Exploiting Contextual Change in Context-Aware Retrieval

Peter J. Brown
Department of Computer Science
University of Exeter
Exeter EX4 4PT
United Kingdom
P.J.Brown@exeter.ac.uk

Gareth J. F. Jones
Department of Computer Science
University of Exeter
Exeter EX4 4PT
United Kingdom
G.J.F.Jones@exeter.ac.uk

ABSTRACT

Information retrieval systems are usually unaware of the context in which they are being used. We believe that exploiting context information to augment existing retrieval methods can lead to increased retrieval precision. This approach is particularly important with the development of wireless mobile information appliances, such as PDAs. Many of these devices are aware of the user's physical context, and this has led to the evolution of *context-aware applications*. Such applications can automatically utilise the user's current context, e.g. location or ambient temperature. Context-Aware Retrieval is related to traditional Information Retrieval and Information Filtering, but is potentially more challenging due to the often continuous changes in user context. To meet these challenges we suggest a potential advantage of Context-Aware Retrieval: this is that the current context is often changing gradually and semi-predictably. In this paper we suggest new methods based on a *context-diary* and *caching* aimed at improving both the precision of relevant retrieved information and the speed/availability of retrieval. The methods can be used, in principle, on top of existing retrieval systems.

1. INTRODUCTION

Information retrieval technology has continually grown to meet challenges presented by new forms of usage. One new and expanding application environment is that of wireless mobile devices such as PDAs. It is natural to suggest that users will want to retrieve information while using these networked mobile appliances. With mobile applications some aspects of the user's context, e.g. their location, are often available, and this context can affect what information is relevant to the user. Indeed there are existing applications in areas such as tourism where the user's current context may be the sole driver of what is retrieved.

We are interested in such context-aware retrieval of information. To be exact we are interested in retrieving information that is pertinent to the *user's* current context. As well as tourist applications, this is needed by context-aware applications such as general information aids for mobile users (e.g. exhibition visitors, motorists or inhabitants of Cooltown [7]). Other context-aware applications are useful on static computers as well as mobile ones, where the context may be the user's computing context, e.g. the document they are reading; examples of these applications are ones that supply reminders based on the user's current context (e.g. 'when you were last in a similar context, you accessed document XXX – which gives information about creating electronic links'). This class of application is more generally covered by standard relevance feedback and personalisation methods; we are are more interested in applications where additional context information beyond the textual document content is available.

As all these applications move from laboratory to marketplace, and deal with increased amounts of information and much richer contexts, we believe they can gain from developing retrieval methods that are better designed to incorporate context information into the retrieval process. A particular challenge of this technology comes from the observation that context is often changing continuously. Existing context-aware applications have only explored the use of small well structured datasets. New systems need to be more flexible to handle large amounts of unstructured data and be easily usable by a wide range of users. We draw a parallel with the overall discipline of Information Retrieval, which has moved on from the early Boolean systems to the best-match ranked retrieval systems of today; the latter are highly successful both in performance and in delivery of relevant information, and they can be applied to any sort of user need.

The main focus of this paper is how analysis of change in the current context can lead to improved retrieval precision and to better retrieval speed. Our emphasis throughout is on simple methods such as linear prediction. *Our emphasis is also on postulating ideas and questions*: we have built an experimental system to investigate these ideas, but it will be a while before we have definitive results – indeed part of the research challenge of our work is to design evaluation strategies and data sets. Obviously there is future scope for more elaborate approaches, perhaps based on AI algorithms, but we believe there are many unanswered questions even with simple approaches.

The structure of the remainder of the paper is as follows. Section 2 introduces the basic concepts of Context-Aware Retrieval; Section 3 then discusses the retrieval process. In Section 4 we discuss the importance of change in the cur-

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.
SAC 2002, Madrid Spain
Copyright 2002 ACM 1-58113-445-2/02/03 ...$5.00.

rent context, and in Section 5 how to keep a record of this in the form of a Context-Dairy. Section 6 introduces the idea of the Context-of-Interest, which may differ slightly from the actual current context, and describes how this difference can be used to improve performance. Continuing the performance theme, Section 7 outlines methods to improve retrieval speed and a caching technique, and Section 8 then gives some ideas on tuning the retrieval algorithms. Finally we outline our current experimental Context-Aware Retrieval framework in Section 9.

