Query-Based Document Skimming: A User-Centred Evaluation of Relevance Profiling

David J. Harper, Ivan Koychev, Yixing Sun

Smart Web Technologies Centre, School of Computing
The Robert Gordon University
St Andrew Street, Aberdeen AB25 1HG, UK
E-mail: {djh,ik,sy}@comp.rgu.ac.uk

Abstract. We present a user-centred, task-oriented, comparative evaluation of two query-based document skimming tools. ProfileSkim bases within-document retrieval on computing a relevance profile for a document and query; FindSkim provides similar functionality to the web browser Find-command. A novel simulated work task was devised, where experiment participants are asked to identify (index) relevant pages of an electronic book, given subjects from the existing book index. This subject index provides the ground truth, against which the indexing results can be compared. Our major hypothesis was confirmed, namely ProfileSkim proved significantly more efficient than Find-Skim, as measured by time for task. Moreover, indexing task effectiveness, measured by typical IR measures, demonstrated that ProfileSkim was better than FindSkim in identifying relevant pages, although not significantly so. The experiments confirm the potential of relevance profiling to improve query-based document skimming, which should prove highly beneficial for users trying to identify relevant information within long documents.

1 Introduction

A user faced with finding textual information on the Web, or within a digital library, is faced with three challenges. First, the user must identify relevant repositories of digital text, usually in the form of document collections. In the context of the Web, this might be by identifying appropriate content portals, or by selecting appropriate search engine(s). Second, the user must find potentially relevant documents within the repository, usually through a combination of searching, navigating inter-document links, and browsing. Third, the user must locate relevant information *within* these documents. This paper is concerned with the latter challenge, which is becoming increasingly important as longer documents are published, and distributed, using Web and other technologies. Various approaches have been proposed for within-document retrieval, including passage retrieval [1], and user interfaces supporting content-based browsing of documents [2]. We have proposed a tool for within-document retrieval based on the concept of relevance profiling [3], and in this paper we report on a user-centred, comparative evaluation of this tool.

We have been working on the design, development and implementation of a tool called ProfileSkim, whose function is to enable users to identify, efficiently and effectively, *relevant passages* of text within *long* documents. The tool integrates passage retrieval and content-based document browsing. The key concept underpinning the tool is relevance profiling, in which a profile of retrieval status values is computed across a document in response to a query. Within the user interface, an interactive bar graph provides an overview of this profile, and through interaction with the graph the user can select and browse *in situ* potentially relevant passages within the document.

The evaluation study reported herein was devised to test key assumptions underlying the design of the ProfileSkim tool, namely:

- That relevance profiling, as implemented and presented by the tool, is *effective* in assisting users in identifying relevant passages of a document;
- That by using the tool, users will be able to select and browse relevant passages more *efficiently*, because only the best matching passages need be explored;
- That users will find the tool satisfying to use for within-document retrieval, because of the overview provided by relevance profiling.

We only report experimental results in support of the first two assumptions, which are based on quantitative data collected in the user study. In pursuit of evidence to test these two assumptions, we have conducted a comparative evaluation of two within-document retrieval tools, namely ProfileSkim, and FindSkim which provides similar functionality to the well-known Find-command delivered with most text processing and browsing applications. We investigate the tools within a simulated work task situation [4], in which the participants in the study are asked to compile (part of) a subject index for a book. Within this task setting, we evaluate the comparative effectiveness and efficiency of the within-document retrieval tools, where the task itself requires content-based skimming of a digital version of a book.

This evaluation study is based on an evaluation approach that is beginning to emerge through the efforts of the those involved in the ‘interactive track’ of TREC [5], through end user experiments in the Information Retrieval community [4] [6] [7], and through the effort of groups such as the EC Working Group on the evaluation of Multimedia Information Retrieval Applications (Mira) [8]. Major elements of the approach are:

- The observation of ‘real’ users engaged in the performance of ‘real-life’ tasks (or, at least, convincing simulations of such tasks);
- A range of performance criteria are used, pertaining both to quantitative aspects of task performance (efficiency and effectiveness), and qualitative aspects of the user experience;
- A range of methods for acquiring and analysis of data are used, which can be quantitative in nature (e.g. time for task), and qualitative in nature (e.g. attitudes and reactions to the system, the task, etc.).

The paper is structured as follows. In Section 2, we provide an overview of relevance profiling, and describe how language modelling can be used as a basis for this. An overview is provided in Section 3 of the salient features of the two within-document retrieval tools used in the study. The research questions are presented in section 4, and the experimental methods in section 5. In Section 6, we present the results of the experimental study, and these are discussed in Section 7. Finally, we offer some concluding remarks concerning the efficacy of relevance profiling as a basis for within-document retrieval, and we highlight the advantages of our particular approach for evaluating this type of retrieval tool.

2 Overview of Relevance Profiling based on Language Modelling

Relevance profiling using language modelling was introduced in [3], and we provide a brief overview here. Based on a query, we want to compute a relevance profile across the document, and presented this profile to the user in the form of a bar graph. By interacting with this bar graph, the user can identify, and navigate to, relevant sections of a document. Effectively, a retrieval status value (RSV) is computed for each word position in the document. This RSV will be based on a ***text window*** (fixed number of consecutive words) associated with each word position. Language modelling is used to construct a statistical model for a text window, and based on this model we compute the window RSV as the probability of generating a query.

