ELECTRONIC PUBLISHING, VOL. 4(4), 205–217 (DECEMBER 1991)

NRT: news retrieval tool

MARK SANDERSON AND C. J. VAN RIJSBERGEN

Department of Computing Science
The University
Glasgow, G12 8QQ, UK

SUMMARY

The amounts of information that mankind produces are vast, running into billions of documents. Traditional ways of holding this information have become impracticable and so methods of storage are being switched from paper and microfiche to magnetic and optical disks. In the last thirty years, as more information has been put onto computers, work has gone into using the computer to get away from the restrictiveness of manual indexing and move towards a more flexible system of information acquisition.

Many companies offer (for a price) the opportunity to access the information stored on their systems. Unfortunately, most of these companies use software that was developed in the sixties when the field of information retrieval (IR) was still very young. This means that the services they offer are rather primitive. The Financial Times' IR service, Profile is typical of such commercial systems. It has been the aim of the NRT project to investigate ways of incorporating into Profile the new ideas in IR, that have occurred in the last ten to fifteen years.

KEY WORDS Weighted key term information retrieval Relevance feedback Wide area networks User interfaces

1 THE PROFILE SYSTEM

Profile holds four to five years' worth of articles from every British quality newspaper. It also has specialist business and finance journals. The system is updated daily with the new editions of each publication. At present, the size of the data in Profile is about 18 gigabytes growing at about 250 megabytes a month. It is run on conventional mainframe computers using magnetic hard disk storage. The search and retrieval software is based on the STAIRS IR system. All that is required to connect to Profile is a modem and a teletype terminal. What follows is a short example of a search on the Profile system.

```
select ft

Financial Times Newspapers, (C) 1984,85,86,87,88,89,90
```

The user has selected the *Financial Times* newspaper.

```
get disney in france

GET DISNEY IN FRANCE
COMMON WORD IN REMOVED
         0 ITEMS RETRIEVED
```

The user is interested in articles about Disney's choice of France for its European Disneyland, no articles are retrieved because Profile was searching for an exact match to the phrase "disney in france". The best strategy with Profile is to work word by word.

0894–3982/91/040205–13$11.50

Received 13 June 1991
Revised 4 November 1991

```
get disney

GET DISNEY
       151 ITEMS RETRIEVED
>
pick france

PICK FRANCE
        19 ITEMS RETRIEVED
```

The pick command searches on the 151 items retrieved by the get command. The user now lists the retrieved articles which are ordered by date of creation.

```
headline all

HEADLINE ALL
SORTING

 1   FT  20 Nov. 89 Arts: Museum through the looking-glass -
Architecture (895)
...
```

The search shown above is a simple one, however Profile supports complex Boolean (AND, OR and NOT) and proximity (retrieve document if words occur in same sentence/paragraph etc.) operators.

After using Profile for some time, it became clear that there were two distinct areas where there were problems, the retrieval model and the interface.

- The retrieval model is a classic, database-like, Boolean retrieval model. The problems of such a model are well documented [1] and won't be entered into here. But in addition to the problems of the model not retrieving well, there is the problem of the kinds of users wanting to access Profile's information. They are unlikely to be computer literate and so won't be capable of, or be interested in, constructing the complex Boolean and proximity search expressions needed to use Profile effectively.
- The interface is a typical text-only command-driven interface. The user has to know at least five commands in order to perform a simple search, not including any of the query language operators. There are a number of modalities that can cause confusion, particularly to new users. The interface was designed with the simplest computer/terminal in mind with the remote retrieval engine controlling the interface and the user's search state. So if the remote machine goes down, everything is potentially lost.