2. CONCEPTS OF CONTEXT-AWARE RETRIEVAL

2.1 Representing User Context

The basic components of Context-Aware Retrieval (CAR) are: a *document collection*, which contains the documents that may be retrieved annotated with details of their associated contexts, and the user's *current context*. The current context normally consists of a set of separate fields; typically most or all of these are set by sensors (e.g. GPS for Location), but the sensor-derived fields may be augmented by other fields set by the user or application (e.g. Interests or, more generally, fields representing a user profile). Fields can be of various data types, such as numbers, two-dimensional or three-dimensional locations, sets, texts, images, etc.

Often numerical values will be ranges rather than single points: for example a Location may be represented by a circle or rectangle, and a Temperature field by a range of temperatures. Fields may be structured, e.g. representing a Location field in a tree of levels of abstraction with 'My office' at a high level and the physical location at a lower level. Structured fields are used, for example, in MemoClip [1] and in CybreMinder [4] – both systems for supplying reminders when the fields of a user's context match some previous situation: the idea is that retrieving documents relating to the previous situation will help in the present one.

In this paper, however, we assume for simplicity that fields are unstructured name/value pairs. Thus a field may have the name 'Location' and a value giving a location. It is easy to represent most aspects of physical context as name/value pairs, but for more human-related aspects of context (e.g. mood, current activity) it may well be a research problem to derive the values – see Pepys [10], for how activity in the office (e.g. a meeting) was guessed from low-level sensors. Our interest is retrieval rather than representation/derivation of context, so we assume the name/value pairs have been constructed.

2.2 Retrieval Paradigms

CAR is related to the well established fields of Information Retrieval and Information Filtering. To make this relationship clearer we distinguish two CAR paradigms as follows:

- *proactive*: some or all of the documents in the document collection contain a triggering condition, and when this matches the current context the document is retrieved. This has parallels with Information Filtering; the triggering condition has the role of the profile, and the current context acts as the current document; when the current context changes, a new current document is derived and a new retrieval takes place. One difference with Information Filtering is that the triggering conditions (profiles) are specified by the provider of the document collection, not by the user. They apply to all users.
- *interactive*: the current context is used to derive a retrieval query which is applied to the document collection in the standard manner used in information retrieval. Retrieval may be initiated directly by the user or automatically by the application (e.g. whenever the current context has changed). The current context is converted into a standard IR query.

A crucial property of many context fields is that they are *continuous*: as the user's context changes new information may need to be retrieved. Such continuous applications normally require fast retrieval, so that the user has the illusion that new information arrives immediately there is any change in their context. This is absolutely different from the 'one-off' nature of traditional information retrieval requests, and presents many research challenges, especially given the constraints of high precision and fast performance.

Finally, an observation on the word 'context': this has a wide variety of interpretations. Some authors [5] have looked at the coupling of information retrieval and the context of the information itself (who wrote it and when, whether the information is 'official', whether it has been cited by other, respected, people). The information's context can be important to the user in evaluating the information, e.g. whether they believe it to be true. This type of context is not, however, our interest here: we concentrate on the *user's* context.

2.3 Retrieval Environment

Two key properties of retrieval are recall and precision. As Rhodes and Maes [13] have observed, in CAR *precision is generally the more important*, a key observation that influences many of the ideas presented in this paper. The main reason is that the user is typically mobile, and is involved in other tasks. When a retrieved document is brought to her attention this is an intrusion. This is especially true when the retrieval was not explicitly asked for by the user. It is less true in interactive retrieval initiated directly by the user, though even here, assuming the user is constrained by a small screen, they cannot easily browse through reams of information. Overall a useful maxim for the design of CAR applications is *assume each retrieval is an intrusion; therefore try to make sure that it is relevant.* Even if the information is relevant, it still needs to be presented to the user in a manner that does not interfere too much with their other activities. HCI issues are not a prime focus of this paper, though they are, of course, central to the success of an application. They can affect the way information is delivered (e.g. by audio rather than textually) and presented (e.g. via a 'ramping' interface that allows easy transition between levels of detail, and is suitable for a small screen). See [13] for more details of HCI issues.

