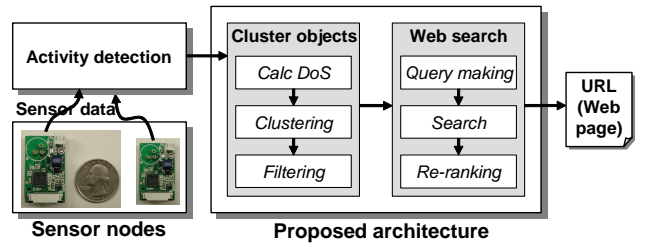


Figure 2: Architecture of web search for daily living.

call the measure for determining whether two objects are used in the same ADL the ‘*Degree of being used in the Same ADL (DoS)*.’ We assume that the ‘probability’ with which two objects are used in the same ADL increases as the *DoS* of the objects increases. We compute the *DoS* for all the objects that are used in the time period (*Calc DoS*) and cluster the objects according to the *DoS* (*Clustering*). We then filter the clusters including only objects that are used for a short time in the time period (*Filtering*).

4.1.1 Calc DoS

We explain how to calculate the *DoS*. This procedure computes *DoS* between objects X and Y in time period t (three minutes) based on the following three measures.

- $Temp(X, Y, t)$: This measure represents the degree to which X and Y are used simultaneously in t . It uses the activities of X and Y in t . This is based on the assumption that simultaneously used objects are employed in the same ADL with high probability.
- $Hist(X, Y)$: This measure represents the degree to which X and Y are used simultaneously in a past dataset. We prepare a past dataset of a certain period in advance. This measure reflects the view that the frequency with which objects were simultaneously used in the past dataset increases as the probability with which the objects are used in the same ADL increases.
- $Sem(X, Y)$: This measure represents the semantic relevance between X and Y . We use the co-occurrence of X and Y calculated by using page counts (hit counts) obtained from an SE. This is based on the view that objects that are ordinarily used together in the real world co-occur in the web, which reflects the real world.

Due to the page limitation, we do not provide detailed formula of three measures. By using the measures, the *DoS* is represented as follows:

$$DoS(X, Y, t) = Temp(X, Y, t) \cdot Hist(X, Y) \cdot Sem(X, Y).$$

4.1.2 Clustering and filtering

Here we briefly describe *Clustering* and *Filtering*. In *Clustering*, we use Ward’s method [6] to cluster objects. We use the *DoS* as inverse distance between objects described in section 4.1.1. After that, we eliminate outliers from each cluster as a postprocessing. *Filtering* eliminates clusters that have no object whose total seconds of activity in t exceed a threshold ε_{ac} . We set $\varepsilon_{ac} = 5$.

4.2 Web search

The *Cluster objects* procedure produces clusters that consist of a set of objects. The cluster corresponds to user’s context. We construct subqueries from each cluster and obtain web pages corresponding to the query. As shown in Fig. 2, this procedure consists of three procedures. First, we build multiple subqueries from a cluster (*Query making*) and obtain multiple search results for the subqueries (*Search*). Then, we re-rank the search results by computing the similarities between each of the web pages and the cluster (*Re-ranking*) and output the URL of the top-1 page in the re-ranked list.

4.2.1 Query making

Query making uses a vector representation of a cluster. In the representation, a cluster is represented in a vector that reflects the ‘importance’ of an object in the cluster. The importance of an object is simply defined as the largest *DoS* between the object and another in the cluster. The importance shows the contribution of the object used in its corresponding ADL. It reflects the semantic relationship between objects, whether an object is simultaneously used with another object, and an object use in a past dataset. *Query making* first produces a list (vector) of objects with their importance such as $\langle \text{juicer}, 3.0 \rangle, \langle \text{cup}, 3.0 \rangle, \langle \text{milk}, 2.0 \rangle, \langle \text{cup}, 1.0 \rangle$, and $\langle \text{sugar}, 0.5 \rangle$. If a list includes multiple objects with the same name, we remove the duplicate objects except for the object with the largest importance. $\langle \text{cup}, 1.0 \rangle$ is removed in the above example and a four-dimensional vector is obtained. We call the vector a **context vector**. The *Web search* method finds a web page that is related to an ADL by using a context vector. *Query making* applies the following three techniques to a context vector in order.