We employ the language modelling approach proposed for document retrieval in [9] [10], and adapt it for relevance profiling. We model the distribution of terms (actually stemmed words) over a text window, as a mixture of the text window and document term distributions as follows:

$$\mathrm{P}(query \mid window) = \prod_{t_i \in query} p_{mix}(t_i \mid win) \qquad \textbf{(1)}$$

where: $p_{mix}(t_i \mid win) = w_{win} * p_{win}(t_i \mid win) + (1\text{-}\, w_{win}) * p_{doc}(t_i \mid doc)$

Thus, the probability of generating words is determined in part by the text window, and in part by the document in which the window is located. The estimates are smoothed by the document word statistics using the mixing parameter, w_{win}. The best value for this parameter needs to be determined empirically, and we have used 0.8 in our system. The individual word probabilities are estimated in the obvious way using maximum likelihood estimators:

$$p_{win}(t_i \mid win) = n_{iW} / n_W \quad p_{doc}(t_i \mid doc) = n_{iD} / n_D \qquad \textbf{(2)}$$

where n_{iW} (n_{iD}) and n_W (n_D), are the number of word occurrences of word *i* in the window (document), and total word occurrences in the window (document) respectively.

The relevance profile is given by the retrieval status value at each word position *i*:

$$RSV_{window}(i) = \mathrm{P}(query \mid window_i) \qquad \textbf{(3)}$$

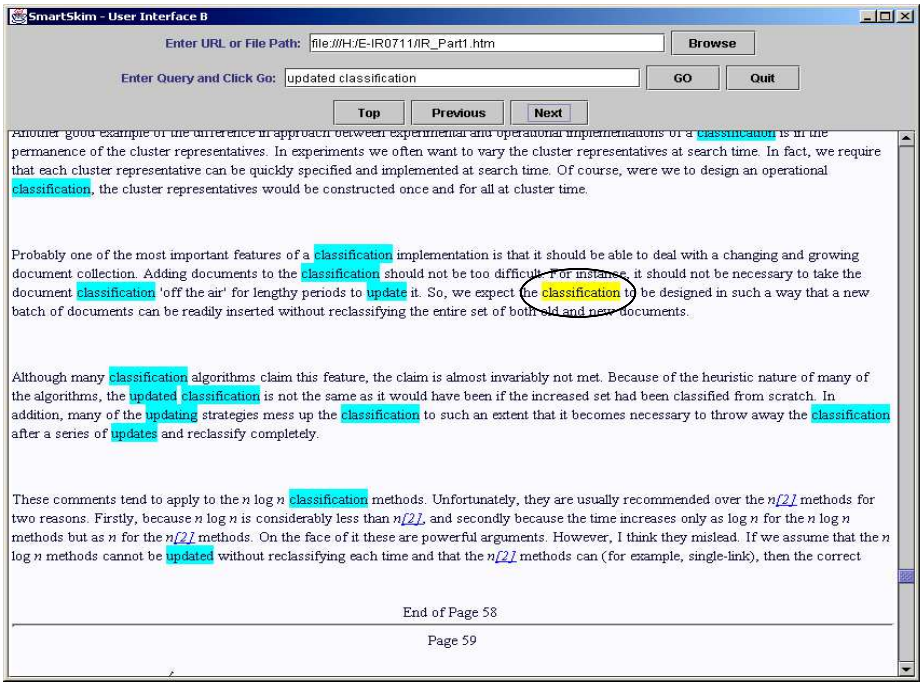
where text window *i* is the sequence of words [$w_i..w_i+L_W\text{-}1$], and L_W is the fixed length of each text window.

In order to provide a plot of the relevance profile, and to support direct navigation to relevant parts of a document, retrieval status values are aggregated over fixed size, non-overlapping sections of text we call ***text tiles***. We assume that the document text is divided into fixed length, non-overlapping text tiles. Let us assume that each tile is L_T words long. The aggregate RSV for a given tile *j* is given by:

$$RSV_{tile}(j) = agg\text{-}fun(\{RSV_{window}(i), i = (j\text{-}1) * L_T + 1 \,..\, j * L_T\}) \qquad \textbf{(4)}$$

Examples of aggregate functions (agg-fun) include average, minimum and maximum, and we opt for the maximum as this corresponds to the best text window starting within the tile. Note that some text windows will extend beyond the end of a tile.

Text windows and text tiles, although related, serve two different purposes. A text window is used to compute an RSV at each word position in the document. The fixed size of a text window is set to the “typical” size of a meaningful chunk of text, such as the average size of a paragraph (or possibly section). The average size of a paragraph can be determined empirically, and in our system we have set it to 200 words. A text tile is used to aggregate or combine the RSVs of ***all*** text windows that ***start*** within the given tile, and tiles are used for summarizing (and thence displaying) relevance profiles. The size of a fixed tile is computed based on the length of the document, and depends on the number of tiles, and hence bars, we wish to display in the relevance profile meter. The heights of the bars in the profile meter are proportional to the tile RSV, and are based on logarithm of the tile RSV (see [3] for reasons).