3. CONTEXT-DOCUMENT MATCHING

We believe that a best-match rather than a Boolean approach is preferable in CAR for the same reasons as it is in many other retrieval applications. In particular the scores derived in best-match retrieval provide a building block for deciding the order in which items are presented to the user – or whether any items are presented at all; this can help achieve the goal of high precision. Existing CAR systems have concentrated on database approaches: we have not done this, partly because we want our returned documents to be ranked, and partly because, as contexts become richer and involve more fields, information becomes less structured. Moreover they often contain textual fields for which retrieval methods are better suited.

Our CAR research uses ideas from *stick-e notes* [2]; stick-e notes are like ordinary paper Post-it notes, except that they are attached to an electronic context rather than stuck in a certain physical location. This electronic context can be a rich one covering any or all of such contextual fields as location, time, temperature, orientation, text of an electronic document currently being read or composed, state of equipment, camera image, ... Stick-e notes, like many other CAR systems, use Boolean retrieval. Our new system uses best-match retrieval. It is not the first CAR system to do so. Best-match technology is employed in Savant, which is used for Just-in-time retrieval agents [13]. Savant is designed to work on existing multi-field document collections, such as archives of e-mail or news stories. The purpose of the new ideas presented in this paper is that they could be applied to any existing best-match CAR system, including Savant. Thus our interest is not in establishing a new approach to retrieval, but rather finding augmentations that are apposite for CAR.

A characteristic of ordinary best-match information retrieval is that, provided there is not a complete mismatch, the application always delivers some documents, even if all the retrieved documents have low scores. The assumption is often made that the user would rather have something than nothing at all. In CAR, on the other hand, where retrieval can be an intrusion, the application may well decide to ignore all the results of a retrieval if the retrieved documents have low scores. Best-match retrieval potentially provides the wherewithal to do this, though picking thresholds is always a hard problem [15]; it is our hope that augmenting the retrieval process with contextual information can make this process more reliable.

3.1 Matching with Multiple Context Fields

As we have said, a characteristic of CAR is that the current context is typically divided into separate fields (e.g. the retrieval process needs to match the user's Location field, their Temperature field, their Current-need field, etc.). The documents in the collection are, we assume, similarly divided into fields, and each document has the same overall format as a current context. For example a document about a tourist site might have a (large) textual field giving a description of the site, together with other fields giving the context in which the site would be of interest (its location, its opening hours, etc.). In theory some of the contextual fields could be derived automatically from the document content (e.g. if the document is about The Tower of London its Location field could be set to the location of this London landmark), but at present our fields are set explicitly. (The symmetry between current context and document has benefits, as we shall see later: for example, in a 'memory aid' application, a set of previous current contexts could be placed in a document collection, and used as an object for retrieval. This allows users to retrieve past contexts similar to their present one.) A retrieval operation, when matching a document against the current context, involves matching individual fields and then combining the results to obtain an overall score for the match. Only a subset of the fields may be 'active' in the sense that they are involved in the retrieval. (In essence CAR has similarities to traditional bibliographic retrieval, where documents have fields such as 'Title', 'Author', 'Date', 'Abstract', etc.). The following factors – which we discuss in more detail later – are pertinent to an overall score for CAR:

(1) The algorithm for calculating a score to measure how well two fields (one in the query derived from the user's context, and one derived from the document) match. Such algorithms will usually depend on the nature of the fields: an algorithm for matching two locations will be different to an algorithm for matching two dates, which in turn will be different to an algorithm for matching two textual fields. Matching of textual fields has been extensively studied, and we can use standard best-match term weighting strategies, such as those adopted in the Okapi system [14] or the SMART system [16]; matching of other field types is much more open to experiment.

(2) The relative weights of fields; for example a Location field may have twice the weight of a Temperature field. (A weight is called a *bias* in [13]). Determining such weights for optimal retrieval performance in individual applications is very much an open research question; one issue is the amount of context evidence, including not just its current value, but how it may be changing.

(3) An algorithm for combining (1) and (2) above into an overall score. Often this algorithm computes a weighted arithmetic or geometric mean.

