Context-aware retrieval: bringing information retrieval and information filtering ever closer together

Peter J. Brown

Computing Lab., The University, Canterbury, Kent CT2 7NF, UK
pjb@ukc.ac.uk

Introduction

In a seminal paper, Belkin and Croft [1] pose the question "*Information filtering and information retrieval: two sides of the same coin?*". Here we look at a different type of retrieval, *context-aware retrieval*, which is becoming increasing important with the rise of mobile computing. Context-aware retrieval not only brings information filtering even closer to information retrieval than Belkin and Croft showed, but also, and, more surprisingly, also bring out a symmetry between retrieval queries and retrieved documents — a symmetry that allows these to reverse roles.

Belkin and Croft's question is itself highly relevant to our discussion, so we will start with that. Information Retrieval (IR) is familiar to most people: it is the activity of sending queries to find, from of a repository of documents, those documents that are of relevance to the user. (Belkin and Croft use the word "texts" for what we call "documents".) Information filtering (IF) is where the user sets a *profile* to say what documents they are interested in, and the retrieval system uses this regularly to feed to the user the documents that are relevant. Often IF is applied to sets of new documents that are continually arriving, and the profile might specify, say, that the user was interested in papers on education in China, and therefore wanted to be notified whenever any paper matching this topic arrived. The profile is logically the same as a query, and we will use the terms interchangeably here.

Belkin and Croft's conclusion is that IR and IF are conceptually the same. Obviously in implementation terms, particularly when dealing with optimizations to cater for massive numbers of documents, the two retrieval methods can differ a lot. The practical difference between IR and IF applications can be characterized by noting that in IR the documents are relatively static and the queries are dynamic — indeed each query tends to be different from its predecessors — whereas in IF the queries (profiles) are relatively static whereas the documents are dynamic; in cases where IF is only applied to new documents, a document, once processed, is never looked at again by the IF retrieval system.

Retrieval can apply to unstructured documents, or to semi-structured documents, e.g. documents with fields such as "Author", "Title", etc., but where the material within fields is largely unstructured. We concentrate here on the latter case, and in particular on the case where the documents are split into named fields. We also concentrate throughout on the basic case: almost every element of IR and IF can, and has, been made more elaborate. This applies to the matching process, the nature of queries and profiles, the selection of retrieved material, the repository of retrieved material (is it a fixed collection or is information dynamically sought from numerous sources?), ... In general what we say here is often applicable to the more elaborate cases, though sometimes there is a need for corresponding elaboration in the techniques we apply.

Symmetry between queries and retrieved documents

We will start by considerating the simplest form of IR, in order to draw a parallel between queries and the documents they retrieve from. To many users, the most familiar form of IR is based on templates. This is commonly used where, as we are assuming here, the documents are divided into named fields. Thus the template might consist of three fields: "Author", "Title" and "Year". The author fills in one or more or these fields — if there is more than one they are combined together. The resultant query is structurally the same as a document, i.e. it is a set of named fields, though almost always the query will be much shorter and with fewer fields than the documents to be retrieved. Here we will use SGML tags to represent the names of fields within documents: thus an example of a query, represented as a mini-document is:

```
<Author> Knuth
<Title> programming
```

The retrieval process consists of matching the mini-document representing the query against the real documents. The retrieval system will have rules for matching values of fields: a simple one for string values is that the string in the query must be a substring of the string in the document, and a rule for numerical values, like the year field, is that the year or range of years in the query must overlap the year or range of years in the document.

A small extension to the idea of representing the query as a document is to add extra fields to it which do not affect the matching process, but serve other purposes, such as identifying the query itself. Thus we could add an extra field called `<Intent>` which identifies the purpose of the query. The query might then become:

```
<Intent> material for my
  CO859 lecture
<Author> Knuth
<Title> programming
```

Such extra fields are particularly apposite when a query is a profile.

This leads on to a further general point. After the retrieval has occurred, the next stage is to send the material to the client, e.g. to present it on the user's screen. Often this consists simply of showing the matched document; however in general it can mean extracting some parts (fields) of the matched document and some parts of its matching query, e.g. the `<Intent>` field we have suggested above. As we shall see, there are cases where all the retrieved material comes from the query, not from the document it matches.