[1. Vector expansion]

Assume that a user made tea using green tea and a cup in a given time period. He/she then drank the tea using only the cup after that time period. Retrieving web pages that relate to drinking green tea is difficult by using a query constructed solely from the cup. However, we can easily retrieve web pages related to green tea by building a query from the ‘green tea’ that was used with the cup. We thus expand a current context vector V_i by using vectors that were constructed in the past. Let us define a similarity $Sim_h(V_i, V_j)$ between vectors V_i and V_j as follows.

$$Sim_h(V_i, V_j) = \lambda^{d(V_i, V_j)} \cos(V_i, V_j),$$

where $\cos(V_i, V_j)$ is a cosine similarity between V_i and V_j ; $d(V_i, V_j)$ is the temporal distance (minutes) between time periods of V_i and V_j ; λ ($0 < \lambda \leq 1$) is a forgetting factor [16] that controls the effect of past values. Assume that the number of dimensions (the number of objects) of a context vector V_i is equal to or less than a threshold ε_{hq} . Then the procedure *Vector expansion* finds a past context vector V_j that satisfies the following conditions; $Sim_h(V_i, V_j)$ has the maximum value among past vector similarity values that are greater than a threshold ε_{hw} ; not all the dimensions of V_j are identical to those of V_i . Otherwise the procedure does nothing. After finding V_j , the procedure expands V_i by multiplying V_j by $\lambda^{d(V_i, V_j)}$, and then adding it to V_i . In our implementation, we set $\varepsilon_{hq} = 2$, $\varepsilon_{hw} = 0.7$, and $\lambda = 0.99$.

[2. Making subqueries]

The procedure *Making subqueries* makes multiple subqueries by extracting some objects (terms) from a vector. By extracting objects, we can construct some subqueries that are not too strict and do not include noises, i.e., the names of objects wrongly included in the cluster constructed in the *Cluster objects* procedure. Before the construction, we compute the inverse document frequency (idf) of an object name for each dimension of the context vector. We use the function $\log(D/f(X))$, where $f(X)$ is the frequency of the object X ’s name in the document collection and D is the number of documents, to compute the idf. Because we use Yahoo! web search to compute $f(X)$, we set D as 20 billion. Then, we multiply the importance of each dimension by its idf. This multiplication can decrease importance of objects whose name appears frequently in web documents such as cup.

We simply construct subqueries from all combinations of objects

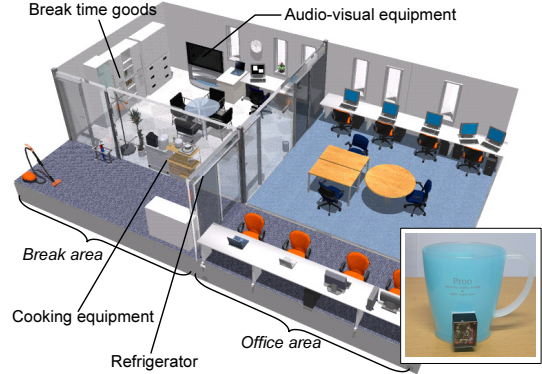


Figure 3: Experimental environment and our sensor node.

in a context vector. Assume that a context vector has i objects and the desired query length is l . We can make iC_l subqueries. Also, to limit the number of subqueries, we discard subqueries that do not include an object with a top- s importance. When $l = 2$ and $s = 2$, the example context vector (juicer, cup, milk, and sugar) shown at the beginning of this section produces subqueries “juicer cup,” “juicer milk,” “juicer sugar,” “cup milk,” and “cup sugar.” That is, this algorithm outputs combinations of objects while giving priority to more important objects. We set $s = 2$ in our implementation.

[3. Query expansion]

A query that includes only the names of objects may be ambiguous. That is, it is difficult to obtain our desired pages (daily life information) from such a query as “cup green-tea.” In [10], it is reported that combining a topic related term and a genre related term makes a good query. If we want to buy a camera, for example, we combine the topic term “camera” and the genre term “buying” or “choosing,” i.e., we make the query “camera buying.” We also expand the query by using genre terms that may yield web pages about daily lives. This algorithm randomly selects a genre term from the four terms (“advice,” “how-to,” “tips,” and “trivia”).

4.2.2 Re-ranking

We can obtain multiple search results (rankings) by sending multiple subqueries constructed in *Query making* to an SE. The *Re-ranking* is performed by using a similarity measure between a context vector and a web page in the search results to combine the multiple rankings into one ranking. The top-1 URL (page) in the re-ranked list is produced. We have prepared five re-ranking algorithms based on commonly used (re-)ranking algorithms in context search and/or meta search. We explain them in section 5.3.1.

5. EXPERIMENT

5.1 Dataset

We performed a manual evaluation of our experimental results by referring the evaluation methodologies of query free search and context search [7, 11]. By using the method presented in this paper, we produced web pages from sensor data observed in our experimental environment as a dataset; people who were in the environment during the experiment evaluated the pages. Fig. 3 shows the experimental environment which was constructed for our project [14]. We installed a table, break time goods, cooking utensils, etc. into the break area to emulate home environments. About ten workers (not researchers) undertook their normal work from 9 a.m. to 5 p.m. every weekday. Our experiment lasted for 16 days from 10 a.m. to 5 p.m. Fig. 3 also includes a photo of our sensor node attached to a cup. In the environment we attached our sensor nodes

Table 1: Objects in experimental environment.

