Ranked-Listed or Categorized Results in IR 2 Is Better Than 1

Zheng Zhu[1], Ingemar J. Cox[2], and Mark Levene[1]

[1] School of Computer Science and Information Systems,
Birkbeck College, University of London
zheng@dcs.bbk.ac.uk, mark@dcs.bbk.ac.uk
[2] Department of Computer Science, University College London
ingemar@ieee.org

Abstract. In this paper we examine the performance of both ranked-listed and categorized results in the context of known-item search (target testing). Performance of known-item search is easy to quantify based on the number of examined documents and class descriptions. Results are reported on a subset of the Open Directory classification hierarchy, which enable us to control the error rate and investigate how performance degrades with error. Three types of simulated user model are identified together with the two operating scenarios of correct and incorrect classification. Extensive empirical testing reveals that in the ideal scenario, i.e. perfect classification by both human and machine, a category-based system significantly outperforms a ranked list for all but the best queries, i.e. queries for which the target document was initially retrieved in the top-5. When either human or machine error occurs, and the user performs a search strategy that is exclusively category based, then performance is much worse than for a ranked list. However, most interestingly, if the user follows a hybrid strategy of first looking in the expected category and then reverting to a ranked list if the target is absent, then performance can remain significantly better than for a ranked list, even with misclassification rates as high as 30%. We also observe that this hybrid strategy results in performance degradations that degrade gracefully with error rate.

1 Introduction

Search engines play a crucial role in information retrieval on the web. Given a query, search engines, such as Google, Yahoo! and Windows Live, return a ranked list of results, referred to as the result set. For many queries, the result set includes documents on a variety of topics, rather than a single topic. This variation is often due to ambiguous queries. For example, the query "Jaguar" will often return documents referring to both the car and the animal. While the user is only interested in one topic, it is not possible for the search engine to know which topic is relevant based on the query alone. Moreover, the standard ranking of the documents in the result set is independent of the topic. Thus, the rank-ordered result set has an arbitrary topic ordering. Referring to the "Jaguar" example, this means that a user must scroll through a ranked list in which many documents are not relevant.

E. Kapetanios, V. Sugumaran, M. Spiliopoulou (Eds.): NLDB 2008, LNCS 5039, pp. 111–123, 2008.

There have been several proposals [1, 2, 3] to assist the user by organising the documents in the result set into groups, all documents within a group referring to a common topic. Thus, for the query "Jaguar", a user might be shown two distinct groups of documents, one referring to the animal and the other referring to the car. A user can immediately ignore the non-relevant topic and focus his attention only on the relevant topic. For this simple example, this grouping may, on average, halve the number of documents the user must examine.

Intuitively, we would expect grouping to substantially reduce search time[4], where search time is measured by the number of documents a user must examine before finding the desired document. Although previous researchers have evaluated their prototype systems, there has been no attempt, to our knowledge, of formulating a generic user interaction model for a retrieval system, which allows the benefits of grouping to be quantified in comparison to a standard retrieval system which does not group its results.

In this paper, we attempt to quantify the benefits of grouping documents based on classification, where for demonstration and ground-truth purposes we make use of the Open Directory (dmoz)[1], a large and comprehensive human-edited directory on the web. However, we note that our experimental approach may be applied to any method of grouping documents.

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 then describes the architecture and ranking method of the classification-based information retrieval (IR) system that have investigated. Section 4 describes the experimental methodology used to evaluate the system and Section 5 describes the experimental results. Finally, Section 6 provides a summary and discussion.

2 Related Work

This concept of grouping search results has been discussed in Hearst and Pedersen [1], where it was shown that relevant documents tend to be more similar to each other than to non-relevant documents, indicating that relevant documents can be grouped into one category. The two main methods of grouping results are clustering and classification.

Clustering methods typically extracts key phrases from the search results for grouping purposes and attach to each group a candidate cluster label [5, 3]. The search results are treated as a bag of words/phrases, which are ranked according to the statistical features that have been found.

