# Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering

**P. J. Brown and G. J. F. Jones**

*Department of Computer Science, University of Exeter, Exeter, UK*

**Abstract:** The opportunities for context-aware computing are fast expanding. Computing systems can be made aware of their environment by monitoring attributes such as their current location, the current time, the weather, or nearby equipment and users. Context-aware computing often involves retrieval of information: it introduces a new aspect to technologies for information delivery; currently these technologies are based mainly on contemporary approaches to information retrieval and information filtering. In this paper, we consider how the closely related, but distinct, topics of information retrieval and information filtering relate to context-aware retrieval. Our thesis is that context-aware retrieval is as yet a sparsely researched and sparsely understood area, and we aim in this paper to make a start towards remedying this.

**Keywords:** Context-aware; Information filtering; Information retrieval; Mobile computing

## 1. Introduction

Sensors are becoming ever cheaper and more ubiquitous, and wearable computers are steadily moving from being outlandish to being mainstream. As a result, there is increasing interest in context-aware mobile applications. A recent analysis of promising future context-aware applications [1] presents six areas; in five of these a key component is *context-aware retrieval* (CAR) of information. We believe the important role of CAR applies to context-aware applications in general, and the purpose of this paper is to analyse the nature of CAR and to relate it to the existing, well-developed fields of information retrieval (IR) and information filtering (IF). Typically CAR applications involve a mobile user, i.e. a user whose context is changing. We explain CAR in more detail below; if the reader wants an example, the archetypical CAR application is a tourist guide. Here a user carries sensors that record their context (perhaps their location, the current temperature, etc.), and the application continually retrieves information that is relevant to the current context. Thus, for a tourist guide the information would relate to nearby tourist attractions and facilities. After retrieval has occurred, the application presents (parts of) the retrieved documents in some way: for example, information might be displayed to the recipient on a PDA screen, a mobile phone or a head-up display; alternatively the retrieved information might represent a program to be run. We will not, however, concentrate on the display aspects here, as our topic is the retrieval stage.

The essential aims of IR and IF are to be *efficient and effective*. Efficiency means delivering information quickly and without excessive demands on resources, even when there is a massive amount of information to be retrieved. Clearly efficiency is extremely relevant to CAR as late information is often useless information; moreover, CAR applications often work in a resource-hungry environment, where minimisation of use of storage and other resources is crucial. Effectiveness is concerned with retrieving relevant documents, i.e. documents that the user will find useful. A significant amount of the focus of IR/IF research has been concerned with deriving weighting schemes to rank documents most accurately according to their apparent relevance to the user's needs. Again, this carries over to CAR, since it is a requirement of any form of information delivery: indeed accuracy of information delivery is likely to be of more importance in CAR where display space is often limited, and the CAR application may be peripheral to the user's current task. In particular, empirical evidence suggests that if a CAR user keeps receiving information of doubtful relevance they soon abandon the system [2]. Thus, overall we feel that IR/IF technology is strongly related to CAR, and that there are many potential gains and insights from relating them.

As an aside, it is a (sad?) current phenomenon that the world of IR/IF and the world of databases are usually seen as separate. Our work relates to the former world, but some CAR applications choose to use a database approach, especially if all the data are well-structured and uniform, and if there is no requirement for a weighting score to measure quality of match. Other CAR applications use a hybrid approach, with a database as the underlying storage layer, but with an IR/IF front-end (as in the "fuzzy-query" database used by Rhodes [3]), and our work does relate to these.

We believe that as context-aware computing develops and deals with increasingly large amounts of information and increasingly rich contexts, context-aware retrieval, which most applications need, will need to develop correspondingly. So far, with relatively small amounts of data, CAR has not been a focus of attention, but we believe the time is now ripe to try to understand this area properly, and our paper aims to be an initial contribution.