All of the above three offer scope for tuning to improve precision. Our experimental system allows any of the three to be changed dynamically, i.e. between one retrieval and the next. Thus an application could, for example, increase the relative weight of a field that was changing fast – we mention later this example, which is also covered in [13]. (We think it relatively unlikely that the application would want to change (3) above dynamically, e.g. to suddenly switch from an arithmetic mean to a geometric one, but it is possible if necessary.) We refer to the above three the *retrieval plug-ins* of our experimental system.

The focus of our work has so far been on users who are independent of each other rather than collaborating in their retrieval. In collaborative systems further types of weighting are possible: e.g. weightings that encourage a user towards (or even away from) another colleague whose location is known. Issues relating to collaborative systems are not explored further in this paper.

4. THE NATURE OF CONTEXT CHANGE

In our previous work [3] we analyzed the relationship between, on the one hand, CAR, and, on the other hand, traditional IR (Information Retrieval) and IF (Information Filtering). One of our conclusions is that CAR has elements of both IR and IF, but is potentially harder than either of them: (a) because the current context is changing, often continuously, and the document collection may be changing too, and (b) because CAR often needs near-continuous and fast retrieval.

To counter all the potential difficulties of CAR, it is important to try to find something that can give CAR an advantage. We believe that one potential advantage is that the current context is usually changing gradually and semi-predictably. This potential advantage is a focus in the rest of this paper.

In some applications there will be fields of the current context that do not change gradually and predictably. This will apply especially to fields set by the user: their Interest field may, for example, go from documents whose Content field relates to architecture to documents about transport systems. (Perhaps, however, even in this case a clever AI program would have some success at prediction.) Paradoxically when a predicted change does *not* occur, this in itself can be a significant event that could cause extra weight to be given to the changed field (e.g. the temperature suddenly dropping in an industrial context). In our case, following our keep-it-simple aim, we only set out to exploit fields that are easy to analyze and predict.

There are, of course, other aspects of change in addition to change in the current context. For instance there may be change in the document collection and/or change resulting from adaptive feedback from the user concerning the relevance of previously-delivered documents. Such aspects of change have already been studied in conventional IR and IF – though some aspects of feedback may relate specifically to context. Change in the current context, which is germane to CAR, is much more an open field, and that is why we concentrate on it here. In order to make use of change in context we need to keep a record of previous contexts and expected future contexts; to achieve this we explore, in the next Section, our proposal of the *Context Diary*.

5. THE CONTEXT DIARY

If we wish to analyze change in the current context we need to maintain a history of it. Clearly the more wide-ranging the history, in terms of users and events covered, the more information is available to adapt future CAR behaviour. On the other hand the more useful this information is, the more it is a danger to personal privacy, particularly in applications covering interacting groups of people. We will assume here a compromise position where the history relates to a single user, and has access controls that are acceptable to that user.

History can be maintained automatically by the application; for example a record of the current context may be added to the history:

- every N minutes,
- whenever the current context changes by more than a certain threshold amount,
- whenever any retrieval request occurs,
- whenever user feedback indicates that a particular current context was important.

History may be confined to those fields that are easy to analyze and predict; for example, location and temperature.

Interestingly, history can sometimes be generalised to include the future as well as the past. For example, if the user's diary says he is planning to be at a meeting at a certain location in two hour's time, then this can be recorded as a 'future' item in the history of their Location field, and may well turn out to be valuable in making intermediate predictions. Thus, in this paper from now on we will not use the word 'history' to describe the record of contextual values: instead we will use the term *Context Diary*. The Context Diary can cover both past and future (though, because the past is certain and the future is not, the detailed mechanisms for storing future events may have some special properties, e.g. their probability of occurrence, which may itself be estimated from past experiences). The Context Diary may be derived from an existing calendar/diary system and/or from an electronic notification system such as Khronika [9] and its successors; such systems are most valuable when they supply multiple contextual fields, such as a time and a place. The Context Diary can also derive information from weather forecasts, e.g. a likely temperature at a certain location at a certain future time. Of course as time goes by, future diary events can become past diary events – but perhaps only if they are detected as really happening, e.g. that sensor values indicated that a user really did attend a scheduled meeting or alternatively that the meeting was missed, which is itself a contextual element that may have implications for the future.