2 THE NRT SYSTEM

Commercial IR systems are so tricky to use, it is often the case that organizations have to employ search intermediaries to access them. It was one of the aims of NRT to make an IR system simple enough to allow anyone to use it. An overview of NRT is shown in Figure 1. On the left, a graphics workstation (controlling the interface tasks) communicating with the

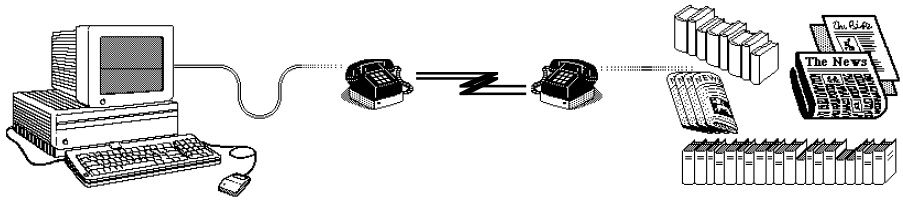

Figure 1.

remote FT computer, which performs the retrieval tasks. The division of labour between the two computers was split so that use of the slow communications line is kept to a minimum. The two machines communicate using a protocol agreed between Glasgow and Profile [2]. The protocol makes no assumptions about the workings of the two machines which means that different interfaces and servers could be written and used as long as the protocol is adherred to.

2.1 The retrieval model on the server

The choice of retrieval model to be used by NRT was a simple one. It had to be quick to implement and had to work better than Boolean retrieval; the obvious choice was a form of term-based retrieval. Figure 2 shows an overview of the term-based retrieval system used by NRT.

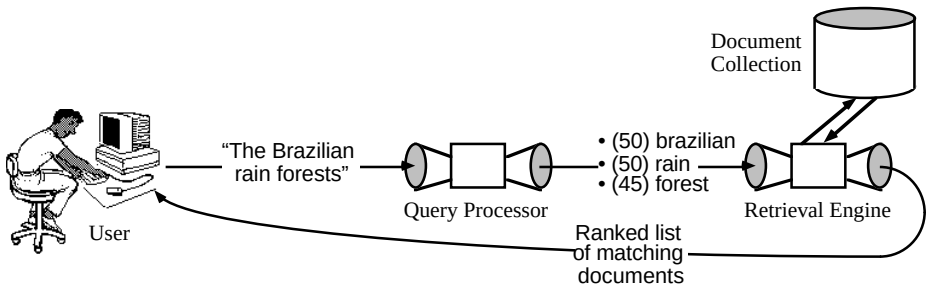

Figure 2.

The user's query is passed to a query processor which breaks the query up into its constituent words normalizing lettering to lowercase. Any non-content-bearing words ("the", "but", "and", "or" etc.) are discarded, suffixes are removed from the remaining words using a process called conflation (which is explained below) and finally a score or weight is assigned to each word which indicates the word's "usefulness" to the query (this will be explained in more detail below). What remains then is a list of query term/weight pairs which are passed to the retrieval engine. Conceptually what happens now is the retrieval engine matches every document in the collection with the term list and a score indicating the degree of match is assigned to each document.[1] The documents are then

[1] Although there are a few retrieval systems that actually do this [3], for most systems, NRT included, to match the term list with every document in a large collection would be impracticable, so a term index file of the document collection is built and used in the matching process.

ranked by their score and the top N documents are presented to the user. N is specified by the user and is typically a value between 10 and 50.

As was mentioned above query terms are assigned a weight to indicate their "usefulness" to the query. The weight is an indication of how well a term can resolve the issue of whether a document is relevant to the query or not. In term-based retrieval, the weight of a word is inversely proportional to its frequency of occurrence in the document collection, so high-frequency words (e.g., "system", "computer") have a low weight and low-frequency words (e.g., "Goretex", "Macintosh") have a high weight. There are many ways to calculate term weights (for a fuller discussion see Reference [4]), but the function chosen for NRT is (see Reference [2] for reasons why),

$$w_i = \log \frac{N}{n_i}$$

where the w_i is the weight of term i, N is the number of documents in collection and n_i is the number of documents with an occurrence of term i.

Like the weighting function, there are many different ways of calculating the document matching score (again for more detail see References [2,4]), but the chosen NRT function is,

$$\text{score} = \sum_{i=1}^{t} d_i \cdot w_i$$

The sum is evaluated over all the t terms in the query. d_i indicates presence or absence of the query term i in a document.

So the matching score of a document is simply the sum of the weights of the query terms occurring in that document. A document containing more higher-weighted query terms will get a higher score.