## 2. Terminology

In this paper we will generally follow IR/IF terminology. We assume that the information is a *collection* of discrete *documents*. Each document may be sub-divided into *fields*. These fields may be textual, such as title, author, keywords, full text of paper – the focus of much IR/IF research. Alternatively, they may represent data types that relate to the document or accompanying metadata, e.g. numbers, locations, dates, images, sets (say, a set of people detected as being in the same room by means of active badge sensors). Numerical fields in CAR are often ranges of values rather than single ones, e.g. a temperature in the range 0 to 10, and a numerical location within a certain circle or rectangle. The retrieval needs are captured in a *query* (in some situations referred to as a *profile*, see later): the retrieval task is to deliver the documents that best match the current query; each retrieved document may be accompanied with a score that gives a weighting of how well it matches.

We shall assume that the user's current context, like the documents, consists of a set of discrete fields. A field may be set directly by a sensor or, alternatively, fields may be set:

- directly by the user; for example the user may set a field to represent their current interests, or they may override a field that is normally set directly by a sensor, either because the sensor is faulty or because they want to pretend a different value
- by automatically synthesising data from sensors: for example, a field may be set to say whether a user is currently in a meeting, and this might be derived from the values of a lot of sensors in their office environment (as used in *reactive environments* [4] or in Pepys [5], though Pepys works after the event rather than in real time). A special case of this involves recording the user's computer activity; for example, a field may be set to record the content of the web document the user is currently viewing, or the notes they are currently making. Here the underlying "sensors" are recording the user's interaction with the computer and the computer's responses.

Potentially, the current context may consist of a large and diverse set of fields. Since it has the same overall form as a document in the collection – a set of fields – it can be treated, if required, as a document like any other. As we shall see, we may want to use the current context in two possible ways:

- to derive a retrieval query in order to extract information for the document collection
- to treat the current context as if it were a document, and to use this as the *target* of retrieval queries derived from the documents in the collection.

There are potentially three parties involved in a CAR application:

1. the *author* of the document collection, i.e. the information provider
2. the *user* whose context is being recorded
3. the *information recipient* of the retrieved documents.

In IR/IF the user and the information recipient are almost invariably one and the same, but in CAR this is often not so. Whether the two parties are different does not, however, greatly affect the retrieval process: it just affects where the retrieved documents are ultimately delivered.

Each of the three above parties might be single or multiple. Moreover, even if an application is regarded as a single-user system, it will gain from having as rich a context as possible for that user, and this rich context will often involve the context of other objects (people, vehicles, farm animals, mobile office equipment, etc.) that are relevant to that user.

## 3. Types of CAR Application

Many different kinds of application are described as "context-aware". It is useful to clarify the different types of CAR application. In this section we illustrate different types of CAR application by means of a series of examples. For simplicity we assume the user and the information recipient are the same. Specific examples of applications and the way they interact with the user can be found in [6] – arguably the paper that founded the discipline of context-aware applications.

A CAR application may be *interactive*, where the user directly issues a request to retrieve relevant documents, or (more usually) *proactive*, where documents are presented to the user automatically. We will start with the simplest interactive case, taking the example of a location-aware application; obviously real applications have a much richer context than just location, but the simplest example is often the best. We first assume that there are no automatic sensors to determine current location, but instead the user must specify their location (e.g. by typing it in or by pointing to a location on a map), and then interactively hit a *Retrieve* button; the system then retrieves the documents that match this location. This scenario just about qualifies as a CAR application, but we can extend the example and move more squarely into CAR by having the location set automatically, say by a GPS sensor. The retrieval mechanism itself is the same as before. The same query is used, and thus the same retrieved output is generated, but the query is created automatically using the location recorded by the GPS sensor. We can further extend the example to become proactive rather than interactive by causing the retrieval requests to be sent automatically instead of requiring the user to hit a button.

The criteria for performing a proactive retrieval request are normally based on change, either in the user's current context or in the documents in the collection. Thus a retrieval request may be issued automatically after (a) every N second period, (b) whenever the user's location has changed by amount M, (c) whenever the content of the document collection has changed, or (d) any combination of (a) to (c). We are now deep into CAR, but in terms of retrieval nothing has changed: as in all previous cases, the retrieval query is derived from the current context; we call this *user-driven*.