Object names and #objects
cup*11, kettle, black-tea, green-tea, sugar, coffee-maker, cocoa, milk, jug, dish-soap, toothbrush, toothpaste, vacuum-cleaner, watering-can, CD-player, CD, trash-can, aromatherapy-diffuser, essential-oil, aquarium, fish-food, razor, preshave, hand-mirror, perfume, hanger, clothes-brush, pasta, pan, pasta-server, rice, rice-paddle, rice-cooker, cabinet*2, juicer, dietary-supplement, book, television, refrigerator

Table 2: Expected ADLs and their distribution ratio (%).

A	make tea: 4.13	N	practice aromatherapy: 0.74
B	make green tea: 2.95	O	feed fish: 0.89
C	make coffee (pour coffee): 7.52	P	shave: 2.80
D	make cocoa: 1.33	Q	use perfume: 0.59
E	drink something: 32.74	R	brush clothes: 1.47
F	pour water into a cup: 0.44	S	cook pasta: 0.88
G	pour milk into a cup: 1.03	T	cook rice: 3.54
H	pour water into a kettle: 0.88	U	make juice: 1.62
I	wash dish: 5.31	V	take supplement: 1.18
J	brush teeth: 2.06	W	read book: 0.88
K	vacuum: 1.18	X	watch TV: 0.59
L	pour water on plant: 1.03	Y	others: 22.41
M	listen to music: 2.21		

to the 50 objects listed in Table 1. We selected terms that well represented the nature of the objects from WordNet or Wikipedia as the object names. For example, we selected ‘milk’ for a sensor node attached to the surface of a milk carton. We selected objects based on ADLs listed in ADL estimation studies [27]. By attaching nodes to the objects listed in Table 1, we expected the 24 ADLs shown as A to X in Table 2. ADLs from A to N were also being performed daily outside the experimental period. We asked especially the workers to perform the all ADLs during the experiment, e.g., to make lunch in the environment. Most ADLs were performed in the break area. The sixteen-day experiment consisted of two sessions: the first eight days and the last eight days. We changed the positions of the objects and furniture for the two sessions. Also, we used data from the first four days for algorithms that leverage history as training data and used the last four days data as test data. During the days of the test, the average number of ADLs in Table 2 that occurred in a three-minute time period was 2.53. As regards the average, we did not include time periods in which no ADLs occurred. Table 2 also shows the distribution ratio of each ADL. We observed behavior that was classified under ‘others’ such as carrying around a cup or just looking at an object after picking it up.

5.2 Evaluation of clustering objects

We first provide our evaluation of the clustering method described in section 4.1. Due to the page limitation, we can only provide a summary of the evaluation about the clustering method. To obtain clusters, we applied the clustering algorithms to the test dataset. The workers manually compared the output clusters with recordings made by eight video cameras fixed to the ceiling. We achieved 0.942 precision and 0.969 recall. Surprisingly, *Sem* makes the most significant contributions to the precision and F-measure; we can achieve good precision and F-measure simply by using *Temp* and *Sem* when there is no history, e.g., just after a new object has been introduced into the environment.

5.3 Evaluation of web search

This section presents an evaluation of the web search method to confirm its accuracy. We applied the web search method to the

results of the clustering method proposed in section 4.2. We used Yahoo! [28] as the SE.

5.3.1 Evaluation methodology

The web search method outputs a URL (a web page) for each cluster. The workers evaluated the pages by comparing video recordings and the pages. In the evaluation, the workers use an ordinary web browser. They evaluated the pages mainly as regards two aspects: the relevance between a page and a corresponding ADL and the usefulness of the page. They selected the relevance between a page and an ADL performed in a given time period from {exactly-relevant, somewhat-relevant, not-relevant, page-not-found, no-result}. Assume that a user makes tea in a certain time period. If the page that corresponds to the ADL had information about how to make tea, for example, “exactly-relevant” was selected. If the page included information about the health benefits of tea, “somewhat-relevant” was selected. If the page described the history of the World Cup, “not-relevant” was selected. When the workers could not access the page due to a 403 or 404 error, “page-not-found” was selected. If no URL was output for a context vector, “no-result” was selected. The web search method cannot output a URL when an SE returns no result. In terms of usefulness, the workers subjectively graded the usefulness of the pages from {useful, somewhat-useful, not-useful, page-not-found, no-result} considering their situation by watching the recordings.