2.1.1 Conflation

When matching words in a query to words in the document, suffixes can be a problem. The query "work practice" would not match to a document containing the words "working practices". The solution to this is to use a conflation algorithm which strips the suffixes off words, leaving a root stem (e.g., "working", "worked", "worker's" conflated to "work"). As was outlined above matches between query and document are performed on stems. Of course errors sometimes occur, for instance "community" and "communist" conflate to the same stem "commun", however the advantages of correct stemming outweighs the mistakes. The conflation algorithm (written by Porter [5]) used in NRT is a simple one that uses a set of rules that progressively removes the suffix from a word. The rules were designed to be as general as possible with no special cases. It was found that it performs as well as many of the more complex algorithms that have been developed.

2.1.2 Relevance feedback

Relevance feedback is a process that allows a user to add to his query whole documents which he considers as relevant. When a document is marked as relevant (added to the query), the retrieval system analyses the document text, picking out words that are statistically significant to the document and adds these words to the query.

Relevance feedback is a very good method of specifying an information requirement, because it releases the user from the burden of having to think up words for the query. Instead the user deals with the ideas and concepts contained in the documents. It also fits in well with the known human trait of "I don't know what I want, but I'll know it when I see it". An example of the usefulness of relevance feedback was given in a paper by Thinking Machines [3]. They wanted to search for articles about the Chernobyl disaster. Initially they typed in the word "Chernobyl" and got back articles on the disaster. They then marked these articles as relevant, the relevance feedback system added a number of words, from the relevant documents (words like "nuclear", "radiation", "russia" etc.), to the query and then performed another search. While looking through the new retrieved list, they found an article dated the day before the story broke. It was about a Finnish radiation detection centre finding unusually high radiation levels coming, they believed, from Russia. This highly relevant article could not have been retrieved on the initial keyword, only by words added using relevance feedback.

Obviously the user cannot mark a document as relevant until some are retrieved, so the first search has to be initiated by a textual query. This will return a list of documents covering a range of topics, but probably at least one document in the list will cover, or come close to covering, the user's interest. The user will mark the document(s) as relevant and perform another search, the next list should be closer to the user's requirement. This process of continually refining a query and retrieving is called the relevance feedback loop.

When relevance feedback is performed, the relevant document texts are processed in a similar fashion to a user's query (see above). The texts are broken into their constituent words, lettering is normalized, non-content-bearing words are removed and remaining words are conflated and weighted. However, the function used to calculate weighting is different, as might be expected many different functions have been devised. Sparck-Jones [6] provides a good overview of them. The function used in NRT is,

$$w_i = \log \frac{r_i(N - n_i - R + r_i)}{(R - r_i)(n_i - r_i)}$$

N is the number of documents in the collection, n_i is the number of documents with an occurrence of term i, R is the number of relevant documents in the collection and r_i is the number of relevant documents with an occurrence of term i.

Essentially what this rather complex function does is compare the frequency of occurrence of a term in the relevant collection (documents, the user marked as relevant) with the term's frequency of occurrence in the whole document collection (see Reference [2] for more detail). So if a term occurs much more frequently in the relevant collection than in the whole document collection it will be assigned a high weight. All terms are assigned a weight and are then ranked by that weight. The top N terms ($N \approx 20$) are then added to the user's query.

2.1.3 Adapting commercial systems

One of the problems with adapting commercial systems to term-based retrieval is the commercial vendors' reluctance to change the retrieval software they are running. This has led to much research into simulating term-based retrieval on Boolean systems [7,8]. However this was not the case with Profile. They were willing to re-index a portion of

their document collection (130 000 articles) and make the necessary software changes to implement full term-based retrieval.

2.2 The interface

Apart from a few exceptions [9,10] the issue of user interfaces to IR systems hasn't really been addressed. Some commercial Boolean retrieval systems have been produced with windowing interfaces but these are little more than the text interfaces with buttons in place of the commands.