Context-aware applications

Recently much attention has been focussed on *context-aware* retrieval. This is our main focus here, because its throws new lights on IR and IF. The interest in context-aware retrieval has been driven by the increasing cheapness and availability of sensors that can automatically detect the user's current context. For example there are lots of technologies for detecting a user's location, including GPS, active badges [10] and cellphones [5, 7]. A typical context-aware application is a city tour: the user moves around a city, carrying say a PDA, and, because they are carrying the relevant sensors, their *present context* is continually recorded. The present context will typically consist of a number of fields, covering location of the user, time of day, season of the year, orientation, and perhaps more exotic measures such as heart rate. (Obviously, as with most of the discussion in this paper, this is the simplest example, and there are applications where the present context is something much more elaborate than a set of named fields.) An author for the application has prepared a guide to the city based on pieces of information attached to different contexts, and, when the present context matches the context of the information, the information is automatically retrieved and presented on the PDA screen. We use the term *note* to describe the pieces of context-aware information. A sample note, where the `<At>` field is used for location, is:

```
<At> [[Some location]]
<Info> The tall Market
 Cross is ...
```

Here we assume that the `<Info>` field is not normally part of the matching process, but is to be used primarily in the role of information to be presented to the user. We use the term *active field* to denote those fields that take part in the retrieval matching process. A simple, global, approach is to associate active fields with certain names: thus `<At>` fields are active whereas `<Info>` fields are not. A useful addition to this approach is to allow individual notes to override the global settings on some individual fields. Often the context of a note will consist of a single active field, perhaps a location as in the above example, and the note will be retrieved if the value of this active field is matched by the present context, e.g. if the user is at the specified location. Sometimes, however, a note will consist of several active fields: for example information about a public garden may have the context of the location representing the area near the garden, together with a time corresponding to the opening hours of the garden. Information is only retrieved if all the active fields of the note are matched, as with the fields of a simple IR template. Thus the information about the garden would only be retrieved if the user's present context showed that they were in the specified area and that their `<Time>` field matched the specified time.

It turns out that context-aware retrieval straddles many areas of retrieval. In particular:

- it straddles IR and IF.
- it straddles the static and the dynamic cases: it is not possible, except in special-purpose applications, to make any pre-judgement as to what constituents are static and what are dynamic.
- it brings together queries and documents, indeed the two can sometimes interchange roles: hence the title of this paper.

Some examples and some needs for context-aware applications

Before analysing the retrieval needs of context-aware applications, it is useful to look at some practical matters. Firstly, numerical quantities are often ranges: thus a note might have the context of a certain range of temperatures, rather than a single value, and a location might be specified as everywhere within a given circle or rectangle. Ranges may also be used for values of sensors because of uncertain accuracy, as for example in non-differential GPS. Secondly, context-aware applications become much more powerful if values of sensors can be overridden by values set by the user. For example the user could pretend they were at a certain location, thus retrieving information for that location. Indeed the whole world may be a make-believe world where information is stored in different places, and retrieved as the user moves around the pretended world. Pretence extends the use of ranges: the user could, for example, defy the maxim that you cannot be in two places at once: they could pretend they were in a whole city, thus retrieving all the information for that city.

At a higher level it is useful to consider the different characteristics of context-aware applications. Here are four examples.

(1) Our first example is the city tour as we have proposed above. Here there is a reasonably static set of notes relating to various contexts within the city, and the present context changes continuously as the user moves around.

(2) A complement of (1) is the case where there are a lot of constantly changing notes that relate to a city, e.g. hourly weather forecasts, information on the current location of public transport vehicles (as in BT's Route 66 information [9]) and information about locations of current traffic congestion. The

user may be sitting in an office looking at this information, and retrieving it (automatically) whenever a relevant change occurs. Thus the present context is static, but the notes are dynamic. This application is logically identical to the city tour, except that what is static and what is dynamic have been reversed. (Another example that well highlights the possible relationship between static and dynamic aspects are Reactive Environments [4], where there may be a large number of stationary sensors in, say, an office environment, which know what is dynamically going on around them.)

(3) Assume the user interface for the city guide described in (1) displays a map, perhaps centred on the user's present location. The application wishes to highlight on the map, by using a red dot or rectangle, each location to which a note is attached. To retrieve these locations, the application sets the present context to be the whole area covered by the map, and hence retrieves all the notes that are relevant to this area. What the application asks to retrieve for a note is just its location field, so that it can plot the red dots and rectangles. It might also be interested in a `<NoteTitle>` field, if such existed, to use as a label to the red dots. In this example, the retrieval matching task is exactly similar to (1) and (2), but the fields that are to be retrieved from the matching notes are completely different.