To validate the *Query making* and *Re-ranking* methods described in sections 4.2.1 and 4.2.2, we tested the following nine web search methods. The first three methods do not expand the vector using the history to validate its effect. In the following methods, we often obtain top-*n* pages from an SE. Note that the list of top-*n* pages does not contain pages that had already been presented to users. This is because we avoid presenting a web page repeatedly.

- Baseline method [*Base*]

This baseline method makes a query (not multiple subqueries) from the names of all the objects of a context vector and expands the query using genre terms. Then, it outputs a URL of the top-1 page of a search result for the query. This method is for validating the effect of the query length.

- Top-2 method [*Top-2*]

This method builds a query from the names of the top-2 objects of a context vector in descending order of importance. Other procedures are identical to *Base*. This method is for validating the effect of the query length. This method may correspond to a simple method that extracts two terms with largest tf-idf weight from sentences (context) [7].

- Top-3 method [*Top-3*]

This method constructs a query from the names of the top-3 objects of a context vector in descending order of importance. Other procedures are identical to *Base*.

- History method [*History*]

This method forms a query from the names of the top-2 objects of a context vector in descending order of importance after the vector expansion, which uses the history. It performs the query expansion using genre terms. It outputs a URL of the top-1 page of a search result for the query. This method is for validating the effect of the history.

- Markov chain method [*MC4*]

After the vector expansion, this method obtains multiple subqueries, i.e., multiple rankings, by following the procedures described in section 4.2.1. Here, we set the desired query length $l = 2$. The method combines the rankings into one ranking simply by using

rank order and it outputs a URL of the top-1 page of the combined ranking. A procedure for obtaining a ranking from multiple rankings is called rank aggregation (RA), which is studied in the meta-search research field [17]. The method uses a Markov chain to perform RA. [4] uses a Markov chain for RA and obtains a good result in practice [5, 11]. In [4], the states of the chain are the pages (URLs) in the union of multiple rankings. The MC4 algorithm proposed in [4] defines the transition probability in the transition matrix from pages P to Q as follows. First, the algorithm uniformly selects Q from all states. If the rank of Q is higher than that of P in the majority of rankings, i.e., Q is better than P in the majority, the state transits from P to Q . If not, the state stays in P . The algorithm computes the stationary probability of the chain from the transition matrix. The descending order of the probability is the rank order in the combined ranking. This Markov chain method uses the MC4 algorithm.

- Cosine similarity method [*Cosine*]

This method also makes multiple subqueries and obtains multiple rankings for these subqueries. The method downloads the top- $(r/\#subqueries)$ pages of each ranking ($r = 50$). Then, it computes the cosine similarity between a context vector and each of the tf-idf vectors of the downloaded pages to re-rank the pages. It outputs a URL of the top-1 page of the re-ranked list. This method is for validating the efficacy of the most common similarity measure. Some simple context search methods use the similarity measure to compute similarity between a document and context (bag-of-words or word vector).

- Term distance method [*Dist*]

This method uses the following similarity measure in place of the cosine similarity in *Cosine*. A similarity measure between a page and a query that reflects the distance between query terms has been proposed in the meta-search research field [12]. The study shows that a web page where the query terms are located near each other includes text that may match the query well. The measure well matches our web search because we consider that a description (tips and how to) about an ADL may include the names of objects together that are also used together in the ADL. Some modifications of the measure enable us to use it as a similarity between a context vector and a web page. Let $V = \{ \langle t_1, w_1 \rangle, \dots, \langle t_N, w_N \rangle \}$ be a context vector. Also, let t be a stemmed term obtained with a Porter stemmer [22], w be the importance of the term, and W be a stemmed web page text from which HTML tags have been removed. The modified similarity measure $R_d(V, W)$ is defined as

$$R_d(V, W) = c_1 N_p(V, W) + \frac{N_t(V, W)}{c_3} + \left(c_2 - \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^N \min(D_1(t_i, t_j, W), c_2)}{\sum_{k=1}^{N-1} (N_p(V, W) - k)} \right) \frac{c_2}{c_1},$$

$$N_p(V, W) = \sum_{i=1}^N n_p(t_i, W) w_i, \quad N_t(V, W) = \sum_{i=1}^N n_t(t_i, W) w_i,$$

$$n_p(t, W) = \begin{cases} 1 & (n_t(t, W) > 0) \\ 0 & \text{otherwise,} \end{cases}$$

where $n_t(t, W)$ is the number of occurrences of term t in W ; $D_1(t_i, t_j, W)$ is the minimum distance (the number of characters) between t_i and t_j in the page; c_1 is a constant controlling the magnitude of R_d ; c_2 is a constant specifying the maximum distance between terms; and c_3 is a constant controlling the importance of the term frequency. We set $c_1 = 100$, $c_2 = 5000$, and $c_3 = 10c_1$ according to [12].