As a completely different example, the matching of context can be turned back to front: each document in the collection has a trigger context. A trigger context might, for example, be a certain location, or a certain location plus a time-of-day; if the trigger context on a document matches the user's current context, then the document is retrieved. For example, a document representing a tourist site might contain four fields: a location field together with a time-of-day field (e.g. opening hours) representing a triggering condition, plus a title field together with an information field that take no part in the matching operations but which are delivered as part of the document if the document is matched. The current context here acts as the target document to be matched, while the "document" in the collection is only used to form a query. This form of retrieval is attractive if the conditions necessary for retrieval are most naturally set by the information provider. It will normally be used proactively. We call it *author-driven retrieval* to distinguish it from the user-driven approach we have previously outlined, and we call the triggering condition on a document its *trigger context*. The more contextual fields the user has, the more attractive author-driven retrieval becomes. This relates to the quip about IR: "you can only find it if you know it is there in the first place". Thus, if we assume there are 20 sensors, then it is hard to create automatically a user-driven query that involves the values of some or all of these sensors and that finds the documents which really match the user's needs. Instead, it might be better to drive retrieval from the author side: the author knows what information is there, and is better placed to design the query.

In all cases the CAR system might either be a personal one, tied to a single user (who is also the recipient), or a shared system. In a shared system the users may be independent of each other, in which case the system behaves for each user in the same way as would a personal system, or the users might be interdependent. An example of the latter is where users are employees of the same company, and certain documents, when triggered, are sent to the most senior person present in a room.

In summary, we have two possible types of CAR: one driven by requests based on the user's current context, one driven by the information provider. In practice, an application might combine both sorts of retrieval; for example, the documents that are triggered by author-driven retrieval might be filtered by a second, user-driven, pass, or the user might manually override an author-driven retrieval application. This combination of retrieval types is particularly easy to do if there is a symmetry in form between the documents and the current context, i.e. the current context is a document like any other.

## 4. Information Retrieval and Information Filtering

We now move on to the relation of CAR to other retrieval technologies. As introduced in Section 1, the two main existing paradigms for information delivery for unstructured data are information retrieval (IR) and information filtering (IF). In this section we give an introduction to relevant details of these concepts. Both IR and IF are concerned with the finding of information, often in the form of text documents, which is in some sense about a topic that a user is interested in. Thus we can say that both are concerned with satisfying the user's underlying information need. As outlined in Section 2, users typically express their information need as some form of search query, which is then matched against the available documents. Although for IR and IF the search queries are traditionally mainly concerned with textual information, they could also involve numerical information such as dates. Those documents "matching" the query are delivered to the user. The user then inspects the documents to extract information relevant to their need. Thus IR and IF are concerned only with the delivery of documents, in the manner of a web search engine, and not delivery of the information itself.

Early IR and IF systems used Boolean queries and retrieved those documents that exactly matched each query. Most modern systems use queries that are in a form more natural for users to create, perhaps near to natural language, and use best-match retrieval: a matching score is computed between the search query and the document, and returned documents are presented to the user in descending order of matching score. The syntax of queries is not of great concern to us here, since in CAR the queries are often automatically derived from other information, such as the current context, rather than written directly by the user; we are, however, strongly concerned with the type of matching used, and we note that best-match methods have generally proved best in meeting the information needs of the majority of users who are not skilled in the use of Boolean logic to construct or interpret queries.

Regardless of the query type used, there are many parallels between IR and IF. In particular, both IR and IF are designed to handle unstructured or semi-structured data. (CAR data are, as we have said, most often semi-structured.) The sections that follow give a simple explanation of the basic features of IR and IF and highlight relevant differences; a much more detailed analysis is contained in [7].

### 4.1. Information retrieval

At one time, IR systems were almost exclusively the domain of the specialist librarian. The advent of the World Wide Web has changed this situation radically, and many people are now familiar with the use of IR systems in the form of web search engines. The user's search query to an IR system has an exact parallel with those CAR systems that we have called user-driven. The retrieval engine replies to the request by returning a set of potentially relevant documents to the user.

