Predicting Task-Specific Webpages for Revisiting

Arwen Twinkle Lettkeman, Simone Stumpf, Jed Irvine, Jonathan Herlocker

1148 Kelley Engineering Center
School of EECS, Oregon State University
Corvallis, OR 97331, U.S.A
{lettkema, stumpf, irvine, herlock }@eecs.oregonstate.edu

Abstract

With the increased use of the web has come a corresponding increase in information overload that users face when trying to locate specific webpages, especially as a majority of visits to webpages are revisits. While automatically created browsing history lists offer a potential low-cost solution to re-locating webpages, even short browsing sessions generate a glut of webpages that do not relate to the user's information need or have no revisit value. We address how we can better support web users who want to return to information on a webpage that they have previously visited by building more useful history lists. The paper reports on a combination technique that semi-automatically segments the webpage browsing history list into tasks, applies heuristics to remove webpages that carry no intrinsic revisit value, and uses a learning model, sensitive to individual users and tasks, that predicts which webpages are likely to be revisited again. We present results from an empirical evaluation that report the likely revisit need of users and that show that adequate overall prediction accuracy can be achieved. This approach can be used to increase utility of history lists by removing information overload to users when revisiting webpages.

Introduction

Initially just a source for scientific information, the World Wide Web is rapidly becoming the preferred destination for both work-related and personal information. With this increased use has come a corresponding increase in information overload that users face when trying to locate specific information – webpages – on the web.

In this paper, we address the research question of how we can better support web users who want to return to information on a webpage that they have previously visited, an common activity for web users. In particular, we want to design usable solutions that reduce the overhead cost that those users experience each time they want to return to information. The overhead cost includes both the cognitive and physical activity that the user must engage in to relocate and revisit a page. These costs are present for over half of all webpage visits (Tauscher and Greenberg 1997). We consider activities to be overhead cost if they could be removed without affecting the user's performance of their work tasks. Examples of overhead costs in our situation include time spent remembering where a webpage was located, iteratively generating search queries, or browsing hyperlinks.

There are four main access paths that users can follow to re-locate a page and that narrow down their entry points for their revisit.

A1. Type a URL into the address bar of a web browser
A2. Use a search engine to re-locate a page in a list of search results
A3. Follow a hyperlink embedded in a manually-created bookmark, including a URL emailed to self or a link added to a personal webpage.
A4. Follow a hyperlink embedded in the web browser visit history

(If their entry point does not contain the information they seek, further costs may be encountered in conducting localized browsing to find the information or page they want.)

Entering URLs directly (A1) does not have low cost unless the user can quickly recall the full URL, e.g. home pages for brand names, or glean the URL from a memory aid, e.g. a printout, hand-written note or auto-complete features. Most webpages have URLs too complex to be recalled from memory. Search engines (A2) can provide low cost means to re-locating webpages, but as the quantity and variety of information on the web increases, users face increasingly higher costs to generate search queries that uniquely identify the desired webpages. Bookmarks (A3) and history lists (A4) of previous browsing sessions can be very low cost access paths, but studies have shown that these mechanisms are not widely used (Jones et al. 2001).

It is easy to understand why manually created bookmarks or automatically created history lists are not widely used. Users perceive additional effort and time cost to place bookmarks, yet there is a high risk that the time invested will not pay off in the future, as many pages are not revisited. While web browser history lists are created automatically and involve no additional work or risk to create, the cost of accessing pages comes from users having to search or navigate through the large volume of webpage visits that appear within those histories, even for relatively short browsing sessions. Many of the records in the history lists will point to pages not be related to the users' current information need, and many records will point to

pages that have no revisit value, regardless of the information need.

We believe that, through the integration with intelligent systems, we can create history lists automatically that substantially reduces the costs for users wishing to re-locate and revisit information or a page that they have visited before on the web. We apply a combination of two approaches. First, we semi-automatically segment the webpage browsing history into groups of pages associated with a specific task. In this way, the user can limit the web history to pages that are relevant to the active task. Then we apply a combination of heuristics and machine learning to predict, for each task, which resources are likely to be of value if revisited. This allows us to remove or hide non-relevant history records from the view of the user, further increasing the usability of the webpage history.