Enter URL or File Path: file:///H:/E-IR0711/IR_Part1.htm

Enter Query and Click Go: updated classification

Another good example of the difference in approach between experimental and operational implementations of a classification is in the permanence of the cluster representatives. In experiments we often want to vary the cluster representatives at search time. In fact, we require that each cluster representative can be quickly specified and implemented at search time. Of course, were we to design an operational classification, the cluster representatives would be constructed once and for all at cluster time.

Probably one of the most important features of a classification implementation is that it should be able to deal with a changing and growing document collection. Adding documents to the classification should not be too difficult. For instance, it should not be necessary to take the document classification 'off the air' for lengthy periods to update it. So, we expect the classification to be designed in such a way that a new batch of documents can be readily inserted without reclassifying the entire set of both old and new documents.

Although many classification algorithms claim this feature, the claim is almost invariably not met. Because of the heuristic nature of many of the algorithms, the updated classification is not the same as it would have been if the increased set had been classified from scratch. In addition, many of the updating strategies mess up the classification to such an extent that it becomes necessary to throw away the classification after a series of updates and reclassify completely.

These comments tend to apply to the *n* log *n* classification methods. Unfortunately, they are usually recommended over the *n[2]* methods for two reasons. Firstly, because *n* log *n* is considerably less than *n[2]*, and secondly because the time increases only as log *n* for the *n* log *n* methods but as *n* for the *n[2]* methods. On the face of it these are powerful arguments. However, I think they mislead. If we assume that the *n* log *n* methods cannot be updated without reclassifying each time and that the *n[2]* methods can (for example, single-link), then the correct

End of Page 58

Page 59

Fig. 1. Screen shot for FindSkim Tool.

3 The Document Skimming Tools

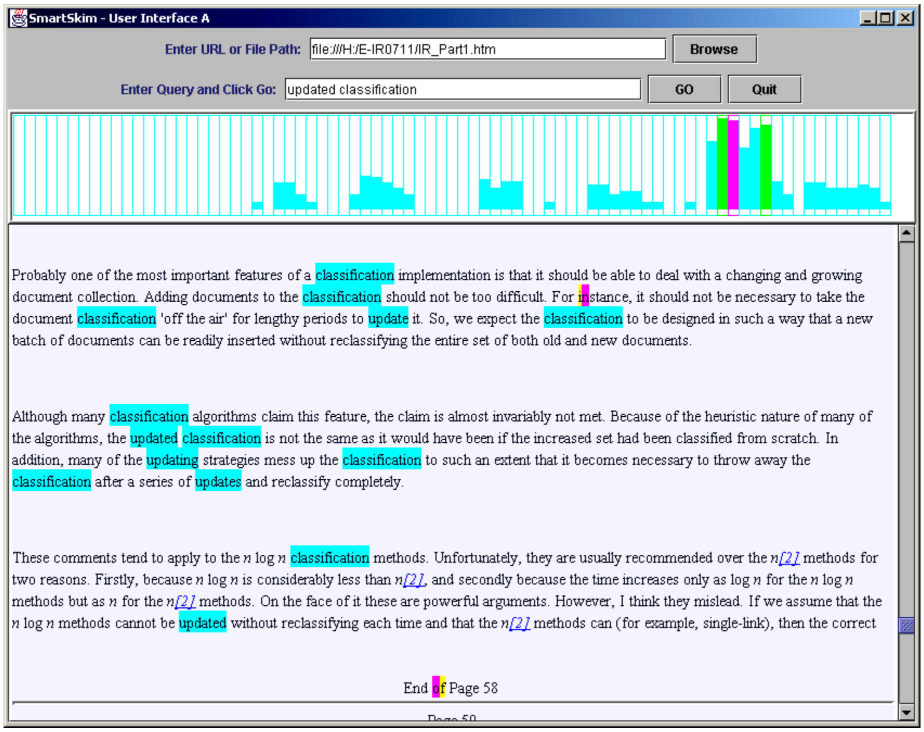
Two within-document retrieval tools are used in the comparative user evaluation. One, ProfileSkim, is based on relevance profiling, and the other, FindSkim, is based on the ubiquitous Find-Command provided within most word processing and web browser applications. FindSkim will be described first, as much of its functionality is common to both tools. Then, ProfileSkim is described.

3.1 The FindSkim Tool

The FindSkim tool is based on the Find-command, although in many respects it provides additional functionality. A screenshot of the tool is illustrated in **Fig. 1**.

A user selects a file to skim, using the file chooser, and the file is displayed in a scrollable panel. Given a query, the tool highlights all query word variants that appear in the document in cyan. The document is positioned in the display panel at the first word occurrence, which becomes the *current word*. The current word is always highlighted in yellow (circled in **Fig. 1**.). The user can navigate from the current word to the next (or previous) query word occurrence in the document using the Next/Find buttons. Query words which are not present in the document are flagged as possible misspellings, and the user may choose to edit the query, if appropriate.

Note, that the query is treated as a "bag of words". Hence, no phrase matching is performed based on query word order.