Each matching score gives a weighting of how well the document matches the query. In CAR systems these matching scores are even more important: as well as being useful for ranking, they can be used in deciding whether to deliver any documents at all. For example, a proactive system may decide that, since the best matching document still has a rather low score, it is not sensible to distract the recipient with it; thus nothing is delivered. The problem of setting a cut off criteria is an ongoing research issue in IR [8]. In practice, thresholds often have to be empirically tuned for individual applications. This problem is often less important for standard IR systems, which usually return something, even if of doubtful relevance: the user is left to decide whether the returned documents contain any useful information.

Most real-world applications involve retrieval from a huge number of possible documents and, unless some optimisations are made, there will soon be performance problems. Thus much research has been devoted to such optimisations: the basic strategy is normally to take those parts of the data that are relatively static and to preprocess these parts so that the retrieval engine can do its matching more quickly. One such strategy used in information retrieval is that texts are pre-processed to produce representations of the text, which are then organised into a database of text surrogates. This process is referred to as *indexing* and often involves:

- the elimination of frequently-occurring words such as function words, e.g. prepositions, which have little utility for matching documents and queries but greatly increase the size of the index file
- the conflation of words to a root form (referred to as search *terms*) to encourage matching between different word forms.

For best-match retrieval, performance can be improved by the application of term-weighting strategies that give higher weights to terms which are more likely to be important in retrieval of relevant documents. These weights are statistical in nature and rely on evaluating the distribution of terms within individual documents and across the whole document collection. Term weighting methods for information retrieval are described in detail in [9] and [10]. These techniques are equally useful in processing textual fields in CAR, provided, of course, that the data are relatively static, and the overhead of preprocessing can be repaid by enough gains in speed of matching before the data has changed and it is time to preprocess it again.

To summarise, the following features characterise information retrieval systems:

1. There are large numbers of documents, and these are to be matched against one query.
2. The document collection is usually assumed to be relatively static.
3. The document file structure is optimised to give rapid response to individual queries. The most popular way of doing this is by using an *inverted file* structure [11].
4. Queries usually describe transient information needs.

## 4.2. Information filtering

Information filtering (IF) systems are aimed at relatively stable, long-term information needs, although IF systems usually allow these interests to be modified gradually over time as conditions, goals and knowledge change. In this environment, rather than actively searching collections, users are often more passive, waiting for individual documents to be brought proactively to their attention. The user's interests are again represented by queries that describe their information need, but here these queries are referred to as *profiles*. IF systems typically apply the same text preprocessing strategies as IR systems to improve efficiency and reliability of matching between profiles and documents. An example of IF is an environment where a user wants to be kept informed of new papers that relate to their interests: they therefore create a profile that expresses these needs, and the system continually checks whether new documents that it receives match the needs. Those that do are sent to the user. The needs are often described in much more detail than in an IR search query, and will have been developed over time to match the user's interests in detail.

Most IF systems have a (large) number of users, and a system will maintain a number of active profiles. Individual profiles may be shared among more than one user where there are common interests. Whenever a new document arrives it is compared to each profile and delivered to those users whose profile is matched.

Given a particular document from the incoming document stream and a set of profiles, an important issue here is to consider what exactly does it mean to "filter" the document. In the case of Boolean matching we are able to say simply that it either matches the profile or it does not. For the best-match case the situation is rather more complicated. The best-match matching score gives a probability that a given profile is true for the incoming document. In the case of retrieval this probability is simply used to rank the documents. This situation would also apply for IF if documents were batched together and presented to each user in a ranked order. This might be possible in some applications, but in many filtering applications a decision must be made on document relevance as each document appears. Two options are:

- the document could be directed to the owners of the top ranking profiles. In this case the users are not independent, but the document is sent to the people who most need it – this might be a company's employees to which the document appeared to be most relevant
- a threshold could be set to define the minimum score necessary to count as a match, though a suitable threshold can be difficult to determine. There are parallels in a CAR system where the recipient only wants to get information that matches the current context very closely.

In IR, parameters such as term weights are estimated from the static document collection. For IF these parameters must be estimated from large samples of previously-seen documents; new documents are thus assumed to have similar characteristics to those seen previously. IF systems raise their own issues of efficiency. A single document is compared in parallel to a potentially very large number of profiles. Efficiency can be achieved here by using an inverted file of the search profiles [7].