Related Work

There have been several studies of navigation behavior that have included examination of web revisits. One study (Tauscher and Greenberg 1997) found that 58% of an individual's visits are revisits. They demonstrated a new history list UI that is more effective than the "stack-based" history view in commercial web browsers. The approach described in our paper could be integrated into a UI that follows the design guidelines of Tauscher and Greenberg. While usage patterns of the web may have changed dramatically since 1994-1997, we still believe that many of the individual visits are revisits.

Previous research on task-centric environments has examined user interfaces for segmenting information and interaction events by task. Previous user interface approaches have allowed users to manually group and organizing resources by task (Card and Henderson 1987, Robertson et al. 2000, Bellotti et al 2003). Some task-centric systems have also implemented extensive event collection frameworks which, in addition to the resources used in a task, also record user's actions performed on those resources (Kaptelinin 2003, Fenstermacher and Ginsburg 2002, Canny 2004). Our TaskTracer approach combines both user interfaces and extensive data collection and, in addition, uses learning components to semi-automatically segment resources and interaction events into tasks.

Predicting which documents a user would want to visit is of value in many informational retrieval areas, from predicting product pages that the user is likely to purchase to predicting which hyperlinked pages should be preloaded. Most of this work is focused on predicting new pages that the user is likely to visit, a problem which we believe to have different properties than predicting revisits. Even the past work on using implicit measures to predict document value, (e.g., Badi et al. 2006, Claypool et al. 2001, Kelly et al. 2004) needs to be re-evaluated in a context that differentiates between documents that are valuable once and documents that will be revisited.

Our technique presented in this paper extends previous work to predict revisit value. Additionally, we develop not only a learning model for each user, but a model that is sensitive to their active task.

Semi-automatically Segmenting Web Histories

To semi-automatically segment webpage history into groups associated with tasks, we make use of the TaskTracer system (Dragunov et al. 2005). The TaskTracer system is based on two main premises: (a) the behavior of the user at the desktop is an interleaved timeline of different tasks and (b) a user's current task is highly predictive of the resources (such as webpages, files, etc.) that will be used on that task. Thus, TaskTracer assumes that tasks provide a useful abstraction for organizing and accessing resources. In the initial, basic operation of TaskTracer, the user defines a hierarchy of longer-lived tasks, such as "teach CS534", "prepare NSF proposal B", etc. Once the tasks have been defined, the user can then easily indicate to TaskTracer which task they are currently working on – their active task. TaskTracer further supports the user by predicting the active task from observed user activity, enabling prediction of task switches (Shen et al. 2006).

TaskTracer records all resources that are accessed and associates them with the active task. A URL is considered a resource and is associated with the task that was active. Through this process, the webpage history records are semi-automatically segmented by task. For example, if the task is "NSF Proposal", the list of webpages associated with that task will likely include the NSF proposal submission web site, the university's internal research office site, etc. Webpages for each task can be easily viewed and revisited through the TaskExplorer user interface, which presents previously accessed resources by task.

TaskTracer reduces the cost of re-locating information on previously visited webpages. At a minimum, the number of history records that the user has to search or browse through within a task is significantly reduced. Even more powerful is the fact that all of the history records presented to the user are relevant to the user's active task.

In spite of this improvement, users still face costs in re-locating information on previously visited webpages. Since all browsing activity on a task is recorded, longer running tasks soon accumulate a glut of webpages. In order to reduce this cost, we have designed a technique to further reduce the set of web history records that are exposed to the user as part of the web history of a task. This technique combines heuristics and a machine learning approach that predicts which URLs represent pages that the user is likely

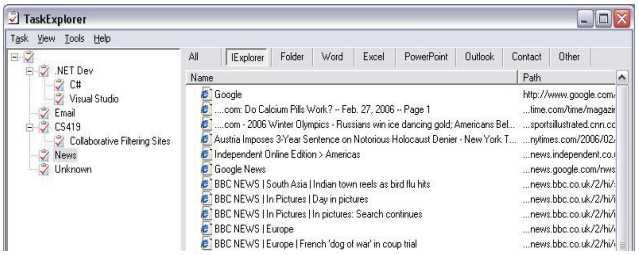

Figure 1: TaskExplorer with webpage resources

to want to visit again. In the remainder of this paper, we present in detail this technique and evaluate it on empirical user data. Finally, we provide a discussion of these results.

Predicting Webpages for Revisiting

Heuristics

We use three heuristics to filter initial noise from the web browsing history. These heuristics were developed through analysis of user web browsing data. The analysis revealed many easily identifiable webpages of no intrinsic value for the user, regardless of task.