- Text analysis method [*Text*]

We can obtain web pages including tips and trivia about ADLs with

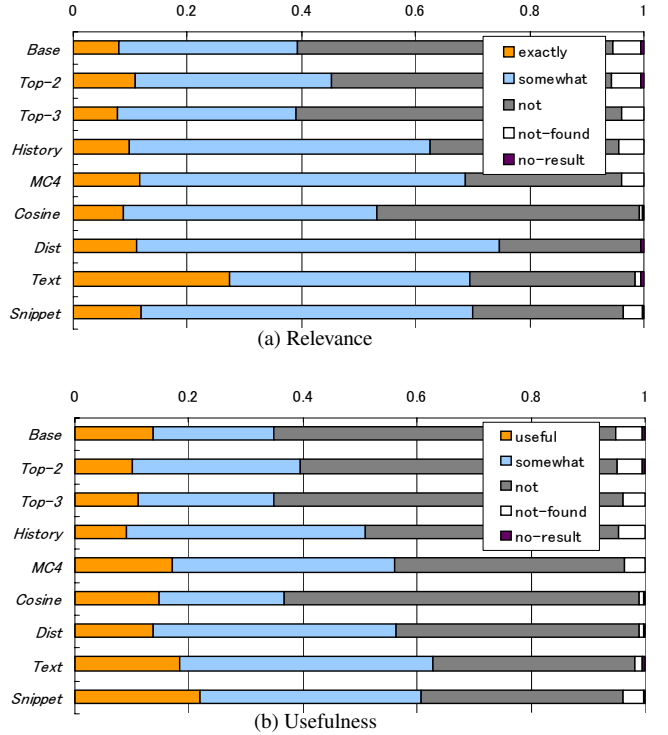


Figure 4: Evaluation results of relevance and usefulness.

high accuracy by combining several features in addition to the term distance [1]. The method calculates the scores of web pages using several heuristics, and it re-ranks the pages according to the score. (Other procedures are identical to *Dist*.) We introduce some of them.

- (1) A preliminary experiment indicates that most product promotion and shopping pages relate to ADLs (or ADL related objects), but some participants dislike such pages; we thus assign a negative score to a page that has many hyperlinks to shopping sites registered in the ‘Shopping’ category of the open directory project [20].
- (2) An examination of ‘good’ pages in a preliminary experiment showed that many include a description of an ADL that includes object names that follow a subtitle or a preamble such as “House-keeping trivia.” We assign a positive score to a page where a paragraph that follows a genre term includes a query term.
- (3) Leading paragraphs and page title provide useful information [26]. We thus assign a positive score to a page whose first paragraph or title include a query term.

- Snippet method [*Snippet*]

Dist and *Text* methods must download web pages. The search results of most SEs include a short summary of each ranked page called a *snippet*. A snippet text includes query terms. The text in the snippet is separated by a delimiter such as ‘...’ if the query terms are not located near each other on the page. The method uses a snippet instead of a downloaded page. The similarity between a context vector V and a stemmed snippet S is calculated by using $R_d(V, S)$ mentioned in *Dist*. In this case, D_1 returns c_2 when there is a delimiter among the terms. We set $c_3 = \infty$, i.e., we do not use the term occurrence, because snippets are short. This method computes the similarity between a context vector and each snippet of the top- $(r/\#subqueries)$ pages in each ranking ($r = 200$).

5.3.2 Web search results

Fig. 4 shows the experimental results for each method. The

workers totally evaluated 12,762 pages. It took about 250 hours totally. The usefulness results are the averages for two workers to reduce the bias imposed by a subjective evaluation.

[Query length]

Fig. 4 (a) shows the relevance results. *Base* retrieved ADL-related pages had about a 39.4% accuracy (the sum of the ‘exactly-relevant’ and ‘somewhat-relevant’ ratios). *Top-2* achieved 45.3%. On the other hand, the accuracy of *Top-3* was 39.1%. The accuracies of *Base* and *Top-3* were probably worse than that of *Top-2* because *Base* and *Top-3* return search results that are too narrow since they employ a long query. In fact, the average hit count for queries whose length in *Base* was four or more was 10,722. On the other hand, that of queries whose length was three in *Top-2* was 77,297. The #hit count column in Table 3 shows the average hit count (page count) of the search result for each method. Note that, with a method that makes multiple subqueries from a context vector, the number is the average over the search results including the output URLs (web pages) as result of our method. The #terms column in Table 3 shows the average query length of each method. The methods whose average length is about three achieved high accuracy (*MC4*, *Dist*, etc.). Also, the average accuracies when the query length was not three in *Base* and *Top-3* were only 23.1% and 22.8%. A query consists of three terms, i.e., two object names and a genre term, is reasonable to obtain object/ADL-related pages.