To summarise:

- IF is mainly concerned with selection or elimination of documents from a dynamic datastream.
- Normally one document is taken at a time, but a large number of different queries (profiles) are applied to it.
- IF is concerned with the delivery of documents to individuals whose profiles are matched.

## 4.3. Document structure

For both IR and IF, the simplest approach to matching queries/profiles with documents is to treat the whole document as a single object. However, when the document is divided into distinct fields, it is straightforward to take these into account in the matching process if desired. This relates to CAR retrieval, where documents often have many separate fields.

The existence of multiple fields may lead to a multi-stage retrieval process. For example, Boolean queries could be applied to any appropriate field and potentially used as a filter to remove documents from further consideration. Best-match queries may be applied separately to individual fields and matching scores associated with the individual fields summed in some way. Alternatively, multifield matching may be used. Multifield matching is relevant to many CAR fieldwork applications, as illustrated by the following example of work from a member of our group [12]. This CAR application is concerned, *inter alia*, with identifying rhino footprints. When the user (who is typically out in a game reserve) has found a footprint, the current context will consist of various fields representing dimensions and other properties of the footprint (these fields could potentially be derived from a sensor that was a camera), together with the user's current location. The CAR application then wishes to retrieve, in ranked order, documents representing previous sightings, the aim being to find footprints that most likely belong to the same rhino as the present one. The matching algorithm, however, needs to be cleverer than just accumulating scores for how well each field matches; it needs to do multifield calculations involving, for example, the ratio of two field values, in order to decide what is the best match. One way of approaching this is to allow a field to be an aggregate of sub-fields. Thus a rhino footprint could be one such aggregate field, and it could have its own matching algorithm, based on the values of its sub-fields. Overall we see that structuring of documents in CAR can be stronger than is typical in IR/IF, though this structuring will probably be less complete than for a database application.

## 4.4. Relationship between CAR and conventional IR and IF

If, in CAR, we assume there is a single user, and thus a single current context, and if we ignore any trigger contexts associated with the documents, we have a situation similar to conventional IR, where a single query is able to retrieve any document in the collection. Thus, obviously, *what we have called user-driven retrieval is simple conventional IR* where the query is automatically constructed from the present context. In the simplest case, the query can just represent each field of the current context, e.g. if the current context consists of a location L and a temperature T, then the query may ask for documents that (best) match a location field of L and a temperature field of T. In general, the query may be constructed from a template where the values of various fields are inserted.

If we then look at author-driven retrieval, and consider trigger contexts attached to documents, we have a situation similar to conventional IF where some form of filter is applied to retrieve the documents of potential interest; the only difference is that in CAR:

1. the profile (i.e. trigger context) is associated with the document, not with the user
2. the current context is treated as a document which is the target of retrieval. Thus we have a single, but constantly changing, target document instead of, as in IF, a stream of different target documents.

Thus *author-driven retrieval is essentially identical to information filtering*.

We will now move on to two more elaborate scenarios. Firstly, as we have said, a CAR application might be part user-driven and part author-driven. For example, the tourist might be using author-driven triggering, but might wish to apply a further user-driven retrieval stage to the retrieved documents: he might only want to see documents that concern architecture, or documents that have a field concerned with temperature. We can regard these stages as two separate activities, IF and then IR, but it is a research question as to what efficiency gains can be made from combining the two stages.

Secondly, there may be multiple users being served by author-driven retrieval, as there would be in an author-driven tracking system (e.g. a document might have a trigger that matched if any of the tracked "users", racehorses say, were in a certain area). In some respects we now have a hybrid situation with multiple queries, which is characteristic of IF, and multiple documents (i.e. multiple current contexts to be matched against), which is characteristic of IR.

Thus many CAR application environments represent novel retrieval environments in which techniques from conventional IR and IF may offer effective solutions to retrieval problems, but which also pose new retrieval questions requiring further research.

### 4.5. Summary of approaches to information delivery

To summarise the overall position, Table 1 shows the typical qualities of the forms of retrieval we have considered.