Enter URL or File Path: file:///H:/E-IR0711/IR_Part1.htm

Enter Query and Click Go: updated classification

Probably one of the most important features of a classification implementation is that it should be able to deal with a changing and growing document collection. Adding documents to the classification should not be too difficult. For instance, it should not be necessary to take the document classification 'off the air' for lengthy periods to update it. So, we expect the classification to be designed in such a way that a new batch of documents can be readily inserted without reclassifying the entire set of both old and new documents.

Although many classification algorithms claim this feature, the claim is almost invariably not met. Because of the heuristic nature of many of the algorithms, the updated classification is not the same as it would have been if the increased set had been classified from scratch. In addition, many of the updating strategies mess up the classification to such an extent that it becomes necessary to throw away the classification after a series of updates and reclassify completely.

These comments tend to apply to the *n* log *n* classification methods. Unfortunately, they are usually recommended over the *n[2]* methods for two reasons. Firstly, because *n* log *n* is considerably less than *n[2]*, and secondly because the time increases only as log *n* for the *n* log *n* methods but as *n* for the *n[2]* methods. On the face of it these are powerful arguments. However, I think they mislead. If we assume that the *n* log *n* methods cannot be updated without reclassifying each time and that the *n[2]* methods can (for example, single-link), then the correct

End of Page 58

Page 59

Fig. 2. Screen shot for ProfileSkim Tool.

3.2 The ProfileSkim Tool

The ProfileSkim tool is based on relevance profiling, and displays an interactive analogue of the relevance profile for a given query, in the form on a bar graph. A screenshot of the tool is illustrated in **Fig. 2**.

File selection and query input are identical to the FindSkim tool. Query term variants are also highlighted in cyan, and the document is displayed in a scrollable panel.

Based on a query input by the user, a relevance profile is computed over the document (see Section 2), and presented in the form of an interactive bar graph. Each bar corresponds to a fixed length section (tile) in the text of the document, with the leftmost bar corresponding to the start of the document, and the rightmost bar to the end of the document. The height of a bar corresponds to the computed retrieval status value of the corresponding tile. By clicking on a bar, the corresponding tile within the document is centred in the document viewer. Effectively, the bars of the relevance profile meter act as “hypertext links” into the body of the document.

To assist the user in browsing the document using the relevance profile meter, feedback is provided as to which bars (and corresponding tiles) had been visited. Colour coding of the bars indicates which bar/tile has: yet to be visited (cyan), currently being visited (magenta) and visited (green). This colour-coding scheme reinforces the view that the bars acts as hypertext links, and the colours used correspond broadly to those used typically when browsing web pages. The currently visited tile is also indicated with yellow/magenta and magenta/yellow “brackets” on the document display.

A critique of the ProfileSkim interface using the Cognitive Dimensions Framework [11] is provided in [3].

3.3 Choice of skimming tools

In setting up the comparative user evaluation of ProfileSkim, we gave careful thought to the choice of the other skimming tool.

We opted for a tool based on the Find-command for three reasons. First, the Find-command is the *de facto* standard for document skimming, albeit in a number of guises in word processing applications and web browsers. Relevance profiling is a possible alternative to the Find-function, and it is therefore useful to provide comparative performance data. Second, we wanted to measure the relative performance of ProfileSkim against FindSkim to provide a benchmark for future developments of ProfileSkim itself. Third, developing our own Find-command variation might suggest ways of improving the Find-command itself.

We accept that the functionality of the tools is different, and in particular that additional information is made available to the users through the relevance profiling tool. However, we thought is best to establish the comparative performance of ProfileSkim against a *de facto* standard in the first instance, and investigate possible variants of relevance profiling tools at a later stage.

4 Research Questions and Hypotheses

In general terms, we wanted investigate whether within-document retrieval based on relevance profiling was more efficient in user time, and more effective in identifying relevant sections of long documents, than the competing tool based in functionality similar to the Find-command. Specifically, the user experiment was designed to test both user efficiency, and user effectiveness in performing the book indexing task. The effectiveness measures we use are described in Section 6.4.

More formally, a number of hypotheses were formulated, based on the expected performance of ProfileSkim and FindSkim. These are, with justifications:

Hypothesis HT: *That 'time to complete' the indexing task would be less using ProfileSkim compared with FindSkim (one-tailed).*

We expected that the relevance profile meter would enable the user to readily identify relevant sections of the text, and importantly not spend time browsing less relevant sections.

Hypothesis HP: *ProfileSkim is more effective than FindSkim as measured by Precision (one tailed).*

Hypothesis HP is based on the observation that ProfileSkim encourages a user to explore the highest peaks of the relevance profile (potential relevance hotspots), and thus we might expect a user to achieve higher precision when using ProfileSkim.

Hypothesis HR: *FindSkim is more effective than ProfileSkim as measured by Recall (one tailed).*

Hypothesis HR is based on the observation that FindSkim encourages a user to visit all query word occurrences in the text and thus we might expect a user to achieve higher recall, and this possibly at the expense of precision. However, it is possible that ProfileSkim might achieve comparable levels of recall, depending on the extent to which a user is prepared to explore comprehensively the relevance profile.

Conjecture CF: *Supposing that hypotheses* **HP** *and* **HR** *hold, then we conjecture that effectiveness, as measured by the combined F-measure, will be comparable.*

This conjecture is simply a consequence of the fact that the F-measure "trades off" precision against recall.