For all the faults of the current Profile interface, its one advantage is that any kind of terminal can use it. It was suggested that a text-only interface be built for NRT, but this idea was rejected for the following reasons,

- It was felt that it is not unreasonable to expect some form of windowing environment on all personal computers in the not-too-distant future and so it was worth investigating what such an interface would look like.
- Most of the users of Profile aren't familiar with computers: it is extremely important to keep the interface as simple as possible and reduce the number of commands the user is required to know. A windowing interface would be able to provide this.
- Most importantly, the authors believe that term-based retrieval requires a windowing environment to allow it to be usable at all. This is because term-based retrieval requires the user to keep track of and edit several lists of objects, e.g. the user's typed query (viewed as a list of weighted terms), the documents marked as relevant and the result(s) of retrieval(s). These sorts of requirements would be very hard to accommodate with text only.

It was also decided that the state of the user's search should not only be displayed, on the local machine but also stored on it. So if the remote server or the communication line goes down, the user can locally save his/her search state and resume it later on.

It was with these criteria in mind, that the interface was built [11]. The best way to see the workings of the NRT interface is to see a run-through.

2.2.1 Run-through of a search

When the user starts up NRT he/she sees three windows (see Figure 3).

- The Keywords window (bottom right) displays the query to the remote computer. It also holds the two main commands/buttons that the user is going to use. "Search" is clicked to perform a search and "Start Again" is clicked to abandon the current search and start afresh.
- The ✓ (tick) window (top right) holds documents marked as relevant.
- The window in the bottom left holds a list of the windows belonging to the current search.

A search is initiated by typing a free text query into the text box in the Keywords window and clicking the "Search" button. Two things now happen.

- First the user's query is sent to the remote server to be cleaned up. Capital letters are converted to lowercase, fluff words ('the', 'and', 'but' etc.) are removed, suffixes are

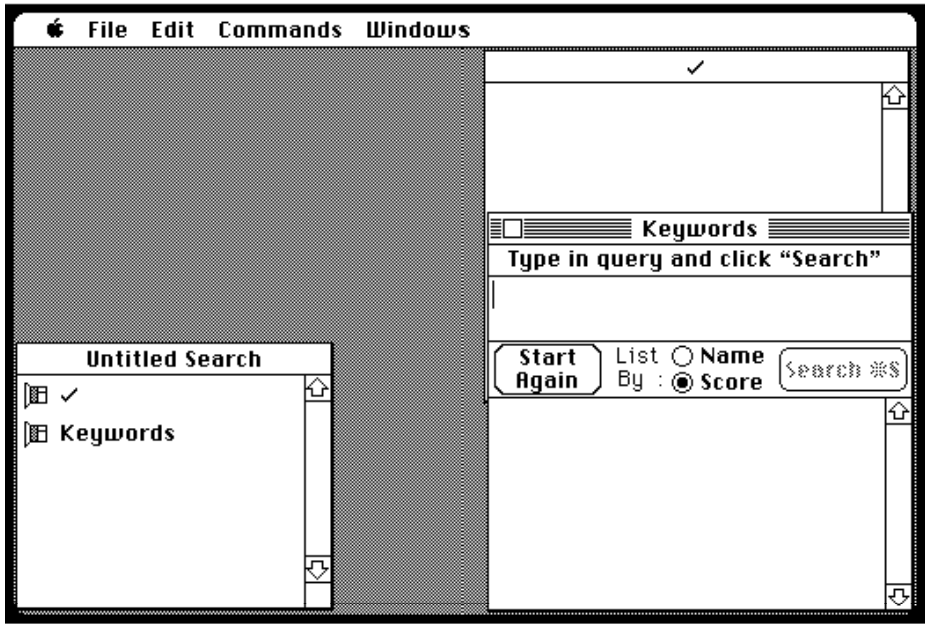

Figure 3.

stripped (e.g. 'forests' to 'forest') and finally a weight is assigned to the remaining words. A list of word/weight pairs is then sent back to the interface for display (see Figure 4). Note that the weight, is indicated by the slider on the right. The user can adjust this weight by moving the slider with the mouse. On the left of each word is a small face icon (☺), this indicates that the word has been entered by the user.