## 5. Issues for CAR Applications

We will now consider some issues in CAR that either do not arise at all in IR and IF, or only arise weakly. These issues are concerned with change and its partner, history. We will consider history first, since it is the simpler. There are two issues:

1. It is useful if the application remembers which documents have already been retrieved and presented to the user. Depending on the application, this historical information can be used: (a) to prevent the same information continually being presented to the user; (b) to indicate to the user that information has been viewed previously; or (c) to indicate that a previously-viewed document has changed.

**Table 1.** The different types of retrieval in their basic forms.

| Retrieval type | Request | Retrieved from | Output |
|---|---|---|---|
| IR | query from single user | document collection | matching documents |
| IF | queries (profiles) from many users | single document | document + set of profiles it matches |
| user-driven CAR | queries derived from users' current contexts, one query for each user | document collection | matching pairs: document/user's current context |
| author-driven CAR | each document in the collection may incorporate a query | documents derived from users' current contexts, one document for each user | matching pairs: user's current context/document |

2. The application should remember past values of each field of the present context, and it is especially important to do this between one retrieval operation and the next. This information can be used for prediction and interpolation – see later examples.

We now move on to discuss change. The first point to make is that for most CAR applications, the current context is continually changing and for many applications the collection of documents is changing too (e.g. for traffic information). IR is geared to static collections of documents, and IF to static queries; the case where everything is dynamic has been referred to as the "grand challenge" [13].

However, in most CAR applications based on mobile users, change in the current context is *gradual*. Indeed, if the user is moving, she will think of herself as moving continuously, and there is a mismatch between this world and the world inside the computer, with change in discrete steps. Even a static user may have a time field in their current context which they will perceive as advancing continuously, but in the computer may be updated every few microseconds. We have said that we would like to perform a new retrieval operation every time there is change, and we might like to reinforce the user's perception that change is continually monitored, but we need to temper this to match the speed of our retrieval engine and the costs associated with it. Such issues do not arise with conventional IR and IF, where change is much less frequent. Sometimes in CAR, when there has been a sudden radical change, we might want to abort any current retrieval operation and start a new one. In some key applications, such as triggering when certain share price thresholds are passed, speed of retrieval may have absolutely vital importance, and be the driving force of the application.

A further facet of the mismatch between what the user sees as continuous change and what the system sees as discrete change is the possibility of missed information. An example is the following: a document is to trigger when the temperature is 0; one retrieval is made when the temperature is 1; the temperature is dropping fast, and by the time the next retrieval is made, the temperature is –2. The result may be that the document is not triggered. In our experience, however, such problems can usually be solved by using ranges rather than single values: thus the author might set the trigger context as a temperature in the range –2 to 2, rather than 0; alternatively the application, when setting the current context prior to retrieval, can set each field to the range of values that has occurred since the last retrieval – this assumes, of course, it is keeping some history: in our example this interpolation would yield the range between 1 and –2, and would match the original document, even if it specified a value of 0. Overall this interpolation is a simple example of more general inferences that may be made from contextual data.

In other applications one might wish to take the rate of change into account in deciding which documents are important. For example, in a system designed to automatically assist drivers in avoiding traffic congestion it may be useful to provide information with differing degrees of granularity depending on the driver's current rate of progress. There is little point in suggesting routes to avoid problems further into the journey if the driver is enmeshed in local congestion problems.

Finally, there is an important positive quality of change: if change is gradual and consistent, it may be possible to exploit this to perform forecasting and optimisation. We discuss this later.

## 6. Presentation to User

This is not an HCI paper – though HCI issues are of course crucial in CAR applications as in most mobile applications. Nevertheless, it is foolish to consider retrieval in isolation, without any thought for how the recipient can exploit it.

Overall we want to be sure that there is at least one credible model for presenting information to the recipient. There is indeed: an approach similar to that used for presenting incoming email is one that is already familiar to most users and need not be intrusive. Individual applications may well, however, have their own user interfaces that are especially tailored to the task in hand. Two interesting ones – both potentially adaptable to a range of applications – are described in papers by Rhodes [2,3]. The first of these describes the "Wearable Remembrance Agent", the output of which is viewed using a head-up display; each retrieved document is initially represented by a single line of text at the bottom of the display. The second system, "Marginal Notes", assumes the user interface is a web browser, and augments the web page the user is reading by adding additional suggested links in a margin beside the main page; these added links are determined by the current context.