5 Methods

In this evaluation of within-document retrieval using relevance profiling, and specifically the comparative evaluation of ProfileSkim and FindSkim, we wanted to address the following issues:

- the participants in the experiment should be placed in a simulated work task situation [4], such that document skimming is central in performing the task;
- the focus of the task should be document skimming, and not document retrieval;
- the documents used in the study should be long, in order to provide a realistic assessment of the tools being studied;
- the tasks should be realistic, understandable to the participants, and able to be completed in a reasonable time; and
- task performance can be measured against some ground truth established for the task.

A novel work task situation was devised that satisfied our requirements, namely creating a subject index for an electronic book.

5.1 Participants

The participants for the study were all graduate students drawn from various places in our University. We would have preferred to select from a homogeneous group, but this was not possible given that the experiment was performed with 24 participants (plus 6 additional participants for the pilot). Instead, we selected from a number of programmes, namely students in: MSc Information and Library Studies (10), MSc Knowledge Management (7), MSc Electronic Information Management (2), PhD in Business Studies (1) and PhD in Computing (4). Based on the entry questionnaire, the participants were mostly unfamiliar with the field of information retrieval, and hence the (electronic) book used in the study. They had on average of 3.8 years of experience in using computers for reading/browsing electronic text.

5.2 Instruments

Collection. An electronic version of van Rijsbergen's classic information retrieval text was obtained, and we added page numbers which are necessary in creating a subject index. The book was divided into four sections, two sections for training and two for the main experiment (see **Table 1**).

Filename	*Content*	*No of Pages*	*Word Count*
Training1	Chapter 4	29	9526
Training2	Chapter 7	40	13181
Part1	Chapter 2, 3	52	18087
Part2	Chapter 5, 6	49	17296

Table 1. Collection Details

Topics. Eight topics[1] were selected at random from the subject index provided with the original textbook (see **Table 2**). The selected topics met the following criteria:

- between 4 and 7 pages indexed for the topic;
- at least two distinct ranges of page numbers;
- two or more words for the topic;
- (preferably) indexed pages present in both Part 1 and Part 2 of the text; and
- (as far as possible) minimize overlap between the pages for the different topics.

These criteria ensured that the corresponding indexing tasks could be performed in a reasonable time, and that the participants would be required to browse comprehensively both parts of the book. We opted for multi-word topics for two reasons. First, we were interested in assessing the benefits of relevance profiling in a more general setting, e.g. skimming documents retrieved by search engines, and multi-word queries are more typical in this setting. Second, relevance profiling is not particularly interesting for one word queries, as it equates to a simple count of word occurrences. The fi-

[1] Although we normally refer to 'subject indexing' and 'subjects' for books, we will adopt the standard IR terminology of 'topic indexing' and 'topic' in this paper.

nal criterion was included to try and minimize the learning effect of viewing many times the same, albeit, long document.

5.3 Procedures

Scenario for Simulated Work Task. The experiment participants were asked to imagine they were graduate students, who had been asked by their tutor to assist him/her in creating a subject index for a book he/she has written. For a given topic they were asked to locate pages that should appear under that topic, using one of the skimming tools. The criteria for including a page, i.e. assessing the page relevant for the topic, were:

- the page must be topically relevant, i.e. about the subject;
- the page must be substantially relevant, i.e. the page would add to a potential reader's understanding of the topic;
- all pages in a set of contiguous relevant pages should be included; and
- pages in the bibliographies at the ends of chapter were not to be indexed.

These instructions accorded in general with the way the book was originally indexed by the author (Private communication from C. J. van Rijsbergen).

Task group	*Order*	*Topic/Subject*	*File to Skim*	*Indexed Pages*
1	Training	Expected Search Length	Training1	
			Training2	160-163
	First	Loss (or Cost) Function	Part1	29
			Part2	116-117, 126
	Second	Boolean Search	Part1	
			Part2	95-97, 109
	Third	Information Measure	Part1	41-42, 57
			Part2	123, 136, 138
2	Training	Relational Data Model	Training1	67, 90
			Training2	
	First	Maximum Spanning Tree (MST)	Part1	56, 57
			Part2	123, 132, 139
	Second	Relevance Feedback	Part1	
			Part2	105-108, 112
	Third	Cluster based Retrieval	Part1	47, 56
			Part2	103-105

Table 2. Indexing task groups

Tasks and Task Groups. Each topic was the basis for an indexing task, and to assist the participants, a short definition was provided for each topic. This provided some context for evaluating the relevance of page to a topic, and plays a similar role to the extended topic descriptions in TREC-1 [13]. The topics were divided into two groups for the experimental design, and we refer to these as Task Groups (see **Table 2**). Within each task group, the first task was used as a training task, and the other three tasks were arranged in increasing order of difficulty. This ordering was established based on a pilot study we performed.

Experiment Design. The design is summarised in **Table 3**.

Participant Group	*First Task Set (System/Task Group)*	*Second Task Set (System / Task Group)*
1	A / TG1	B / TG2
2	A / TG2	B / TG1
3	B / TG1	A / TG2
4	B / TG2	A / TG1

Table 3. Experiment Design

Experiment Procedure. The procedure is summarised in **Fig. 3**.