[Contribution of history]

History achieves 62.7% accuracy in relevance, which is much higher than that of *Top-2* owing to the history of the object use. The history is effective for context vectors whose dimension is small before vector expansion because the history compensates for low-dimensional vectors. In fact, although the average accuracy of URLs (web pages) obtained from vectors whose original dimension is one was 23.6% in *Top-2*, it was 55.7% in *History*.

[Effect of rank order and multiple subqueries]

The difference between *Top-2* and *History* is also observed in a frequency of the same query issuance. For example, the average number of times that the top-10 frequent queries were issued was 35.6 in *Top-2*. On the other hand, it was 25.6 in *History*. This is because, for example, when a worker drinks something using a cup, *Top-2* forms a query based only on the cup. In contrast, *History* uses green tea, black tea, or cocoa in addition to the cup. The rank column of Table 3 shows the average rank of output URLs in search results for each method. Because *History* queries are more various than those of *Top-2*, the average rank for *History* was high. Because *Base* and *Top-2* produce the highest ranked URLs *except for* those that have already been presented to the users, they output lower ranked pages as the same queries are repeated. The rank average of *MC4*, which achieves 68.7% accuracy, was high because it can select high ranked pages from multiple rankings. Higher ranked pages, which are ranked by various measures adopted by the SE, may be definitely better than lower ranked pages. In fact, we confirmed a significant difference between the rank averages of relevant and irrelevant pages in *Top-2* (5.5 and 28.2) by t-test ($p < .01$). Also, there was a moderate correlation (Spearman $\rho = .571$) between the relevance and the rank order of *Top-2* results. In contrast, the rank averages of *Dist*, *Text*, and *Snippet* are lower than that of *History*, but they achieve high accuracies. This is because these methods produce URLs that are low ranked but match the context vector.

By comparing web pages with context vector, we can assign a low score to a page of a search result for a noisy subquery. (Noise objects usually had low importance.) This is one of the advantages of the multiple subquery making. However, we could not find large

Table 3: Averages of # hit count, query length, rank order, and minimum character distance (relevant / irrelevant).

	#hit count	#terms	rank	minimum character distance
<i>Base</i>	1,989,342	2.54	17.68	553 / 1304
<i>Top-2</i>	2,001,837	2.41	17.52	597 / 1374
<i>Top-3</i>	1,853,503	2.52	17.26	633 / 1039
<i>History</i>	512,431	2.9	7.19	608 / 889
<i>MC4</i>	354,347	2.9	8.25	628 / 916
<i>Cosine</i>	375,878	2.9	16.06	683 / 1121
<i>Dist</i>	393,247	2.9	18.56	495 / 1064
<i>Text</i>	377,683	2.9	23.24	593 / 751
<i>Snippet</i>	356,758	2.9	17.49	476 / 835

differences in the relevance accuracy for noisy context vectors between methods with the multiple subquery making and methods without it (*MC4*: 61.9%, *Dist*: 67.0%, *History*: 63.9%). In this case, contribution of the multiple subquery making was not so significant contrary to our expectation.

[Impact of term distance]

The relevance accuracy of *Cosine* was only 53.2%. This indicates that the simple cosine similarity is not effective for our goal. On the other hand, *Dist* achieved 74.7% accuracy, which was the highest of all the methods. The highest accuracy attained by *Dist* shows that the term distance works well for our goal. Many web pages about ADLs in which we use objects (recipe, how to, and advice pages) include descriptions containing a list of required objects or procedures for the daily activities. In fact, the average minimum character distance between query terms (object names) in *Dist*’s relevant pages was 495 as shown in Table 3. In contrast, those of irrelevant pages of *Cosine* and *Top-2*, which do not use the term distance, were 1121 and 1374. Also, those of *Cosine*’s and *Top-2*’s relevant pages were 683 and 597. Clearly, the term distance of the relevant pages tends to be small. *Snippet* achieved 70.0% accuracy without downloading web pages. Also, the average minimum character distance in *Snippet*’s relevant pages was 476, surprisingly which was the smallest of all the methods. These facts indicate that the term distance in snippet also works well.