Classification uses predefined category labels that are more meaningful to users than generated labels. Chen and Dumais [2] suggest that category search based on classification can improve search time in comparison to the traditional list-based search, where documents within a category are ranked according to their relative ranking in the original search engine results list. A user study comparing the two interfaces demonstrated the potential superiority of a

[1] http://www.dmoz.org

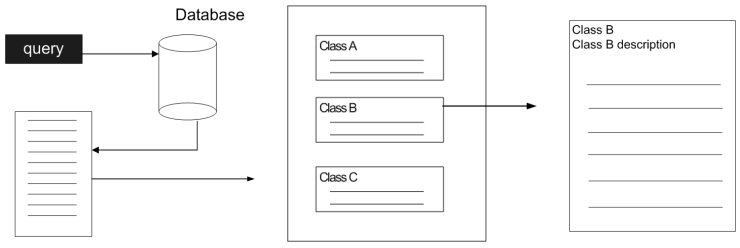
classification-based user interface that can assist the user in quickly focusing in on task-relevant information, this evaluation method is more or less similar to Krishna's work[4].

Previous research in this area has focused on evaluating the grouping of search results versus the traditional list-based method and has not considered a hybrid model; even when a hybrid model is implemented[2], there has not been, to our knowledge, a quantitive analysis of the model as considered here, where users may use either interface to optimise their search performance. Further, the effect of the error rate that can occur when grouping results on users' performance has not previously been given much attention.

3 Classification-Based Information Retrieval

We describe the architecture and ranking method of a classification-based IR system that we have been developing in this section. We assume the existence of a standard retrieval system that, given a query, returns a ranked list of documents as the result set. Web search engines such as Google, Windows Live and Yahoo! satisfy this assumption.

Given a ranked set of documents, it is necessary in our system to classify the documents into their respective classes. Figure 1 provides a conceptual view of the classification-based information retrieval system we are developing. Figure 1a depicts the ranked set of documents provided by a standard IR system. Our system classifies these documents into a number of classes, ranks the classes and then displays a ranked list of classes to the user, as depicted in Figure 1b. When a user clicks on a particular class, the ranked set of documents in this class is then displayed to the user, as shown in Figure 1c.



(a) Standard IR model **(b)** Classification-based IR model **(c)** Results in class B

Fig. 1. Conceptual framework of a classification-based IR system

We now provide a more formal framework for our system. We assume that there are $|D|$ documents in the original result set, D, and we denote the standard IR rank of document, $d_k \in D$ as $s(d_k)$; we refer to this rank as the *scroll rank*

[2] http://demo.carrot2.org/demo-stable/main

(SR). For convenience of presentation we assume that the documents are ordered such that document d_k has scroll rank $s(d_k) = k$. According to the eye tracking experiment[3] and [4], the position in the list can be approximated by the time to find a result.

3.1 Class Rank

After performing classification on the original result set returned by the standard IR system, the documents are grouped into $|C|$ top-level classes. Each class, c_i, consists of a set of documents, $d_{i,j}$, where $1 \leq j \leq |c_i|$, and $|c_i|$ denotes the number of documents in class, c_i.

Each class, c_i, consists of a set of documents, $d_{i,j}$, where $1 \leq j \leq |c_i|$, and $|c_i|$ denotes the number of documents in class, c_i.

Given the set of classes, C, a rank ordering of the classes is necessary. For a document, $d_{i,j}$, let $\psi(i,j) = k$ denote the corresponding index of the same document in the initial result set as output by the standard IR system. Thus, the score associated with document $d_{i,j}$ is $s(d_{\psi(i,j)}) = s(d_k) = k$. Then, the score of class, c_i is given by

$$\phi(c_i) = -min(s(d_{\psi(i,j)})) \quad \text{for } 1 \leq j \leq |c_i|,$$

where $\phi(c_i)$ outputs the score for each class. In our case scores of each class based on the scroll rank of the document within the class. For notational convenience, we assume the classes to be ordered such that class c_i has rank i.

We believe that this simple method of ranking classes is novel and, more importantly, minimizes the affects of the ranking method on the performance of the classification-based system. Conversely, if we had developed a sophisticated ranking system for classes and documents within classes (see Section 3.2), then it becomes increasingly difficult to determine whether differences in performance compared with a standard IR system are due to classification or the new ranking algorithm.