In our current experiments the Context Diary, as the above discussion implies, is indexed by time, but it could be indexed by any contextual field, e.g. Location. Moreover it could be based on a concept such as user trails, which are a combination of Location and Time fields.

The Context Diary has three potential uses: (a) to detect change – clearly if we want to detect when a current value changes we need to know its past value(s); (b) to predict; (c) as an aid to the retrieval process (e.g. to eliminate 'been there, seen it' items). More subtly the Context Diary can be used as the document collection to be retrieved from (each entry in the diary counting as a document); this can enable the user to retrieve contexts similar to their present one, in the manner of a memory prosthesis [8].

6. THE CONTEXT-OF-INTEREST

In many CAR applications, particularly mobile ones, the user may often not in fact be interested in information relating to their current context: instead they are likely to be interested in a context 'just ahead'. This is an example of what we refer to from now on as the *context-of-interest*. For example the context-of-interest of a traveller or tourist might be set with the aim that they retrieve information just before they need it. The Context Diary can be used, together with the current context, to predict the context-of-interest. This predicted context-of-interest is then passed to the retrieval system in place of the true current context, with the aim of retrieving documents that are more relevant to the user's needs at the time of delivery.

The following are examples of how fields within the context-of-interest may be set:

- a Location field may be set to a point (or, more likely, a range of values) ahead of the user's current location, taking into account their direction of travel, since the user is more likely to be interested in sites ahead of rather than behind them.
- the Time field can be set to a value somewhat in the future. In one sense the Time field is an extreme because (a) its advance is totally predictable, and (b) there is

a sharp cut-off in usefulness between information relating to past time and future time. As an example of (b) assume a document relates to a museum and has a Time field with value 9.00am to 5.00pm representing its opening hours; if the current time is 4.59pm, this will match the document's Time field; however the user would be more interested in a building that was *opening* at 5.00pm. Thus their context-of-interest is ahead of the current time.

- a field relating to outside temperature would be useful in determining whether to deliver information about an open-air café. In most countries temperature is fairly predictable, and the most relevant temperature for information delivery is likely to be the temperature in, say, half an hour's time, or even the predicted average temperature over the next two hours. This information might therefore be used in the context-of-interest to form part of the decision about whether to return information about the open-air café. A prediction can be based on the rate of rise/fall of temperature (derived from analysing the Context Diary), the current temperature (as detected by a sensor), and perhaps the future temperature given by a weather forecast that has been incorporated in the Context Diary.

Of course prediction can be wrong, especially, for the above examples, in the evaluation of a future Location value. Sometimes a bad prediction will lead to the retrieval of information that is *less* relevant than if there had been no attempt at prediction. Thus a goal of our research is to find out how much the gains counterbalance the losses. So far the indications are that modest predictions – taking a small rather than a large leap into the future – are winners, at least in tourist applications.

Finally the context-of-interest can be used to improve the setting of the current context. Some sensors give occasional totally wrong values, and others periodically fail to work (e.g. GPS in a tunnel). Prediction can be used for checking and to smooth over difficult periods; the result should be an improvement in the relevance of delivered documents, and an elimination of some irrelevant ones.

7. APPLICATION OF THE CONTEXT-OF-INTEREST

The context-of-interest also has value in improving performance when: (1) retrieval is slow, or (2) connection to the document collection is only periodically available.

7.1 Compensating for Slow Retrieval

The time taken to complete the retrieval process is often a key parameter in user assessment of search engines. Retrieval speed is thus an important concern in the development of CAR technologies. If retrieval takes an average of two minutes because of limited processing or communication resources, then, even if the user wants information about their current context, they will want it to be about their predicted current context two minutes after the retrieval request. Thus the latter should be supplied to the retrieval engine. (This example, incidentally, offers a partial escape if a prediction turns out to be wrong: if the user suddenly changes direction after 30 seconds, thus invalidating their predicted Location, the previous retrieval request can be aborted and a new one initiated.)

To turn this example back-to-front, the same technique can be used when the processing of each retrieval request is slow, but the retrieval engine is currently idle. During this otherwise idle time the retrieval engine can be asked to retrieve information for a predicted future context: if the prediction turns out to be correct, and if the application asks for a retrieval for the predicted context, then response will be immediate since the retrieval has already been done. This can help somewhat to meet the CAR challenge of providing near-continuous high speed retrieval.