[Effect of web text analysis]

The ‘exactly-relevant’ ratio of *Text* is higher than those of the others. This is because *Text* reduces the score of such pages as product introductions that do not directly relate to ADLs. On the other hand, the ‘not-relevant’ ratio of *Text* is higher than that of *Dist*. Because *Text* increases the score of pages including tips and ‘how to,’ *Text* wrongly outputs some pages including them about other ADLs. For example, *Text* outputs trivia about how to clean a carpet by using dish soap for the ADL of washing cups by using dish soap. Also, Fig. 4 (b) shows the usefulness results. The usefulness results were basically associated with those for relevance. However the ‘exactly-relevant’ ratio of *Text* was much higher than those of the others, surprisingly the ‘useful’ and ‘somewhat-useful’ ratios were not very high. By examining the results, we confirmed that *Text* outputs some pages that relate well to ADLs but they include only ordinary information. In the experiment, a ‘somewhat-relevant’ page revealing that the order of pouring milk and tea into a cup was an indicator of rich and poor in some countries because pouring hot tea first may crack a low-quality cup, for example, was more useful for the workers than such ordinary information as how to make good milk tea, which is ‘exactly-relevant.’ In the daily life web search, ‘exactly-relevant’ pages are not always useful. In fact, 30.9% of *Text*’s ‘exactly-relevant’ pages were ‘not-useful.’

[Accuracy of relevance and further improvement]

Dist achieved a 74.7% accuracy in terms of relevance. There may

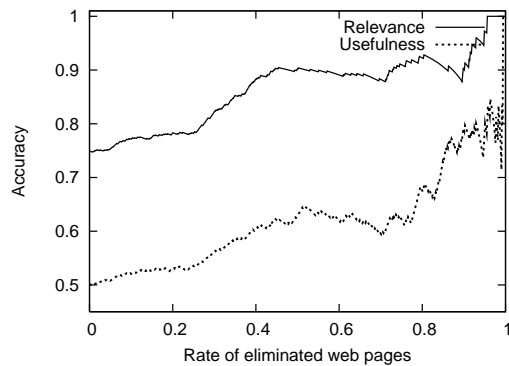


Figure 5: Filtering search results of *Dist* to improve accuracies of relevance and usefulness.

still be room for improvement in terms of accuracy. However, the accuracy of *Dist* is about twice that of the naive method *Base* (39.4%). Furthermore, an analysis of the results shows that only 47.5% of pages relate to the ADL “make tea” in each of the top-20 pages of a Google search result for queries that consist of the term “make-tea” and one of the four genre terms (80 pages in total), i.e., a simple web search using the name of the inferred ADL is low in accuracy. These facts indicate that our method contributes well to improvements in accuracy. Furthermore, we can improve the accuracy by filtering out inappropriate output pages. That is, we incorporate a certain threshold and stop a web browser from displaying a web page that falls below this threshold when our search method outputs the page. We use the similarity measure of a page computed in *Re-ranking* for the thresholding. Note that we multiply the measure by the squared Dice coefficient which is computed by using the top-2 query terms (object names) of the query that was used to retrieve the page. The appearance of many irrelevant pages (26.3%) were caused by vector expansion errors that are usually the result of clustering errors. Thus, by multiplying the coefficient, we can reduce the value of the measure for a page that is retrieved by using a mistakenly expanded query (e.g. “cup preshave how-to”). Changing the threshold permits us to control the number of filtered-out pages (and the accuracies as regards relevance and usefulness). Fig. 5 shows the relationship between the rate of filtered-out pages and the accuracies of *Dist*. When the rate is around 0.5, we can achieve a 90.4% relevance accuracy. In the experiment, an average of 88.6 pages were retrieved per day. That is, when the rate is 0.5, about 44 pages are presented to users each day. In fact, even this number of pages may provide too much information and we should control the number of pages presented to users.

As a result, we were able to retrieve ADL related web pages with high accuracy by using object usage information. Further improvement of the accuracy constitutes future work. For example, *Dist*, *Text*, and *Snippet* do not use ranking information of SEs. However, we found a correlation between the relevance and the rank order. Thus, we will improve their accuracy by combining these algorithms and *MC4*. In the future, we will conduct a user-centric experiment in real time. In the experiment described in this paper, we used sensor data collected in advance to evaluate all the implementations of the method under the same condition. Giving variety to page presentation and a personalized search may be important to the user study. As regards the variety, we may achieve this by not presenting pages with similar word vectors at short intervals. Also, context aware retrieval systems emphasize personalized search by using user’s profile and interest [8, 19].

6. CONCLUSION

This paper proposed a new kind of web search for daily living by using daily objects. To the best of our knowledge, our proposed method is the first object related web search method for electronic appliances that exist ubiquitously in our daily lives such as televisions and music players. One advantage of our search method is its simple settings. That is, the method simply requires the names of objects and the time period in which the objects are used, and this information can be obtained by using various sensors. Also, by using object names for web search, we could connect our daily life to the web. We tested the method with a large-scale experiment by using 50 nodes. In the experiment, we confirmed that we can retrieve web pages related to ADLs in which users are currently engaged in high accuracy by using sensors attached to daily objects.