3.2 Document Rank

Having ranked each class, it is now necessary to rank the documents, $d_{i,j}$, within each class, c_i. To do so, we assume the existence of a function, $\varphi(d_{i,j})$, that outputs a score for each document. Here we adopt one of the popular method, the scroll ranks of the documents, $s(d_k)$, as output by the standard IR system, as a score for each document and rank the documents accordingly. Thus, the score for document, $d_{i,j}$, in class, c_i is given by $\varphi(d_{i,j}) = -s(d_{\psi(i,j)})$.

Documents within the class are then ranked according to their scores, the highest score being ranked first, as before. For notational convenience, we assume the documents to be ordered such that document $d_{i,j}$ has rank j in class c_i.

[3] http://www.useit.com/alertbox/reading_pattern.html

3.3 In-Class Rank(ICR)

When a user selects a class, c_i, the *in-class rank* measures the number of class labels and documents that the user examines, when the target document, $d_k = d_{i,j}$ is in class, c_i. The in-class rank is

$$r(d_{i,j}) = i + j, \tag{1}$$

since the user must look at the first i-ranked class descriptions and then the first j-ranked documents within the known class.

However, a classification-based IR system introduces a small overhead. If the target document is ranked high, then this overhead may be noticeable.

3.4 Scrolled-Classification Rank(SCR)

As we shall see shortly, it is often useful to talk about the *scrolled classification rank*, denoted by $s(d_{i,j})$, which we define as the total number of classes and documents a user must examine to find document $d_{i,j}$ by sequentially scrolling through each class and its associated documents in rank order.

In this case, the user will look at i classes, and all of the documents in the previous i-1 classes together with the first j documents of the last class. Thus, the scrolled classification rank of document, $d_{i,j}$ is given by

$$s(d_{i,j}) = i + \sum_{k=1}^{i-1} |c_k| + j. \tag{2}$$

3.5 Out-Class/Scroll-Class Rank(OSCR) and Out-Class/Revert Rank(ORR)

If the target document is not within the selected class, then the user must perform additional work.Upon failing to find the target document in the chosen class, the user may choose to

(i) *scroll* through the classes and the documents in each class in rank order, or
(ii) *revert* to the standard IR display and sequentially scroll through the ranked result set.

The *out-class/scroll-class rank* and *out-class/revert rank* are, respectively, the number of documents the user must then examine in order to find the target in case (i) and (ii) above. We now formalise these notions.

The *out-class/scroll-class rank*, $p(d_{i,j})$, is the total number of class labels and documents that a user must examine in order to find the document for case (i), where the users chooses to scroll through the classes and documents in rank order, after not finding the target in the selected class. Let c_e denote the class the user erroneously selects. Then the out-class/scroll-class rank is given by

$$p(d_{i,j}) = \begin{cases} (e + |c_e|) + s(d_{i,j}) & \text{if } e > i, \\ e + s(d_{i,j}) & \text{if } e < i. \end{cases} \tag{3}$$

The *out-class/revert rank*, $q(d_{i,j})$, is the total number of class labels and documents that a user must examine in order to find the document for case (ii), where the user reverts to the standard result set, i.e. no classification is used in the second phase of the search. The out-class/revert rank is given by

$$q(d_{i,j}) = (e + |c_e|) + s(d_{\psi(i,j)}) = (e + |c_e|) + s(d_k) = (e + |c_e|) + k, \qquad (4)$$

where $\psi(i, j) = k$. Note that the out-class/revert rank is a hybrid search strategy that begins with a classification-based strategy and reverts to a ranked-list strategy if the document is not present in the first class selected. This hybrid strategy is different from the presentation in cluster-based search engines, where the user is presented with the ranked listing in a main window and the clusters in another.

3.6 Classification

We have, until now, ignored how classification is performed. In the experiments of Section 5 we assume two cases.