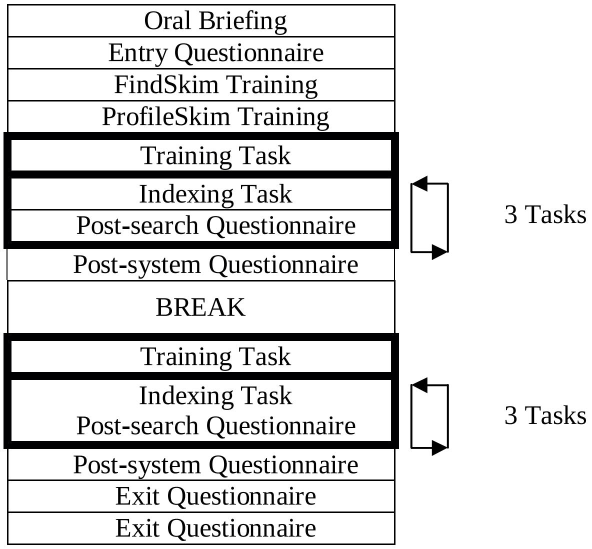


Fig. 3. Procedure for Experiment

The participants were asked to complete the indexing tasks as quickly as possible, while at the same time achieving good levels of indexing specificity and exhaustivity. The pilot study established that most tasks could be completed in 6-10 minutes, and thus we allocated 40 minutes for each task group. However, the participants were asked to complete all tasks in a group, even if they over-ran the allocated time. The majority of participants completed each task group within the 40 minutes.

A few observations are necessary regarding this procedure. We would have preferred to run the experiment with each participant individually. This was not possible due to timetabling and resource constraints. However, we minimised as far as possible interaction between the participants. We would have preferred to do the system training just prior to use of each system. This was not possible given the experiment was performed with participants from all participant groups (see **Table 3**). In mitigation, the training was mostly concerned with task training, as the systems were relatively easy to learn and use. Moreover, prior to using each system, there was a specific training task.

5.4 Measures

For each indexing task, allocated one at a time, the user was asked to record the page numbers of relevant pages they would include in the topic (subject) index. Using this information, we were able to assess the specificity and exhaustivity of the indexing, using traditional precision and recall measures (see below). The time for each task was recorded in minutes and seconds. Using this information, we were able to assess the user efficiency of the indexing process.

Precision, recall and the F-measure were computed as follows. The original subject index of the book provides the ground truth for the indexing tasks. That is, the pages indexed originally by the author of the book, are effectively the pages deemed relevant. Hence for a given subject, if A is the set of pages indexed by the author and B is the set of pages indexed by a participant in the study, then precision and recall can be computed in the obvious way:

$$P = |A \cap B| / |B| \qquad R = |A \cap B| / |A| \qquad \textbf{(5)}$$

The F-measure, which is a single measure of performance, is simply the harmonic mean of precision and recall, namely:

$$F = 2 * P * R / (P + R) \qquad \textbf{(6)}$$

This measure effectively “values” precision and recall equally, and thus it enables us to trade off precision and recall.

6 Experimental Results

In this paper, we will focus on presenting and analysing the quantitative data, as this data is the focus of the major hypotheses of the experimental study. Thus, we concentrate on presenting and analysing data relating to task efficiency, as measured by time for task, and task effectiveness, as measured by precision, recall, and F-measure.

In **Table 4**, the average time for task completion is given for each system. The average time for ProfileSkim and FindSkim is 5.76 and 7.74 minutes respectively, and this result is statistically significant at the level of p<0.001. The average effectiveness measures are presented for ProfileSkim and FindSkim. On average, precision, recall and F-measure are all higher for ProfileSkim compared with FindSkim. However, in no instance are these results significant at the level of $p<0.05$.

The boxplots in **Fig. 4** show the spread of the measures for ‘time for task’, precision, recall and F-Measure, for the ProfileSkim tool (System A) and the FindSkim tool (System B). These plots show that ProfileSkim is better than FindSkim with respect of ‘time for task completion’. The task effectiveness, as measured by precision, recall and F-measure, are also better for ProfileSkim, although less markedly so than for the ‘time for task’.

	Mean (Variance)		*T-statistic*	*P(T<=t) one-tail*
	ProfileSkim	*FindSkim*		
Time	5.8076 (2.4553)	7.7435 (3.7676)	3.5688	0.0008
Precision	0.6224 (0.0237)	0.5503 (0.0126)	1.6962	0.0517
Recall	0.7394 (0.0288)	0.6869 (0.0538)	0.8417	0.2043
F	0.6354 (0.0178)	0.5819 (0.0225)	1.0863	0.1443

Table 4. Summary of experimental results analysis *(DF=23; t Critical one-tail*$_{(0.05)}$ *= 1.7139)*