7.2 Context-aware Caching

In practice we have observed that successive retrievals often yield much the same documents, but with scores gradually changing as the current context changes. This leads to the idea of using the set of documents yielded on one retrieval as a cache. Subsequent retrievals use this cache as the document collection to be retrieved from. The cache improves subsequent retrieval speed: the cache is typically much smaller than the original document collection, and thus retrieval from it is likely to be much faster.

The usage of a context-aware cache is similar to the way virtual memory is used. In cases where the application is continuously connected to the retrieval engine it can continue using a context-aware cache until the user strays outside the predicted range of contexts-of-interest. A bad prediction is not a tragedy, but a good prediction can lead to a big improvement in retrieval speed. Even in cases where a cache needs to be abandoned, its replacement cache can often be created by updating rather than completely replacing the previous one; updating can represent a considerable saving of bandwidth.

Context-aware caches work best when retrieval has the property that a small change in the current context will lead to a small change to the documents retrieved and their scores. Our feeling is that this will usually be the case, but we need to finish our experiments to verify it. Obviously, however, there will be exceptions as there are discontinuities both in the natural world and the man-made world.

To create a cache, the appropriate documents must be extracted from the document collection. The simplest way to do this is to make a retrieval request and use the retrieved documents as the cache for subsequent retrievals. A crude approach would be to make a retrieval request for the current context, and to set a low threshold: the retrieved documents, i.e. those that match the current context at least in a small way, are then used as a cache. This crude approach has the disadvantage that if matching of fields is designed to give a quickly reducing score as differences increase (e.g. a difference in location of more than a mile gets a score of zero), then the cache will soon become invalid as the current context changes.

An improvement of the crude approach to extracting the cache is the following. Instead of using the current context as the query, the approach is to use the union of all current contexts likely to occur while the cache is in use. Thus the query will typically cover a range of locations that the user might reach during the lifetime of the cache, the range of times for which the cache is needed, the range of temperatures likely to be encountered, and so on. These ranges are set on the basis of prediction. This wide-ranging query is fed to the retrieval engine, and the retrieved documents whose score exceeds some threshold are then used as the context-aware cache.

A cache is also invaluable when the user is only periodically connected to the document collection, e.g. because of physical limitations or cost. For instance in fieldwork applications [11], the fieldworker may download a cache in the morning and use it for a whole day. At the other end of the time spectrum a user employing wireless communication may employ a cache for a few minutes to combat communication discontinuities or charges. Generally, as in these examples, the cache will be in the memory of the mobile device, but it is also possible for servers to use caches.

8. USING CONTEXT CHANGE AND LEARNING TO IMPROVE PRECISION

The theme of the above discussion is that the use of the context-of-interest can lead to improvements in both precision and retrieval speed. We now concentrate just on precision, and on other ways to use analysis of change to improve it.

In terms of improving precision we will exploit a mixture of (a) higher level ideas, such as the context-of-interest, and (b) selection and adaptation of matching algorithms, in particular the algorithms for matching of fields, and the weighting of fields. As we have said, we believe that algorithms for field matching need to depend on the nature of the field. As an example, Jones and Brown [6] have postulated two possible algorithms for matching Location fields. As we have already suggested in this paper, an algorithm for matching the Time field on a document (which we assume relates to opening hours) with the Time field on the current context might give a score of zero if an attraction is about to close. If we assume the Context Diary is also available to these algorithms, they can take account of change. For example an algorithm relating to a tourist might assume that the tourist prefers not to backtrack their route, and thus documents whose Location relates to positions already visited get a low score.

Setting the relative weights of fields is an obvious way of tuning the retrieval algorithm. Some of these weights will be permanent: e.g. if you believe the claim that context-aware applications depend on 'location, location, location', then the Location field gets a high weight. Sometimes the relative weights will be adjusted by the user ('I am especially interested in the effects of my heart rate [as measured by a field representing a heart rate sensor]'). On top of these we believe there is scope for changing weights dynamically to improve precision, following the experience of the methods used by the Jimminy system [12]. In particular we plan to investigate the following conjectures:

- a changing field should be given more weight than a static one.
- a field whose *rate of* change has altered should get higher relevance. An example would be a previously static field that had suddenly started to change. (If this conjecture is true there is a roundabouts-and-swings situation; such a change would mean that previously-made predictions would be wrong, thus losing performance, but precision might be gained by adding weight to the suddenly changing field.) This relates to our earlier comment that an unpredicted change in a field is a significant event that should cause the field to have greater weight.