In the first case we assume we have an oracle based on 16 top level categories of the Open Directory that correctly classifies the documents. Of course, in practice, this is not possible. However, analysis of this case provides us with valuable information regarding the best-case performance of the system. Any other system in which classification errors occur will perform worse. In the second case, we assume classification is performed based on a k-nearest neighbour (KNN) classifier [6]. That is, given a document, d_j, we find its k most similar documents in a database of pre-classified documents.

4 Experimental Methodology

Target Testing. The experimental methodology *simulates* a user performing a known-item search, also referred to as target testing.

In our context we make use of target testing to evaluate the performance of a classification-based retrieval system. The motivation is that target testing allows us to evaluate the system automatically without users, and is a precursor to user testing. Additionally, target testing allows us to evaluate the system on numerous queries at a minimal cost in comparison to user testing. However target testing has some shortcoming in that the queries generated for target testing do not necessarily simulate "real" user queries. Moreover, good performance of the system for target testing does not guarantee similar performance when testing the system with "real" users.

Automatic Query Generation. For a given document repository, in our case extracted from the Open Directory, we randomly select target documents. For each target document, a user query is automatically generated by selecting a

Table 1. Summary of the operating conditions and the number of classes and documents examined in each case

simulated user/target	correctly classified	misclassified
knows class	ICR (C1)	OSCR (C4a) or ORR (C4b)
does not know class	SCR (C2a) or SR (C2b)	SCR (C5a) or SR (C5b)
thinks knows class	OSCR (C3a) or ORR (C3b)	OSCR (C6a) or ORR (C6b)

number of words from the target document. This can be performed in a variety of ways (cf. [7], [8]). However, the exact procedure is not important. We only require that queries can be generated such that the target document appears within a designated range of scroll rank. In this way we can simulate a range of good (high ranking) to poor (low ranking) queries.

User/Machine Models. Table 1 summarises the models and corresponding user strategies we described in section 3. The three user models are (i) the user knows class (case 1 and 4); (ii) the user does not know class (case 2 and 5) and, (iii) the user think he knows (case 3 and 6). Note for each user model, there are two cases associated with it because there are two machine models (correct/incorrect classification of the target document). In the Table 1, we assume that the user employs the search strategies we introduced in Section 3.

5 Experiment

The dataset used in our experiments is derived from the Open Directory Project. We have chosen the following 12 top level classes to construct our testset: Arts, Business, Computers, Games, Health, Kids and Teens, Society, Science, Shopping, Home, Sports and Recreation.

We crawled and downloaded all the documents from these 12 top-level classes during September 2006. After removing the noisy data, we divided the remaining 792,030 documents into training set (500,430 documents) and test set (291,600 documents). The training set was used for classification with the k-nearest neighbour classifier.

We randomly selected 600 target documents from the test set, and for each target document we generated 10 queries. The queries were designed so that the scroll rank of the target document output by the standard IR system fell into intervals counting 5 ranks from 1 to 50.Thus, for each target document, we used a set of 10 queries that ranged from "very good" (scroll rank between 1-5) to "very poor" (scroll rank 36-50). The experimental results were averaged over all 600 target documents.

The underlying IR system is based on the open-source search software, Lucene[4]. For stemming we make use of the open-source stemmer, Snowball [5]. The default document ranking algorithm from Lucene was used.

[4] http://lucene.apache.org
[5] http://snowball.tartarus.org

5.1 Experimental Results

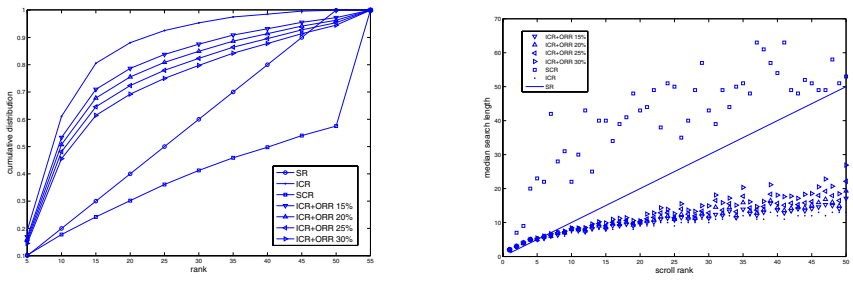
We performed three sets of experiments. In the first set we used the domz directory as an oracle to classify the result set, the second set used a k-nearest neighbour classifier trained on a subset of the Open Directory to classify the result set, and the last set used our classifier in a more realistic scenario.

