On Ranking Techniques for Desktop Search [*]

Sara Cohen
Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology
sarac@ie.technion.ac.il

Carmel Domshlak
Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology
dcarmel@ie.technion.ac.il

Naama Zwerdling
Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology
anaama@tx.technion.ac.il

ABSTRACT

This paper addresses the *desktop search* problem by considering various techniques for ranking results of a search query over the file system. First, basic ranking techniques, which are based on a single file feature (e.g., file name, file content, access date, etc.) are considered. Next, two learning-based ranking schemes are presented, and are shown to be significantly more effective than the basic ranking methods. Finally, a novel ranking technique, based on query selectiveness is considered, for use during the cold-start period of the system. This method is also shown to be empirically effective, even though it does not involve any learning.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms: Experimentation, Human Factors

Keywords: desktop search, personal information management, ranking

1. INTRODUCTION

Due to the increasing storage capabilities of standard personal computers, users are no longer motivated to delete old files. The practice of "never deleting files" has distinct advantages, since it ensures that important data will never be accidentally removed. However, as an unfortunate side effect, the personal computer often becomes an unwieldy mess. Thus, locating a specific file, within the file system, may become a challenging (and even daunting) task. Therefore, numerous research and industrial projects have evolved in the area of personal information management (PIM) [5]. These projects aim to develop technologies allowing users to store, access, and effectively search for information in their personal computer, e.g., [3, 2, 1].

The focus of this paper is on *desktop search*, i.e., effective search within a personal computer. There are currently only limited (published) insights into the question of how to rank desktop-search results, mostly based on very simple techniques. In this paper we focus on the problem finding an effective ranking scheme for desktop search. To this end, we implemented a simple desktop-search tool, which returns unranked results to the users and logs the user interaction. We develop various ranking strategies and analyze their effectiveness, "post-mortem," based on the logs.

2. RANKING TECHNIQUES

A query q is simply a sequence of words. A file f is a *candidate answer* for q if there is at least one word appearing in q that also appears in the content, filename, or path of f. We discuss various techniques for ranking a candidate answer f.

Ranking by Basic File Features.

We considered both textual-based and date-based features. For the textual-based features we consider CONTENT, PATH, NAME, whose values correlate with the cosine distance between the tf.idf vectors of the query and the content, path, name of f, respectively. For the date-based features we allow ACCESSDATE, UPDATEDATE, and CREATEDATE whose values decrease as the distance in time grows between the date of querying and the access, update, and create date of f, respectively.

Learning to Rank.

As we already mentioned, one of the components of our desktop search system is the logger that stores all the data on the past search sessions of the user. In principle, this information can be used for *learning* a better ranking method. We considered two such ranking methods.

We use a Support Vector Machine (SVM) to learn a linear function of the basic file features that minimizes the classification error on our training logger data. (Similarly to, e.g., [4], we actually learned a binary relation over the candidate answers.) Thus, the feature SVM-75 is defined for each user, by using as training data four randomly chosen subsets of 75% of its log data, with the remaining 25% used as the test set. The results were averaged over these four samples and over all users. Similarly, the feature SVM-90, which has a 90/10 data partition setup, was evaluated.

Learning a ranking function using a SVM can be done in time linear in the number of file features, and polynomial in the number of candidate answers of all queries seen thus far. As the amount of the logged search sessions grows, the latter complexity factor significantly slows down the learner. Although learning can be performed offline, at some point the system will unavoidably have to cut down the amount of available training data, possibly using only some chronological suffix of the logged data. Due to this limitation, we also considered the simple learning scheme, called LEXORD (for "lexicographic order"), which corresponds to a lexico-

[*] Sara Cohen and Naama Zwerdling were partially supported by the ISF (Grant 1032/05). Carmel Domshlak was partially supported by the BSF (Grant 2004216).

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

Feature	$\mathsf{Score}(\tau, \mathcal{S}_{\text{all}})$	Feature	$\mathsf{Score}(\tau, \mathcal{S}_{2\text{-}50})$
SVM-90	7.84	SVM-90	4.72
SVM-75	8.15	Selective	4.86
LexOrd	9.19	SVM-75	4.93
AccessDate	9.85	LexOrd	5.15
UpdateDate	10.18	Name	5.66
Selective	10.89	Path	6.32
CreateDate	11.51	UpdateDate	6.43
Name	11.53	AccessDate	6.54
Path	12.60	Content	6.75
Random	*14.45*	CreateDate	6.80
Content	15.43	*Random*	*6.99*

Feature	$\mathsf{TopScore}_k(\tau, \mathcal{S}_{\text{all}})$		$\mathsf{TopScore}_k(\tau, \mathcal{S}_{2\text{-}50})$	
	$k = 1$	$k = 10$	$k = 1$	$k = 10$
SVM-90	34.4	64.9	36.6	71.7
SVM-75	32.8	63.5	34.5	70.5
LexOrd	34.0	62.5	37.2	68.5
Selective	27.1	63.1	29.3	72.7
UpdateDate	21.6	52.5	23.4	57.8
AccessDate	19.2	52.3	19.2	52.3
Name	16.5	51.7	18.1	60.5
CreateDate	17.7	48.5	19.2	54.1
Content	13.2	45.0	14.6	55.1
Path	6.9	46.0	7.2	54.1
Random	*0.0*	*36.3*	*0.0*	*46.7*

Figure 1: Expected placement of features (left table), and effectiveness at $k \in \{1, 10\}$ (right table).

graphic aggregation of single-feature-based rankings based on their relative efficiency when used in isolation.