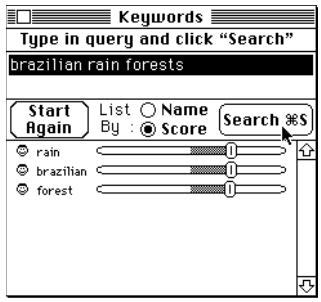

Figure 4.

- Having received the word/weight list, the interface initiates a search to the server, the result of which is displayed in a new window (see Figure 5). The retrieved documents are listed in relevance order with the most relevant at the top of the list. The list can be browsed by the use of the scroll bars on the right-hand side of the window. The user can see the content of an article by double clicking on the article heading in the retrieval window (see Figure 6).

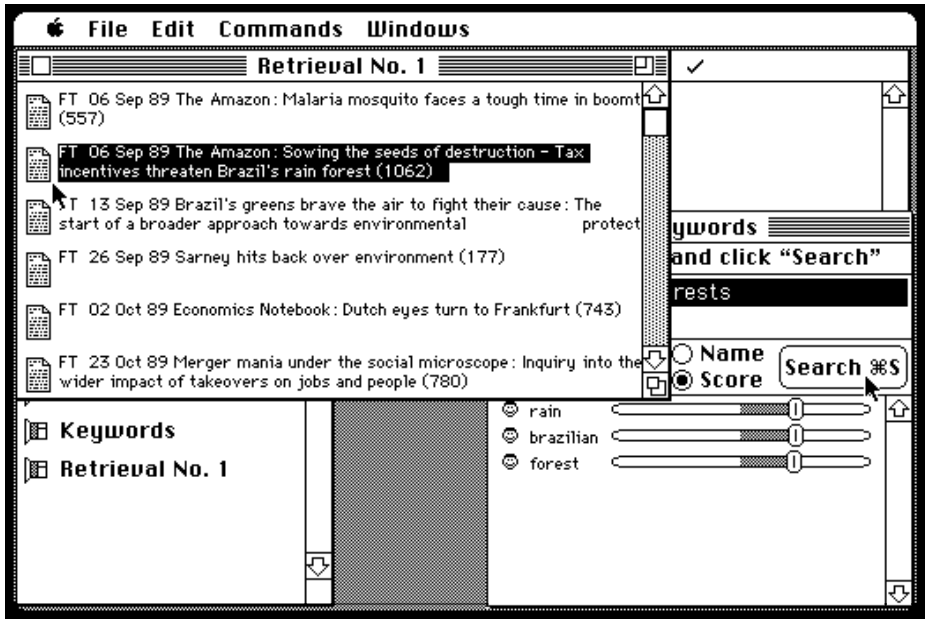

Figure 5.

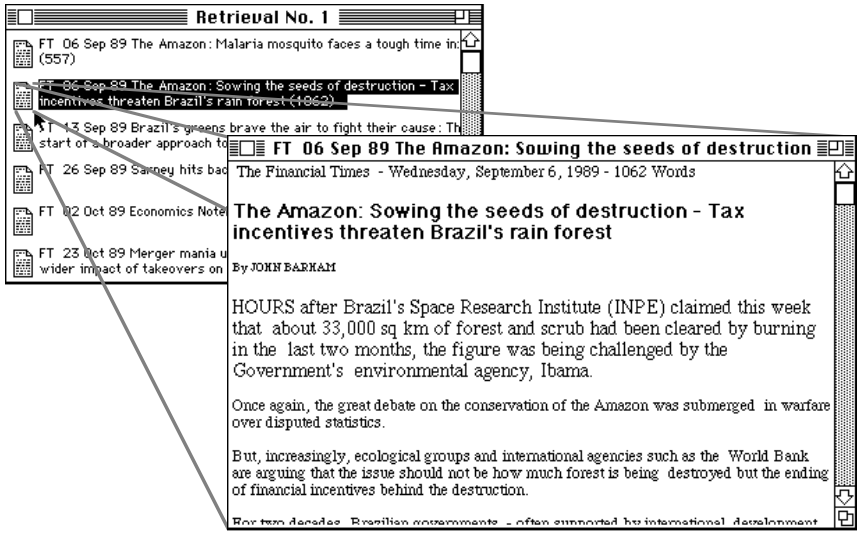

Figure 6.