Classification Based on an Oracle. Each of the 600 documents has been manually classified into one of 12 classes. Thus, dmoz provides us with an oracle with which to classify all documents in the original result set. This allows us to first examine the best-case performance of our classification-based IR system, i.e. when there are no machine classification errors and the simulated user knows the correct class (case C1 in Table 1).

We can also introduce and control error rates for both the user and the machine classifier. Note that, from Table 1, user errors and machine classification errors both result in the same search length (cases C3, C4, and C6). Moreover, the two cases where the user is aware that they do not know the class (cases C2 and C5) are unaffected by the machine misclassification. Thus, when we report error rates, we do not distinguish between human and machine error rates. Rather, the error rate represents the combination of the two.

Figure 2a summarises the results for these cases. It shows the *cumulative* probability of finding the target document as a function of the rank of the target document. We note that for the standard IR system, the rank corresponds to the scroll rank (SR). For the error-free case, the rank corresponds to the in-class rank (ICR). For non-zero error rates, the rank corresponds to the ranks summarised in Table 1.

For the standard IR system, the scroll rank (SR) is a straight line, since the scroll rank is evenly distributed within the 10 intervals described above. We see that for the error-free case (ICR), the classification-based IR system performs significantly better than the standard IR system (SR). In particular, we observe that approximately 60% of all target documents can be found with a rank of 10 or less. That is, for an ideal user and no machine misclassification (case C1), the



(a) cumulative distribution **(b)** Median search length of query results

Fig. 2. Results using dmoz oracle classifier

user must look at no more than 10 classes and documents in order to locate the target document.

The oracle also allows us to control the misclassification rate. And the rates we describe can best be thought of as the combined user and machine error rates. We introduced an error rate of $x\%$ as follows: for $x\%$ of the 600 queries, the user randomly selects a class that does *not* contain the target document, and then uses the out-class/revert (ORR) ranking strategy to locate the document. For the remaining $(100-x)\%$ of queries, the user chooses the correct class and the target document is found using the in-class ranking (ICR) strategy. Thus, the curves for non-zero error rates represent a combination of two strategies, ICR and ORR.

For an overall error rate of 15%, we observe a decline in performance, as expected. However, at this error rate, the classification-based IR system still performs significantly better than the standard system. For example, over 50% of all target documents are found with a rank of 10 or less. As the overall error rate increases, the performance degrades. However, this degradation is rather smooth and even with an error rate of 30%, the performance remains significantly better than that of the standard IR system.

Finally, for completeness, Figure 2a also shows the cumulative distribution for the scrolled classification rank (SCR). This curve is significantly worse than the scroll rank of the standard IR system. Figure 2a shows that in cases C2 and C5, when the user does not know the class, he is best advised to abandon the classification-based IR system and immediately return to the standard system, i.e. follow the second strategy of scroll-rank (SR) in Table 1. Moreover, in cases C3, C4 and C6, where we have either a user or machine error, then if the user does not find the target in the class he knows or thinks is the correct class, the advice is the same, i.e., revert to the standard system following the second strategy of out-class/revert (ORR) in Table 1. That is, a hybrid-based search strategy performs better than either a category-based or ranked-listing alone.

It is important to recognize that the cumulative distribution does not present the full story. Figure 2b plots the median search length as a function of the scroll rank. Note that here we use the median search length distribution since the search length is highly biased by a small number of outliers, while the median is more robust to this bias. It is clear that for target documents with a low scroll rank (less than 5), the median search length using a classification-based system is slightly longer, on average, due to the overhead of inspecting the class description or when an error occurs. Thus, for very good queries, a classification based system actually increases the search length slightly. Conversely, for poorer queries,the median rank of the target document in the classification-based system is always shorter, on average. Interestingly, both the standard IR system and the classification-based system have regions of superior performance. Only when the initial query is poorer, i.e. the scroll rank is below a certain threshold, does the classification-based system offer superior performance.