A more general point concerns the information that may need to be made available to the recipient: when document D has matched current context C, the recipient may wish to look both at material extracted from D and material extracted from C – remember that the recipient may be different from the user and may wish to know about the context C of the tracked user. The technology needed to extract and suitably present this information to the user is an area for further study in HCI for CAR. Rhodes [2] advocates the use of "ramping interfaces", whereby information is provided at increasing layers of detail: "the idea is to get useful information to a user quickly, while at the same time allowing them to bail out [of unwanted retrieval] with as little distraction as possible".

In proactive retrieval, the issue of controlled change impacts the user interface. At one extreme the user interface might consist of a single template into which some value from the most recently received document is inserted. For example, in a CAR application only concerned with locating restaurants the display might consist of a template of the form "The nearest restaurant that meets your needs is currently X". Whenever a new document is triggered its "restaurant name" field can be extracted and filled into the template in place of X. The user perceives a continually changing world and the quicker the change is made by the CAR application the better. At another extreme, an application may retrieve lengthy descriptive information for the user to read; the user would be upset if, when she was half way through reading such a document, it was suddenly overridden by a newly retrieved one. In this case user requirements for discrete steps and graceful change may dampen the need for frequent retrieval.

## 7. Implementation

CAR is an emerging discipline, and most current systems have labels such as "pilot" or "prototype". Most have limited amounts of data, and have not yet had to face retrieval performance problems. Thus they have been able to make do with simplistic retrieval approaches.

Our own practical experience mirrors this [14], and the main limitations on performance have been the setting of the current context, in particular:

- the slowness of sensors and the problems of communicating with them
- the slowness of "clever" sensors, specifically software sensors that intelligently try to combine physical sensors into information at a higher level of abstraction, e.g. that the user is in a meeting.

We believe, however, that in the future, with such applications as context-aware information provided as part of a standard mobile phone service, efficiency of retrieval will become the central issue. Indeed, recent discussions with colleagues concerned with a context-aware exhibition guide show that serious problems can emerge even with only a thousand documents and a few hundred independent users.

## 8. Performance Optimisation

Performance optimisation, together with weighting of matches, is the essence of IR/IF. If these two issues were ignored, much retrieval work would be almost trivial to implement (and, indeed, trivial in its usefulness). As we have said, the main strategy of optimisation is to take advantage of the static nature of some element of the matching, and to do some preprocessing to make that element faster to process. We have a number of approaches for handling large volumes of data, as outlined earlier. There are also techniques for optimising the match of 1D, 2D, and multiple-dimensional numerical values, e.g. quadtrees or R-trees. All of this potentially carries over to CAR in cases where the relevant fields are static. If the CAR application involves a multiplicity of data types, each requiring its own form of optimisation, then the total optimisation task may represent a big challenge. Nevertheless, it is a challenge that must be faced because, when CAR applications become more widespread, there will be massive amounts of data, but still the requirement for good real-time performance.

The most promising route to progress is not to focus on the areas where CAR is harder than normal IR/IF, but instead to look at the special characteristics of CAR that can be exploited: the most obvious of these is that change in the current context may be gradual and, to a degree, predictable. Several optimisations, based not on unchanging static data but on gradual change, then become possible:

- The query may be a constant template, with slightly different values plugged in for each retrieval.
- If a history is maintained, the application may be able to extrapolate in order to forecast future values of some fields ("the user is travelling at 3 mph in a certain direction; therefore when we next need to retrieve the location is likely to be X"). This allows retrieval to be performed in advance, although of course there is a danger of a forecast being upset by a change of direction. The issues here are similar to those encountered in design of computer memory architectures, and also arise in tracking cellphone users [15].
- If the retrieved documents are ranked, and if the data have the property that a small change in the user's current context will produce a correspondingly small change in the retrieved documents and their rankings, then a possible strategy is as follows. First perform a full retrieval, and remember the 100 (say) best matches in a cache; show the best few of these to the recipient; for the next retrievals, just look at the 100 documents in the cache, calculate the new weightings, and again present the best few to the user; periodically do a full retrieval to reset the cache – in particular do this if there is a radical change in the present context. (This optimisation also clearly has potential for reducing network traffic, as well as for reducing retrieval time.)
- Use forecast values as an additional factor in setting matching scores: e.g. documents that relate to locations the user is moving towards weigh more than those behind or off to the side.