If the user finds a document(s) that is relevant, then he marks it as relevant. To do this, the document has to be moved to the tick (relevant) window. This is done by the user pressing down the mouse button over the relevant document(s), dragging the document(s) to the tick window and releasing the mouse button (see Figure 7).). A copy of the document(s) appears in the tick window (see Figure 8).

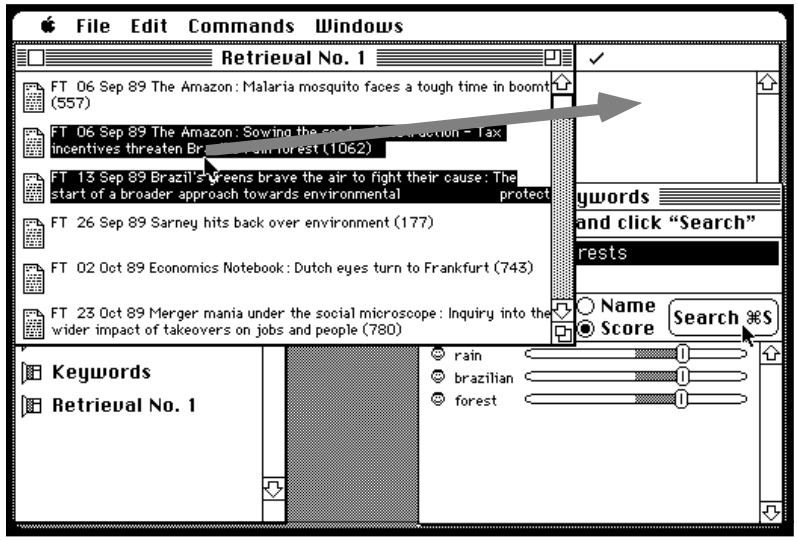

Figure 7.

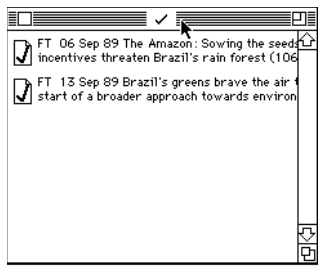

Figure 8.

A second search is now initiated by clicking the mouse on the "Search" button. Before the search takes place, NRT checks to see if the contents of the tick window have been changed since the last search. In this case it has, so NRT first analyses the documents in the tick window and adds new words to the Keywords window (see Figure 9), using relevance feedback.

After the new words are added, the search is performed. A new window with the retrieved documents appears. Not too surprisingly the documents marked as relevant appear near the top of the list. Some of the documents have blank icons. This indicates that the documents have already been retrieved and can be found in at least one of the other retrieval windows. A document with a text-filled icon indicates that this is the first time the document has been retrieved in this search (see Figure 10). The user can gauge the success of a retrieval by the number of blank or filled documents there are. A large number of blank icons indicates that the user may be reaching a dead-end in his search and that most of the relevant documents have been found.

Figure 9.