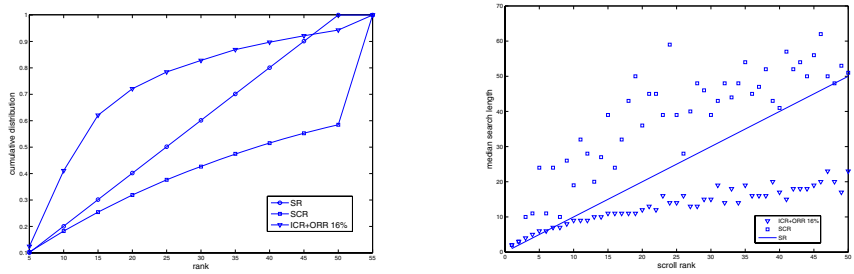
Figure 2b also shows, as expected, that this threshold increases as the misclassification rate increases and it is more evident for poor queries. Thus, for

example, for a misclassification rate of 25%, the scroll rank must be greater than 7 before a classification-based system is superior.

It is also worth noting that in Figure 2b the quality of the initial queries is uniformly distributed, by design. Thus, 20% of queries have an initial scroll rank between 1-5, another 20% between 6-10, and so on. In practice, the distribution of queries is a function of (i) the user, (ii) the distribution of documents in the database, and (iii) the document scoring function [8]. Thus the benefits of a classification-based system will depend strongly on the distribution of the queries. It is interesting to note that a number of studies such as [9, 10] have reported poor correlations between user judgments of document rankings and those produced by search engines, suggesting that classification-based systems may be useful in practice.

***K*-Nearest Neighbour Classification.** The experiments of the previous section show that very good performance can be expected from a classification-based system, especially for poorer queries. The definition of "poorer queries", i.e. queries for which the scroll rank is larger than a given threshold, varies with the misclassification rate. Simulated misclassification rates of 15-30% suggest that (i) performance degrades gracefully as the error rate increases, and (ii) that useful performance improvements can still be obtained with relatively large error rates.

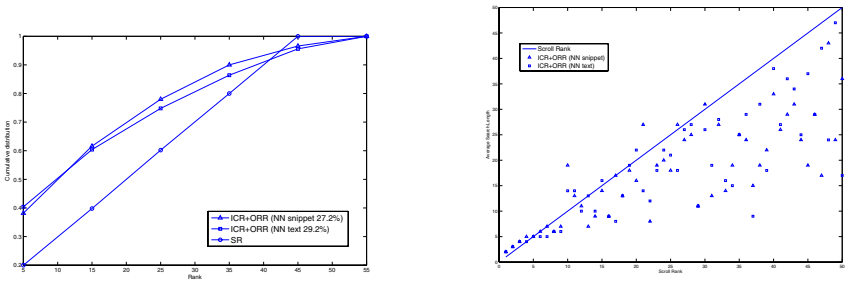
To investigate what misclassification rate we could expect from a classifier, we implemented a simple non-disjoint k-nearest neighbour (KNN) classifier. The repository of documents remains the same, permitting us to measure the misclassification rate at 16%. Figures 3a and 3b show the cumulative distribution and median search length, respectively, in this case. Clearly, even at this error rate, significant improvements can be obtained, depending on the query distribution.



(a) cumulative distribution **(b)** Median search length of query results

Fig. 3. Results for the KNN classifier with non-disjoint classes

***K*-Nearest Neighbour Classification in a More Realistic Scenario.** To investigate what misclassification rate expected from a classifier in a realistic scenario, we implemented a k-nearest neighbor classifier over a real search engine. Due to the absence of an oracle for retrieved results, we adopt the classifier's



(a) cumulative distribution **(b)** Median search length of query results

Fig. 4. Results for the KNN classifier in real case

accuracy on the target document as a measure of the classifier's performance. It may not fully reflect the performance of our classifier, but we can use this measure to approximate our system's performance.