In addition well-known IR techniques such as relevance feedback can be used to adjust weightings. The Context Diary provides a mechanism for recording and using this.

9. EXPERIMENTAL SYSTEM

We have built an experimental system to begin exploration of the ideas presented above. The prototype is written in Java to aid portability and to facilitate usage over the web, and includes a retrieval engine that covers both proactive and interactive retrieval. Ideally we would have liked to base the prototype round an existing retrieval engine, but the need to cover both proactive and interactive retrieval – thus covering both IR and IF aspects – and the need for experimentation in context-aware matching algorithms led us to build our own engine. Figure 1 shows a block diagram of the operation of the prototype system. In order to improve modularity our architecture is based on a pipeline model. A typical use of a pipeline is as follows; we assume in the example that the end-user is a tourist:

(1) a pre-processor takes as input the tourist's current context. Using the Context Diary it predicts the context-of-interest, which is then passed to stage (2). The pre-processor also calculates field weights, based on the way fields are changing, as recorded by the Context Diary.

(2) the first retrieval stage is a proactive retrieval using a document collection about tourist attractions. It uses as the current context the context-of-interest supplied by stage (1). It uses as retrieval plug-ins the field weights calculated by stage (1); it might also use some further retrieval plug-ins supplied by the application designer, e.g. an algorithm for matching locations. The output is a set of retrieved documents; each such document has an overall score and each matched field within it has a score too.

(3) the second retrieval stage is an interactive retrieval, with components designed to factor in the user's interests. This stage uses a completely different 'current context': not the tourist's physical current context, but one representing user preferences. This current context is used as a retrieval query, and represents the tourist's current interests, e.g. that the Body field of a document should contain the word 'architecture'. This stage takes as input the documents retrieved at stage (2) and produces as output a subset of them; this subset will have the original scores changed, according to how well the tourist's interests appear to be matched. (Conceptually stages (2) and (3) can be regarded as one, with the user's interests and their current physical context treated as a single entity; in implementation terms, however, it may be convenient to separate them.)

(4) the post-processing stage takes the retrieved documents output from stage (3) and massages the scores. For example, if retrieval has been unexpectedly slow, the massaging might take account of how the current context has changed since it was supplied in stages (1) and (3). As another example, the post-processor might use information in the Context Diary to decrease the scores of documents whose Location fields corresponded to places the user had already visited.

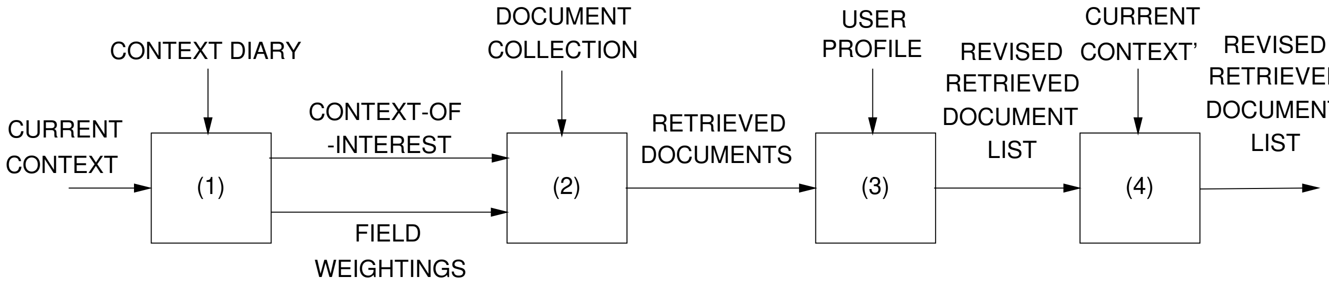

Figure 1: Experimental CAR retrieval prototype pipeline.

(5) those documents from stage (4) whose overall score is above a certain threshold are presented to the user.