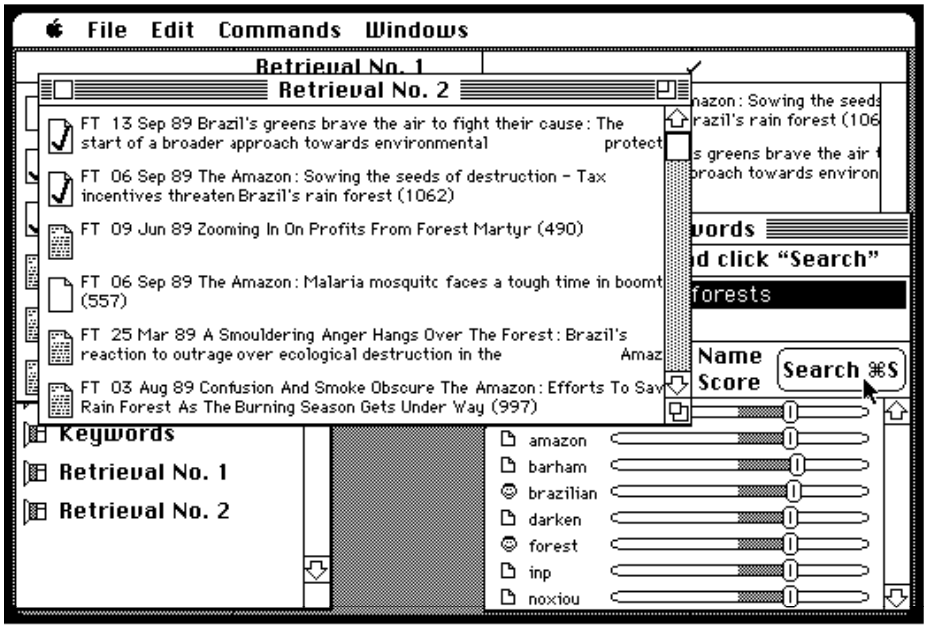

Figure 10.

2.3 The communications

Initially, the communications were done over a telephone line using a 1200 baud modem. Recently TCP/IP communications have been added to NRT to allow it to use a server that is running on a local ethernet. Because the communications accounted for most of the time taken to perform a search, the use of ethernet speeded up NRT dramatically, reducing the time for a search to be performed by a factor of 7. The ability to communicate using TCP/IP means that NRT is capable of using servers on wide area networks like the internet. To prove this capability, a server was set up by a colleague in Switzerland and put on the internet. The interface running in Glasgow successfully connected to the server and retrievals were performed. Although slow by local ethernet standards, the performance of the system was still adequate.

2.4 Testing

Some informal testing was conducted on a number of people who were experienced with the Profile service. Each person was asked to find documents on a number of topics and their answers were noted. Although the tests were informal, some general conclusions were drawn and they are as follows. The windowing interface was found to be a great success and much preferred to the Profile text-only interface. When searching, relevance feedback was found to work well and was liked by the testers. The only problem that was identified was that some testers seemed to have difficulty in formulating a good initial query. It is believed that the reason for this is that the testers were used to searching on the existing Profile service; however, NRT and Profile require different styles of query, Profile requiring small precise queries and NRT wanting longer fuzzier ones. While testing a term-based retrieval system, Dunlop [12] found that, over time, novice users would adapt their style of query to suit the system. However, to require Profile users to change their style of query is not a real solution and in Section 4 "Future work" a number of changes are outlined which will improve the NRT retrieval performance on this initial query.

One advantage of testing this system was that the collection, 130 000 articles from 1988 and 1989 of the *Financial Times*, was large enough and interesting enough for the testers to genuinely want to use it.

3 CONCLUSIONS

NRT has been judged a success. The combination of a better interface and an easy-to-use retrieval model has worked well. NRT has shown Profile that there is an alternative retrieval model to Boolean, although, in order to use it well, an interface better than text-only is required. Despite communication between the workstation and the retrieval engine going over a slow telephone line, NRT is very usable. The split of functionality between the two machines proved to be a good choice.

Relevance feedback proved to be the star of NRT. It was easy and intuitive to use and was successful at retrieving documents. Because the method of performing relevance feedback (dragging documents into a relevant window) is easy to do, it was found that many users were browsing the document collection using relevance feedback. This is a form of dynamic hypertext: the user indicates to the system which documents are of interest and then using relevance feedback; the system effectively creates a link (by performing a search) to related documents. In the case of large document collections where manually creating hypertext links would be too great a task, techniques like relevance feedback may prove to be very useful.

4 FUTURE WORK

Although the NRT project is finished, the authors believe that there is much scope for further work. What follows is a summary of the additions that would benefit NRT, for a more detailed explanation see Reference [2].

4.1 Time

It is obvious the user should be able to specify individual dates and date ranges in his query, but the issue of time in a retrieval system is more complex than it initially seems.

There are two main issues: the method used to incorporate time into a term-based retrieval system and how the interface will present time to the user.