These examples cover just a few of the areas of possible further study for CAR research.

## 9. Conclusions

We have identified two basic types of CAR: user-driven and author-driven. These are logically equivalent to IR and IF respectively. Many CAR applications are hybrids between the two and, in addition, many have multiple documents and multiple queries, whereas IR/IF concentrate on one degree of multiplicity.

Given that CAR is so strongly related to both IR and IF, we believe that the two cornerstones of past – and present – IR/IF research, efficiency and effectiveness, will be equally crucial to CAR.

In order to improve efficiency of retrieval, the focus of IR/IF research has been on preprocessing static parts of the data. CAR applications are less static and we believe that a promising avenue for further research is not to focus optimisation on the static, but instead to focus it on change that is gradual and partly predictable.

We believe that in CAR, as in IR/IF, best-match technology will eventually dominate. Again if change is predictable, weightings can be useful in creating caches, and in deciding whether information is relevant enough for it to be worthwhile to interrupt the user's current activity.

## Acknowledgements

Several of our colleagues have contributed greatly to this work: we would especially like to mention Nick Ryan and Jason Pascoe. The paper was originally inspired by the clear thinking and superb insights shown in an MSc dissertation [16]. We are grateful to one of the referees for their particularly careful analysis, and consequent suggestions. Peter Brown appreciates the Leverhulme Trust's support.

## References

1. Brown PJ, Burleston W, Lamming M, Rahlff O-W, Romano G, Scholz J. Snowdon D. Context-awareness: some compelling applications. Submitted for publication, http://www.dcs.ex.ac.uk/~pjbrown/papers/acm.html, 2000
2. Rhodes BJ. Margin Notes: building a contextually aware associative memory. In: Proceedings of the Conference on Intelligent User Interfaces (IUI'00), New Orleans, LA, 2000
3. Rhodes BJ. The Wearable Remembrance Agent: a system for augmented memory. Personal Technologies 1997; 1: 218–224
4. Cooperstock J, Fels S, Buxton W, Smith KC. Reactive environments: throwing away your keyboard and mouse. Communications of the ACM 1997; 40: 65–73
5. Newman WM, Eldridge MA, Lamming MG. Pepys: generating autobiographies by automatic tracking. In: Proceedings ECSCW '91, Amsterdam, September 1991
6. Schilit WN, Adams NI, Want R. Context-aware computing applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa

Cruz, California, 1994. IEEE Computer Society Press: 85–90

7. Belkin NJ, Croft WB. Information filtering and information retrieval: two sides of the same coin? Communications of the ACM 1992; 35: 29–38
8. Robertson S, Walker S. Threshold setting in adaptive filtering. Journal of Documentation, 2000; 56, 3:312–331
9. Sparck Jones KS, Walker S, Robertson SE. A probabilistic model of information retrieval: development and status. Technical Report 446, Computer Laboratory, University of Cambridge, 1998
10. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing and Management 1988; 24: 513–523
11. van Rijsbergen CJ. Information retrieval. Butterworths, London, 1979
12. Pascoe J. Context-aware software. PhD thesis, University of Kent at Canterbury, 2001
13. Oard DW, Marchionini G. A conceptual framework for text filtering. Report EE-TR-96-25, University of Maryland, 1996
14. Brown PJ, Bovey JD, Chen X. Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications 1997; 4: 58–64
15. Das RE, Sen SK. Adaptive location prediction based on a hierarchical network model in a cellular mobile environment. Computer Journal 1999; 42: 474–486
16. Kennaugh RJ. An agent based information dissemination system. MSc dissertation, Imperial College, London, 1998

*Correspondence to:* Peter J. Brown, Department of Computer Science, University of Exeter, Exeter EX4 4PT, UK. Email: P.J. Brown@exeter.ac.uk