Compared to the previous results shown in Figure 3a, Figure 4a shows the misclassification rate of the k-nearest neighbor is about 30%, which is worse than the previous results. However, we still can see that these results are consistent with the previous conclusions. Moreover, for this more realistic case the classifier trained from dmoz snippets reduces the misclassification rate to about 28%, which is slight better than that trained on the full text of web pages.

Figure 4b shows that for poor queries, the combined class rank will achieve a better performance than scroll rank. However the trend in the curve is not as clear out as the previous curve in Figure 3b. It can also be seen that the curve has high variance. However, our general conclusion that the hybrid-based search strategy performs better than a category-based or ranked-list alone, is still valid.

6 Concluding Remarks

In this paper we have examined how a hybrid model of an IR system might benefit a user. Our study was based on several novel ideas/assumptions.

In order to investigate the best-case performance, we constructed a system using a subset of the Open Directory. All documents in this subset have been manually classified and therefore provide "ground truth" for comparison. The advantage of our approach to rank the class is two-fold. It is simple, however, more importantly, this ranking closely approximates the scroll rank, thus allowing comparison between the class rank and scroll rank. In addition, we identified three classes of simulated users and rational user search strategies. By basing our evaluation on known-item search, we are able to simulate a very large number of user searches and therefore provide statistically significant experimental results. We acknowledge that real users may perform differently and future work is needed to determine the correlation between our simulations, which provide

an empirical upper bound on performance for real users, and the behavior of real users.

Our experimental results not only demonstrate the advantage when the user correctly identifies the class and there is no machine error, but also suggest the strategy the user should take to achieve the optimal performance when the user does not know the class or when there are user and machine errors.

Using the Open Directory, we were also able to control the error rates of both the user and the machine classification. Simulation results showed that the performance degrades gracefully as the error rate increases, and that even for error rates as high as 30%, significant reductions in search time can still be achieved. However, these reductions only occur when the query results in the target document having an initial scroll rank above a minimum rank, and this minimum rank increases with the error rate.

This may imply that a classification-based system may be more beneficial for informational queries [11], where the user will probably inspect several search results, rather than for navigational queries [11], which are similar to known-item queries that target a single web page. Such a system could also be useful for novice users who are more likely to generate poor queries.

References

[1] Hearst, M.A., Pedersen, J.O.: Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 76–84.

[2] Chen, H., Dumais, S.: Bring order to the web: Automatically categorizing search results. In: CHI 2000: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 145–152. ACM Press, New York (2000)

[3] Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.W.: Learning to cluster web search results. In: SIGIR 2004: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 210–217. ACM Press, New York (2004)

[4] Kummamuru, K., Lotlikar, R., Roy, S., Singal, K., Krishnapuram, R.: A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In: Proceedings of the 13th International Conference on World Wide Web, pp. 658–665 (2004)

[5] Osinski, S., Weiss, D.: Carrot 2: Design of a flexible and efficient web information retrieval framework. In: Proceedings of the third International Atlantic Web Intelligence Conference, Berlin. LNCS, pp. 439–444. Springer, Heidelberg (2005)

[6] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. 2nd edn. Wiley-Interscience, New York (2000)

[7] Azzopardi, L., Rijke, M.D.: Automatic construction of known-item finding test beds. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 603–604. ACM Press, New York (2006)

[8] Vinay, V., Cox, I.J., Milic-Frayling, N., Wood, K.: Evaluating relevance feedback algorithms for searching on small displays. In: 27th European Conference on IR Research. ECIR (2005)

[9] Bar-Ilan, J., Keenoy, K., Yaari, E., Levene, M.: User rankings of search engine results. J. American Society for Information Science and Technology 58(9), 1254–1266 (2007)
[10] Su, L.T.: A comprehensive and systematic model of user evaluation of web search engines: Ii. an evaluation by undergraduates. J. American Society for Information Science and Technology 54(13), 1193–1223 (2003)
[11] Broder, A.: A taxonomy of web search. SIGIR Forum 36(2), 3–10 (2002)