7. REFERENCES

- [1] B. Bartell, G. Cottrell, and R. Belew, “Automatic combination of multiple ranked retrieval systems,” *Proc. SIGIR 1994*, pp. 173–181, 1994.
- [2] G. Buscher, “Attention based information retrieval,” *Proc. SIGIR 2007*, p. 918, 2007.
- [3] Chumby Industries, *chumby*, <http://www.chumby.com/>.
- [4] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, “Rank aggregation methods for the web,” *Proc. WWW 2001*, pp. 613–622, 2001.
- [5] R. Fagin, R. Kumar, and D. Sivakumar, “Efficient similarity search and classification via rank aggregation,” *Proc. SIGMOD 2003*, pp. 301–312, 2003.
- [6] J.F. Hair, R.E. Andersen, R.L. Tatham, and W.C. Black, “Multivariate data analysis,” 4th ed. Prentice-Hall, Englewood Cliffs, N.J.
- [7] M. Henzinger, B.W. Chang, B. Milch, and S. Brin, “Query-free news search,” *Proc. WWW 2003*, pp. 1–10, 2003.
- [8] G.J.F. Jones and P.J. Brown, “Context-aware retrieval: exploring a new environment for information retrieval and information filtering,” *Personal and Ubiquitous Computing*, 5(4), pp. 253–263, 2001.
- [9] G.J.F. Jones and P.J. Brown, “The role of context in information retrieval,” *Proc. SIGIR Workshop on Information Retrieval in Context*, pp. 20–22, 2004.
- [10] R. Kraft and R. Stata, “Finding buying guides with a web carnivore,” *Proc. the 1st Latin American Web Congress (LA-WEB)*, pp. 84–92, 2003.
- [11] R. Kraft, C.C. Chang, F. Maghoul, and R. Kumar, “Searching with context,” *Proc. WWW 2006*, pp. 477–486, 2006.
- [12] S. Lawrence and C.L. Giles, “Inquirus, the NECI meta search engine,” *Proc. WWW-7*, pp. 95–105, 1998.
- [13] LG Electronics, LG Internet refrigerator, http://us.lge.com/www/product/refrigerator_demo.html.
- [14] T. Maekawa, Y. Yanagisawa, Y. Kishino, K. Kamei, Y. Sakurai, and T. Okadome, “Object-blog system for environment-generated content,” *IEEE Pervasive Computing*, 7(4), pp. 20–27, 2008.
- [15] J. Mankoff, A. Dey, G. Heish, J. Kientz, S. Lederer, and M. Ames, “Heuristic evaluation of ambient displays,” *Proc. CHI 2003*, pp. 169–176, 2003.
- [16] S. Markovitch and P.D. Scott, “The role of forgetting in learning,” *Proc. ICML 1988*, pp. 459–465, 1988.
- [17] MetaCrawler, <http://www.metacrawler.com>.
- [18] E. Mynatt, J. Rowan, A. Jacobs, and S. Craighill, “Digital family portraits: supporting peace of mind for extended family members,” *Proc. CHI 2001*, pp. 333–340, 2001.
- [19] H. Myrhaug, N. Whitehead, A. Goker, T.E. Faegri, and T.C. Lech, “AmbieSense - A system and reference architecture for personalised context-sensitive information services for mobile users,” *Proc. European Symp. on Ambient Intelligence (EUSAI 2004)*, pp. 327–338, 2004.
- [20] Open Directory Project, <http://www.dmoz.org>.
- [21] M. Perkowitz, M. Philipose, D. Patterson, and K. Fishkin, “Mining models of human activities from the web,” *Proc. WWW 2004*, pp. 573–582, 2004.
- [22] M.F. Porter, “An algorithm for suffix stripping,” *Program*, 4, pp. 130–137, 1980.
- [23] B. Rhodes and T. Starner, “The remembrance agent: a continuously running information retrieval system,” *Proc. Int’l Conf. on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM96)*, pp. 486–495, 1996.
- [24] J. Sohn, N.S. Kim, and W. Sung, “A statistical model-based voice activity detection,” *IEEE Signal Processing Letters*, 6, pp. 1–3, 1999.
- [25] E.M. Tapia, S.S. Intille, and K. Larson, “Activity recognition in the home using simple and ubiquitous sensors,” *Proc. Pervasive 2004*, pp. 158–175, 2004.
- [26] A. Tombros and M. Sanderson, “Advantages of query biased summaries in information retrieval,” *Proc. SIGIR 1998*, pp. 2–10, 1998.
- [27] D. Wyatt, M. Philipose, and T. Choudhury, “Unsupervised activity recognition using automatically mined common sense,” *Proc. AAAI-05*, pp. 21–27, 2005.
- [28] Yahoo! Developer Network, <http://developer.yahoo.com>.