(4) Assume the user of our city guide becomes overloaded with all the information retrieved. She therefore asks to limit the retrieved information by saying that the `<Info>` field of the retrieved notes must contain, say, the word "history". In this case we are applying, as a second activity, a conventional IR operation to the material that has been retrieved by contextual matching. In some cases the problem may be that the user has a lot of sensors, and gets a flow of information related to each sensor value: location-based information, temperature-based information, etc. The user might limit information overload by specifying that one of the existing contextual fields *must* be matched. For example if she is currently focussed on heart-rate monitoring she might require that all notes that are retrieved must match the current `<HeartRate>` field: if a note only had a `<Temperature>` field, say, it would not be retrieved, even if the temperature matched. (A less good alternative is to switch the temperature sensor off: it is less good because this might stop important notes based on both HeartRate and Temperature.) The aim in this application is, of course, to increase relevance; retrieval research is full of new approaches to improve relevance, and, though we do not consider them here, they can also be applied to context-aware research.

Hopefully these examples show that context-aware applications place multiple demands on retrieval, and therefore that the retrieval model must be flexible. Of course, it is equally desirable that it be conceptually simple too, and that is what has driven our model.

A model for context-aware applications

In our model [2, 3] for context-aware applications both the notes and the present context consists of a set of named fields, represented, as in our above examples, as SGML tags. For example `<At>` is the location field. Our approach to matching is based on the Boolean query. Boolean queries are, of course, regarded as an outdated technology in retrieval research circles, but still form the basis of much retrieval in the field. They turn out, however, to be a good basis to meet the needs of context-aware retrieval, at least the simpler applications. This is important, since the symmetry between queries and documents would not be possible if the matching method was fundamentally asymmetric. Typically our matching is based on fields with numeric values. Thus if a note has the context:

<At> *[[some location, X]]*
<Temperature> 20..25

where 20..25 means any value in the range 20 to 25, and this is matched against a present context that contains:

<At> *[[some location, Y]]*
<Temperature> 22
<plus lots of other fields not relevant to this particular query>

then, since the temperature fields match, this matches if location X matches location Y.

In many cases symmetry carries over to the matching of field values. The most common rule for matching the values of numeric fields is that they match if their ranges overlap. If, on the other hand, the value of a field is a string, there are two common possibilities:

(a) the two values to be compared must be identical. This often applies if the field is the name of an object (e.g. a person, an animal, a piece of equipment).

(b) the query is a pattern that the corresponding field value must match. A common special case of this is when the query specifies a word that must be present in the corresponding value.

Apart from this last case of pattern matching — which is rarer in context-aware retrieval than in most other kinds of retrieval — matching is symmetric: if value A matches value B then value B matches value A. We shall see later that there are cases where such symmetry is important.

In relation to retrieval in general, our model is specialised in the two senses that (a) data is semi-structured and (b) the matching process is the (exact-match) Boolean query. Nevertheless a large number of existing retrieval applications fit these constraints, so we are not considering an unusual case.

Representation and matching

In our model all the data, both the notes and the present context, consist simply of a set of named fields. Everything, therefore, is a semi-structured document of the same form. The two parties in the overall retrieval process are

(a) the current set of notes: initially the user and/or the application says which sets of notes they want to use; this is itself a retrieval process — perhaps a context-aware one — but is not the topic of this paper.

(b) the present context: in general there could be several present contexts, but for simplicity we will assume there is only one, i.e. it is a personal context-aware application rather than a group one.

The application specifies which of the field names are to be the active ones used for matching, and, for the case when a match occurs, the names of the fields to be retrieved from the matched note and from the present context.

The basic nature of a context-aware application is that the active fields of the notes are effectively queries that are profiles, and the present context is the document against which they are matched. All the active fields in the query must be matched by the present context, though the present context will typically contain further active fields — e.g. our oft-quoted HeartRate one — which take no part in the matching process for a particular note. Thus there are typically lots of simultaneous queries but only one document to be retrieved from — a big contrast to most retrieval applications. That document, the present context, is, of course, typically changing all the time — if everything were static it would not be a very interesting application. There is also a contrast in what is retrieved: the user will normally want to see some of the further fields of the note, e.g. the <Info> field, not fields of the matched document. Thus the retrieved information is associated with the query, not the document.

This retrieval process covers (1), (2) and (4) of our samples of context-aware applications; in case (2) it is the notes that are changing, not the present context, but the process is logically the same.