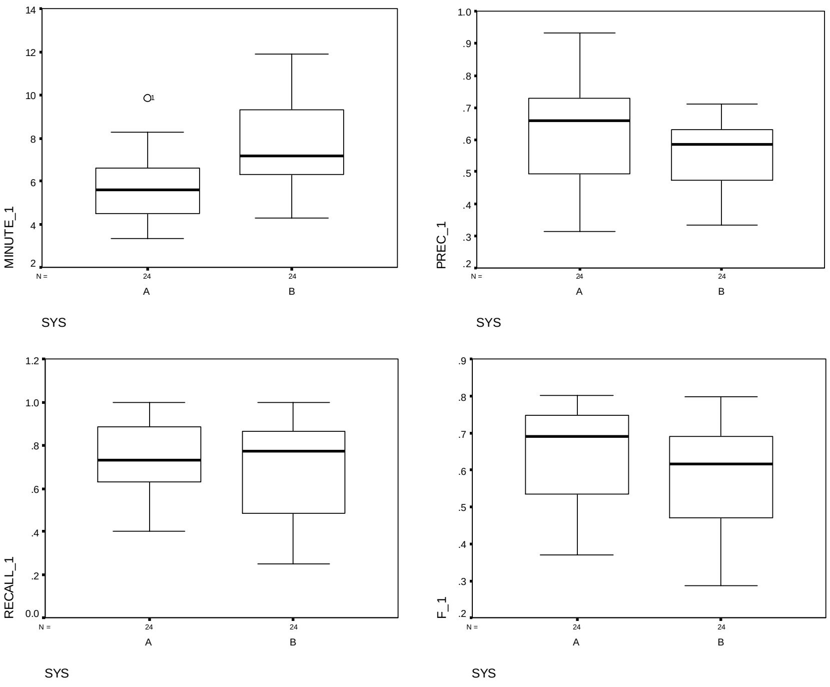


Fig. 4. The Boxplots for Time, Precision, Recall and F for ProfileSkim (A) and FindSkim (B)

7 Discussion of Results

In this experiment, we investigated within-document retrieval tools when used in a simulated subject indexing of a book task. Our results provide evidence that relevance profiling, as presented and implemented in ProfileSkim, is more efficient than the FindSkim for the book indexing task. The average time for ProfileSkim and FindSkim is 5.76 and 7.74 minutes respectively, and this result is statistically significant [p<0.001]. Hence, we fail to accept the null hypothesis corresponding to HT, and our results provide very strong evidence:

That 'time to complete' the indexing task is less using ProfileSkim compared with FindSkim.

In respect of task effectiveness, the general trend suggests that ProfileSkim (PS) is more effective than FindSkim (FS) when measured by Precision (PS: 0.6224, FS: 0.5503), Recall (PS: 0.7394, FS: 0.6869) and the F-measure (PS: 0.6354, FS: 0.5819). However, in no case are the differences statistically significant at the level p<0.05. And, we fail to accept the hypotheses **HP** and **HR**, namely:

ProfileSkim is more effective than FindSkim as measured by Precision and

FindSkim is more effective than ProfileSkim as measured by Recall.

But, while the difference in Precision is not significant at the level p<0.05, it is significant at the slightly higher level of p<0.06. There is therefore weaker evidence that ProfileSkim is more effective that FindSkim as measured by Precision and one might tentatively conclude that relevance profiling is a precision-oriented device.

The F-measure results provide evidence for our conjecture, namely that overall effectiveness of ProfileSkim and FindSkim is comparable when used for the book indexing task. In summary, our results indicate that relevance profiling, as realised in ProfileSkim, is more efficient that FindSkim, and moreover this efficiency is achieved with no significant difference is indexing effectiveness, as measured by the F-measure. Furthermore, the absolute level of performance is pleasingly high, especially given that the indexing task was perceived by the users to be difficult as assessed through the questionnaires.

Given these results, what can we conclude about the efficiency and effectiveness of ProfileSkim, and by implication relevance profiling, for more general within-document retrieval tasks. That is, to what extent will these results carry over into other task settings and situations? The experiment task required the participants to locate relevant sections of long documents using the tools. In particular, given the efficiency of ProfileSkim for the task, we can conclude that it is likely to be equally efficient in more general document browsing settings. Relevance profiling could be usefully provided within word processing applications and document reading/browsing tools as a replacement for the commonly provided "Find" functionality.

The performance of ProfileSkim for the book indexing task, as measured by precision, was better than that of FindSkim, albeit at the slightly higher level p<0.06 than is usually accepted (p<0.05). This provides some evidence that relevance profiling is a precision-enhancing device. Thus, relevance profiling may be valuable in within-document retrieval tasks that require high precision, tasks such as question-answering. ProfileSkim is able to accurately pinpoint relevant sections of large text documents, and to do so using relatively short queries. These are characteristic of many question-answering tasks.

The simulated work task situation we used in our experiment, namely the book indexing task, proved highly successful in many respects. Preliminary analysis of the task questionnaire data shows that the scenario and task were understood by the participants, although admittedly the participants were all postgraduates. The participants were able to perform the tasks both efficiently and effectively, as evidenced by the performance analysis. Importantly, the experiment clearly explored within-document retrieval, as this was central to the indexing task.

The book indexing task provides a ready-made ground truth, namely the original subject index. Certainly, it would not always be straightforward to ascertain the original indexing policy, and incorporate this within the experiment setting. Nevertheless, the book index provides a useful starting point.