When incorporating time into term-based retrieval, it should be used as a scoring mechanism like other keywords in a query; this would be a sort of fuzzy time. How the time and text part of a query is integrated will have to be determined by experimentation.

The user interface would be tackled by a combination of two approaches.

- The first approach is to allow the user to specify a date range by typing it in. In order to make the interface easier to use, the program would need to understand as many different formats of dates as possible (e.g. 01:04:89, 1/4/89, the 1st of April and April Fools' Day). This would mean creating a time conflater, an algorithm that reduces all specifications of time into a standard format. lf a conflater could be built to accept specifications like "last Easter" or "Mother's Day, ten years ago" then this would be a very powerful and accurate way of specifying dates.
- The second approach is to present a graph spanning the range of time that the database covers. The user draws the areas of time required. The graphical method isn't precise, but the dates can be clearly seen and the range can be easily adjusted after it has been specified.

4.2 Improvements to retrieval

Although relevance feedback has been shown to work well, the informal testing found that quite often the first search failed to find any relevant or partially relevant documents that the user could then use for relevance feedback. This is because users tend to enter very few words for their initial query and so the retrieved documents are ranked in a very coarse manner. For example, if a query is made up of two words, the retrieved documents would be sorted into three chunks, one chunk made up of documents containing both words, and the other two with documents containing only one of the words. The order of the documents within these chunks is not defined and this means that potentially relevant documents may be hidden within a very large chunk of a few hundred documents.

The best solution to this is to find ways of sorting the document chunks by exploiting additional properties of the query. Documents having these properties would percolate to the top of the chunks where are more likely to be seen by the user. The two obvious candidates for these properties are,

- If a document not only has a conflated word match with the query, but an exact character-for-character word match as well, then this document is likely to be more relevant.
- If the user types in phrase, and a document not only contains the words from that phrase but an exact match to that phrase, then again the document is likely to be more relevant.

Another potential solution to this problem comes from recognizing that a retrieved set of documents can usually be grouped into collections covering only a few subjects. It might be possible to cluster the retrieved documents, showing the user a sample from each cluster. If the user is interested in one of the clusters, the rest of the documents belonging to that cluster can then be shown.

ACKNOWLEDGEMENTS

The authors would like to thank the *FT* for its previous and continuing support of this work.

REFERENCES

1. J. Verhoeff, W. Goffman, and J. Belzer, 'Inefficiency of the use of the boolean functions for information retrieval systems', *Communications of the ACM*, **4**, 557–558, 594 (1961).
2. I. Campbell, M. Sanderson, and C. J. van Rijsbergen, 'NRT (technical notes)', Technical report, Department of Computing Science, University of Glasgow (1990).
3. C. Stanfill and K. Brewster, 'Parallel free-text search on the connection machine system', *Communications of the ACM*, **29**(12), 1229–1239 (1986).
4. C. J. van Rijsbergen, *Information Retrieval*, 2nd edn, Butterworths, London, 1979.
5. M. F. Porter, 'An algorithm for suffix stripping', *Program*, **14**(3), 130–137 (1980).
6. K. Sparck-Jones 'A statistical interpretation of term specificity and its application in retrieval', *Journal of Documentation*, **28**, 11–21 (1972).
7. S. E. Robertson and C. L. Thompson. Weighted searching: the CIRT experiment, March 1989. Paper presented at Informatics 10, York.
8. G. Salton, 'A simple blueprint for automatic boolean query processing', *Information Processing and Management*, **24**(3), 269–280 (1988).
9. R. H. Thompson, *The design and implementation of an intelligent interface for information retrieval*, PhD dissertation, Computer and Information Science Department, University of Massachusetts, 1988.
10. R. M. Stein, 'Browsing through the terabytes', *Byte*, **16**(5), 157–164 (1991).
11. M. Sanderson, 'NRT (news retrieval tool): A user's guide', Technical report, Department of Computing Science, University of Glasgow (1990).
12. M. Dunlop, *Multimedia information retrieval*, PhD dissertation, Department of Computing Science, University of Glasgow. To be published.