In case (3), where the user wants to limit the information retrieved, we need to apply a second stage: a conventional IR process applied to all the notes that have been matched by the first process. (Actually, whether it is a second, post-processing, stage, or a preparatory, pre-processing, stage is arbitrary; we shall assume the former.) To cater for this second stage, if needed, there is a second, reverse-direction, retrieval stage where certain fields of the present context are treated as query fields. This second stage will use, in general, a different set of active fields from the first stage. For example one field of the present context, set by the user, may be `<Info> history`, meaning that the matching note must match (contain) the word "history"; this is of course an entirely pretended field, not associated with any sensor (unless there is an amazing sensor that automatically senses the user's wishes!). This `<Info>` field would be an active field in the second stage of retrieval. We thus have a completely symmetrical retationship between the present context and the note it is matched against: each contains a query that is applied to the other. The symmetry is carried further if the same field is used in both stages: this would apply in our earlier example where the user was only interested in matches of the `<HeartRate>` field. A note, *if* it contains an active `<HeartRate>` field, must match a corresponding `<HeartRate>` field in the present context, and then, as the second stage, the `<HeartRate>` field in the present context must match a corresponding field on the note (thus eliminating notes that match at the first stage but do not contain a `<HeartRate>` field). Obviously this use of matching A against B and then B against A is only practical if matching of values is fundamentally symmetrical: it does not work if A is a pattern to be applied to the string B. (There are also some problems when a field occurs twice in the present context: e.g. the `<At>` field may be the user's current location, set by a sensor, but the user has set a second `<At>` field, covering the area of a city centre, saying that she only wants notes retrieved if they match this, i.e. if they apply to the city centre. If she wandered out of the city centre, this second-stage `<At>` field would inhibit retrieval.)

Symmetry of authorship

Symmetry, which is very much the theme of this paper, can apply to an entirely different dimension: symmetry between author and user.

As we have presented it, there are two parties in an application: the user to whom the present context applies, and the author of the notes that are to be matched. In many applications the author and the user are one and the same person: the user has written notes to themselves, to be triggered when, for example, they come into their office, or when they pass a certain point on a return from a hike. In other applications, although the author is different from the user, the author not only writes the notes but also has some control over the present context, e.g. the retrieval fields within it.

Thus the normal retrieval concept of a user's query being employed to retrieve material from documents normally written by other authors can be replaced by different arrangements between authors and users, some of which are more symmetrical.

Implementation questions

Many existing retrieval processes have been specially designed to cater for massive numbers of documents, and/or massive numbers of queries. Often the processes use optimizations that depend on some element of the data being relatively static. These optimizations would not be applicable, at least in general, to the processes we have described. Our implementation experience is based on at most tens of thousands of notes, and no special optimizations are used. Performance is acceptable, but finding optimizations to cater for larger amounts of data is likely to be a real challenge. Indeed it has been called the 'Grand Challenge' [8].

Conclusions

Context-aware retrieval certainly brings IR and IF closer together, and represents a world where many of the normal parameters, such as what is static and what is dynamic, are turned on their head. An example like this, that represents a different view of what was previously a reasonably established world, will, we hope, be a catalyst for new thinking, and for further strides, following on from the stride made by Belkin and Croft, into generalizing the concepts and people's understanding of them.

Acknowledgements

Several of my colleagues have contributed greatly to this work: I would especially like to mention John Bovey and Jason Pascoe. The idea for the paper was inspired by the clear thinking and superb insights shown in an MSc dissertation [6].

References

[1] Belkin, N. J. and Croft, W. B. 'Information filtering and information retrieval: two sides of the same coin?', *Comm. ACM*, **35**(12), pp. 29-38, 1992.

[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. 'Triggering information by context', *Personal Technologies* **2**, *1*, pp. 8-27, 1998.

[4] Cooperstock, J., Fels, S., Buxton, W. and Smith, K.C. 'Reactive environments: Throwing away your keyboard and mouse', *Comm. ACM*, **40**(9), pp. 65-73, 1997.

[5] Duffet-Smith, P. 'High precision CURSOR and digital CURSOR: the real alternatives to GPS', *Proceedings of EURONAV 96 Conference on Vehicle Navigation and Control*, Royal Institute of Navigation, 1996.

[6] Kennaugh, R.J. *An agent based information dissemination system*, MSc dissertation, Imperial College, London, 1998.

[7] Klukas, K., Lachapelle, G., and Fattouche, M. 'Cellular phone positioning using GPS time synchronisation', *GPS World*, **9**, 4, pp. 49-54, 1998.

[8] Oard, D.W. and Marchionini, G. *A conceptual framework for text filtering*, Report EE-TR-96-25, Univ. of Maryland, 1996.

[9] Thompson, S.M. and Sheat, D.E. 'Exploiting Telecommunications to Deliver Real Time Transport Information', *IEE 9th International Conference on Road Transport and Control*, IEE, London, pp. 59-63, 1998.

[10] Want, R., Hopper, A., Falcao, V., and Gibbons, J.J. 'The active badge location system', *ACM Transactions on Information Systems*, 10, 1, pp. 91-102, 1992.