The system does not currently include any user feedback functionality to modify field weights and user interest profiles, but we plan to add and explore these as our work develops.

10. SUMMARY

Our belief is that CAR will become an important technology in mobile applications, and there will be an increasing need to improve CAR performance both in terms of precision and speed. In this paper we have presented a number of ideas that can be used to augment retrieval engines to make them good CAR vehicles. A key to the implementation of our ideas is maintaining a Context Diary of past and possibly future values of the fields of the current context. This can be used:

- to derive a 'context-of-interest', which might derive better retrieval results than using the current context.
- to help in building context-aware caches.
- to help write effective 'retrieval plugs-ins' to fine-tune retrieval algorithms.

We hope that these ideas will provide a platform for tackling the future demands of production-quality CAR. We believe there is a large scope for more detailed research and experimentation in each of the above three areas, and we hope the ideas presented here might act as catalysts to these investigations.

Acknowledgements

Peter Brown is grateful both to the Leverhulme Foundation and to Exeter University for supporting this work. Both authors are grateful to Lindsey Ford for his implementation work.

11. REFERENCES

[1] Beigl, M. 'MemoClip: a location-based remembrance agent', *Personal Technologies*, **4**, 4, pp. 230-233, 2000.

[2] Brown, P.J., Bovey, J.D. and Chen, X. 'Context-aware Applications: from the Laboratory to the Marketplace', *IEEE Personal Communications*, **4**, 5, pp. 58-64, 1997.

[3] Brown, P.J. and Jones, G.J.F., 'Context-aware retrieval: exploring a new environment for information retrieval and information filtering', *Personal and Ubiquitous Computing* 2001, In Press.

[4] Dey, A.K. and Abowd, G.D. 'CybreMinder: a context-aware system for supporting reminders', in Thomas and Gellersen (Eds.) *Handheld and ubiquitous computing, HUC 2000*, Springer, pp. 172-186, 2000.

[5] Dourish, P., Bellotti, V., Mackay, W. and Chao-Ying Ma. 'Information and context: lessons from a study of two shared information systems', *Proceedings COOCS'93*, Ca., USA, pp. 42-51, Nov. 1993.

[6] Jones., G.J.F. and Brown, P.J., 'Information access for context-aware applications', *Proceedings of ACM SIGIR 2000*, Athens, pp. 382-4, July 2000.

[7] Kindberg, T. *et al.* 'People, places, things: web presence for the real world', *Tech-Report HPL-200-16*, HP Labs., 2000.

[8] Lamming, M.G., Brown, P.J., Carter, K., Eldridge, M., Flynn, M., Robinson, P. and Sellen, A. 'The design of a human memory prosthesis', *Computer Journal*, **37**(3), 153-163, 1994.

[9] Lövstrand, L. 'Being selectively aware with the Khronika system', *Proc. ECSCW'91*, Amsterdam, The Netherlands, pp. 265-277, 1991.

[10] Newman, W.M., Eldridge M.A. and Lamming, M.G. 'Pepys: generating autobiographies by automatic tracking', *Proc. ECSCW'91*, Amsterdam, The Netherlands, September 1991.

[11] Pascoe, J., Morse, D.R. and Ryan, N.S., 'Developing personal technology for the field', *Personal Technologies* **2**, *1*, pp. 28-36, 1998.

[12] Rhodes, B.J. 'The wearable remembrance agent: a system for augmented memory', *Personal Technologies*, **1**, 1, pp. 218-224, 1997.

[13] Rhodes, B.J. and Maes, P., 'Just-in-time information retrieval agents', *IBM Systems Journal*, **39**, 4, pp. 685-704, 2000.

[14] Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M., and Gatford, M. 'Okapi at TREC-3'. In Harman, D.K., editor, *Overview of the Third Text REtrieval Conference (TREC-3)*, pp. 109-126. NIST, 1995.

[15] Robertson, S., and Walker, S. 'Threshold Setting in Adaptive Filtering', *Journal of Documentation*, **56**, 3, pp. 312-331, 2000.

[16] Singhal, A., Buckley, A., and Mitra, M. 'Pivoted Document Length Normalization', *Proceedings of ACM SIGIR 1996*, Zurich, pp. 21-29, August 1996.