Our experience provides strong evidence that the book indexing task is highly suited to evaluating within-document retrieval. The subject matter of the book is critical, and we were fortunate that our participants were able to comprehend the relatively technical material we used. The provision of both the subject (topic) and a longer definition proved important is enabling these participants to make the necessary relevance assessments. It may be that using more assessable materials, such as general-interest reference books, e.g. an encyclopaedia, would make the task simpler for participants drawn from a wider population.

8 Conclusions and Future Work

In this paper, we have reported the results of a user-centred evaluation of within-document retrieval tools, in the simulated task of providing (part of) the subject index of an electronic book. Two tools were compared, one based on relevance profiling (ProfileSkim), and one based on a sequential search (FindSkim).

The major findings of our investigation are that, for the book indexing task:

- The ‘time to complete’ the task is significantly less with ProfileSkim than with FindSkim;
- While the results were not statistically significant, the general trend is that indexing effectiveness, as measured by traditional information retrieval measures, is on average better when using ProfileSkim compared with FindSkim; and
- The indexing effectiveness, as measured by precision is better for ProfileSkim than FindSkim, at the reduced standard of $p<0.06$.

Thus, a within-document retrieval tool based on relevance profiling is both efficient and effective for the book indexing task. We argued that there is ample justification for believing that these findings will hold in more general task settings, in which document skimming may be useful. Further, relevance profiling should prove a worthy replacement for the familiar Find-Command implemented in most text processing and/or browsing applications.

The book indexing task proved highly satisfactory for evaluating the comparative performance of within-document retrieval tools, and based on our experiences, we would advocate its use for this kind of study. Arguably, an experimenter might need to choose the subject matter of the books carefully, depending on the background of the study participants, and indeed the indexing task may prove too taxing for some.

Relevance profiling on ProfileSkim is based on a relatively simple mixture language model. This model favours term frequency over term discrimination. We would like to investigate other possible formulations of relevance profiling, based on more advanced divergence models, which we believe would allow term frequency to be combined with term discrimination c.f. tf*idf weighting. We would expect to evaluate alternative relevance profiling approaches using the book indexing data, albeit in a batch environment, i.e. without user involvement, at least initially.

Acknowledgements

We would like to thank C. J. "Keith" van Rijsbergen for permission to use the electronic version of his textbook "Information Retrieval" in our experiment. We are indebted to the following people for advice on experimental design and analysis: Diane Kelly, Alex Wilson, Anna Conniff, Stuart Watt, Ayse Goker, and Peter Lowit. We would also like to thank Robert Newton and Alan Maclennan for volunteering their MSc students for the study. Finally, we would like to offer our heartfelt gratitude to the 30 participants who gave freely of their time, and strived so hard to find "relevance feedback" in Part 1 of the book! Finally, we would like to thank the anonymous referees for these helpful comments.

References

1. Kaszkiel, M., Zobel, J.: Passage Retrieval Revisited. In: Proceedings of the Twentieth International ACM-SIGIR Conference on Research and Development in Information Retrieval. Philadelphia. ACM Press (1997) 178-185
2. Hearst, M.A.: TileBars: Visualization of Term Distribution Information in Full Text Information Access. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), Denver, CO, (1995)
3. Harper, D.J., Coulthard, S., Sun, Y.: A Language Modelling Approach to Relevance Profiling for Document Browsing. In: Proceedings of the Joint Conference on Digital Libraries. Oregon, USA (2002) 76-83,
4. Borlund, P., Ingwersen, P.: The Development of a Method for the Evaluation of Interactive Information Retrieval Systems. Journal of Documentation. 53(3) (1997) 225-250
5. Beaulieu, M., Robertson, S.E. and Rasmussen, E.: Evaluating interactive systems in TREC. Journal of the American Society for Information Science. 47(1) (1996) 85-94
6. Hersh, W., Pentecost, J., Hickam, D.: A Task-Oriented Approach to Information Retrieval Evaluation. Journal of the American Society for Information Science 47(1) (1996) 50-56
7. Jose, J., Furner, J., Harper, D.J.: Spatial Querying for Image Retrieval: A User-Oriented Evaluation. In: Proceedings of the Twenty First International ACM-SIGIR Conference on Research and Development in Information Retrieval, ACM Press (1998) 232-240
8. Dunlop, M. (ed):. Proceedings of the Second Mira Workshop, Technical Report TR-1997-2. Department of Computing Science, University of Glasgow, Glasgow (1996). Available online at URL: http://wu.dcs.gla.ac.uk/mira/workshops/padua_procs/.
9. Ponte, J., Croft, W.B.: A Language Modeling Approach to Information Retrieval. In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press (1998) 275-281
10. Song, F., Croft, W.B.: A General Language Model for Information Retrieval. In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press (1999) 279-280
11. Green, T.R.G.: Describing Information Artifacts with Cognitive Dimensions and Structure Maps. In: Diaper, D., Hammond, N.V. (eds.): Proceedings of the HCI'91 Conference on People and Computers VI. Cambridge University Press, Cambridge (1991)
12. Hersh, W.R., Over, P.: TREC 2001 Interactive Track Structure, Proceedings of the Text Retrieval Conference (TREC) 2001, Gaithersburg, MD, (2001) 38-41
13. Harman, D.: Overview of the First Text REtrieval Conference (TREC-1), National Institute of Standards and Technology, Gaithersburg, Maryland (1992) 309-318