Selectiveness of Features. Learning-based ranking is relevant only in presence of sufficient training data. In the context of desktop search, this can be a real obstacle, since users tend to issue only a few queries a day. We suggest a the simple ranking mechanism Selective that combines the textual-based features, based on their relative selectiveness, that can be useful for the cold-start period. Intuitively, the ranking mechanism Selective, combines (i) the information carried by the textual properties of the candidate answers, and (ii) the frequency of textual connection between each such property and query, within the set of candidate answers. Thus, Selective follows the principle underlying the standard idf (inverse document frequency) measure used in IR—the contributions of different textual features to Selective on a given candidate answer are combined via a weighted sum in which less "common" features (in the set of candidate answers) are given larger weights.

3. EXPERIMENTAL EVALUATION

The desktop search engine was distributed to 19 volunteers. In total 1219 queries were issued, where 188 queries had a single result (i.e., candidate answer), 916 queries has 2–50 results and 115 queries had over 50 results.

We analyzed the effectiveness of our ranking mechanisms on the set of all queries with more than one result, with 2–50 results and with over 50 results, denoted $\mathcal{S}_{\text{all}}$, $\mathcal{S}_{2\text{-}50}$ and $\mathcal{S}_{>50}$, respectively. The results for $\mathcal{S}_{>50}$ are similar to those of $\mathcal{S}_{\text{all}}$ and are omitted due to space limitations.

Two measures were used to evaluate the effectiveness of our ranking mechanisms: (1) $\mathsf{Score}(\tau, \mathcal{S})$ ($\mathcal{S} \in \{\mathcal{S}_{\text{all}}, \mathcal{S}_{2\text{-}50}\}$) is the *expected placement* of the user's target file,[1] for queries in $\mathcal{S}$, according to the ranking mechanism τ. (2) $\mathsf{TopScore}_k(\tau, \mathcal{S})$, called the *effectiveness of τ at k*, is the percentage of queries in $\mathcal{S}$ in which τ ranks the target file within the top k files. (We only consider queries that have at least k results in this measure.) Thus, a good ranking mechanism will have a low value for $\mathsf{Score}(\tau, \mathcal{S})$ and a high value for $\mathsf{TopScore}_k(\tau, \mathcal{S})$.

Our analysis of the ranking mechanisms considered is summarized in Figure 1. When considering only the basic file features, the date-based features have significantly better expected placement than the textual-based features for $\mathcal{S}_{\text{all}}$, while the opposite is true for $\mathcal{S}_{2\text{-}50}$.[2] We believe that this can be explained since textual-based features are most useful when they are selective (i.e., when there are few results).

Our learning-based ranking features (SVM-75, SVM-90 and LexOrd) significantly outperformed the basic ranking features. For $\mathsf{Score}(\tau, \cdot)$, we observe that: (1) SVM-75 and SVM-90 ended up clear winners and (2) the difference between LexOrd and the next most effective method was statistically significant. For all $\mathsf{TopScore}_k(\tau, \cdot)$, $k \in \{1, 10\}$, it can be seen that SVM-90, SVM-75, and LexOrd are more effective than all basic ranking methods.

Finally, we consider effectiveness of Selective. For the measure $\mathsf{Score}(\text{Selective}, \cdot)$, we observe that Selective is better than its most effective component (i.e., the textual-based features), and this difference is statistically significant. Somewhat surprisingly, on $\mathcal{S}_{2\text{-}50}$ Selective even successfully competes with our learning-based ranking methods—it appears to be more effective than LexOrd (with statistical significance) and is, in fact, statistically indistinguishable with SVM-75 and SVM-90. Thus, Selective appears highly effective for $\mathcal{S}_{2\text{-}50}$, even though it does not involve any learning. Observe also that Selective is always among the four best features for $\mathsf{TopScore}_k(\tau, \cdot)$.

To summarize, we have shown that our learning-based ranking methods are clearly much more effective than the basic file features, and that Selective is a good choice for $\mathcal{S}_{2\text{-}50}$, during the cold-start period.

4. REFERENCES

[1] S. Dumais, E. Cutrell, JJ Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: a system for personal information retrieval and re-use. In *SIGIR'03*.

[2] R. Fagin, R. Kumar, K. S. McCurley, J. Novak, D. Sivakumar, J. A. Tomlin, and D. P. Williamson. Searching the workplace web. In *WWW'03*.

[3] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. MyLifeBits: Fulfilling the Memex vision. In *ACM Multimedia'02*.

[4] T. Joachims. Optimizing search engines using clickthrough data. In *KDD'02*.

[5] J. Teevan, W. Jones, and B. B. Bederson, editors. *Special Issue on Personal Information Management*, volume 49 of *Communication of the ACM*, 2006.

[1] The target file is the file chosen by the user. We assume that this file is unique and is recognized by the user.

[2] Statistically significant differences in performance are determined using the two-sided Wilcoxon test at the 95% confidence level.