Our heuristics take little computational time to filter out undesirable webpages from the history. As event messages are received from the web browser, heuristics are applied. Each heuristic decides whether or not a page fits specific criteria, discussed in detail below. If a page fails any one of the criteria the webpage is filtered out.

First we remove *error pages*. These include error, blank or redirect pages. To identify these, we examine the text of a webpage for the *presence* of *error keywords* (error, 404, unavailable, not available, 403, not found, redirect, autodirect, autoredirect, refresh) or for the *absence* of *any words* (i.e. blank pages).

Second, we remove obvious *duplicate pages*. We restrict each page to one appearance in the history list, regardless of whether the content has changed. We consider pages with the same URL to be duplicate pages, even if their titles are different.

Third, we remove *transitory pages* from the data. We assume that pages visited less than five seconds have no intrinsic value to the user. We assume either that the user scanned the webpage quickly and decided it was not useful before navigating away or that the page is only used to navigate between pages. We qualitatively picked the thresholds after examining logs of web usage history.

The heuristics remove the pages that we feel are rarely relevant to any task. We then apply machine learning to further predict what webpages a user might want to revisit on a task.

Feature Construction

We extract the following information from each webpage visit:

- time the page is open;
- percentage of links on the page that have been followed;
- number of copy commands executed on that page;
- words in the URL , title and content of the page.

For the word features, we use the term frequency.

We do not weight these features equally. Interviews with users indicated that frequently the webpages they found important to a task had words in common with the task name or task description. Thus, we give words within the page that are also within the task description or task name twice as much weight as other words. We also weight words in the title and URL of a webpage more than words appearing in the content, since words appearing in the title and URL are usually designed to be indicative of the webpage content.

Feature Selection

Feature selection can improve the accuracy and speed of classifiers. Our current implementation allows up to 10,000 words to be collected as features. We employed three feature selection methods to reduce this to the most useful features.

First, as the webpage text content was gathered each word was compared against a *stopword* list to eliminate words common to webpages such as "http" as well as words common to the English language such as "the" and "an". The stopword list used contained over 370 words.

Second, we applied the Porter stemming algorithm to reduce the words to their roots (Porter 1980). This ensures that words that are the same in root form are counted as the same word further reducing the number of features.

Finally, we employed *information gain* to select features with the highest predictive impact. We used information gain in conjunction to restrict the number of features to 100 (K=100).

Classification Methods

We applied two classification methods – one based on Naïve Bayes (NB), and one using linear support vector machines (SVM) both implemented in the Weka v3.4 software package (Witten & Frank 2005). We used the Naïve Bayes classifier that is an implementation of the algorithm described in (John & Langley 1995) and the Sequential Minimal Optimization (SMO) classifier, which is an implementation of the SVM described in (Platt 1998, Keerthi et al., 2001). For the SMO algorithm, we used a polynomial kernel and C=1.

Experimental Evaluation

To evaluate our methods for predicting the task-specific revisit relevance of pages, we ran an experiment with human subjects. This study provided us with statistics describing what percentage of web history pages users desire to keep in their web history, and allowed us to collect the implicit measures that we needed for training, plus explicit evaluations of revisit data by the users that we could use for test set evaluation.

Data Collection

We collected data from eight students and faculty members who had been using the TaskTracer system regularly. For the experiment, the participants used a version of TaskTracer that implemented the feature collection from webpages in the background, but which otherwise did not differ from the current TaskTracer system. Each participant was asked to choose three to five tasks which required some amount of internet research. In the first step of the

data collection, they generated webpage visits as they worked on their tasks. In the second step, after they had completed their tasks, they were asked to delete any pages from the history list that they considered not useful (or unimportant) for future use, in the case they were to return to that task later on. If a user did not remove a page from the history list during the second step, then we assumed that the page in question was one with some revisit value.

A summary description of the data collected is given in Table 1. We evaluated the data from each task separately – thus we had one dataset for each task. All tasks that contained less than ten webpages are not included in our evaluation. Ten pages or less in the web history do not present a significant information overload, and furthermore, are likely to provide insufficient training data for the classification algorithm. We also did not include data from tasks that were comprised primarily of webpages using non-English character sets. In the end, we evaluated our technique on 21 datasets.

Evaluation Methodology

We evaluated each prediction model (Naïve Bayes and SVM) on each dataset, with and without feature selection (K=100 features). In both cases, we used ten-fold cross validation. For each dataset, we computed the percentage of correct predictions.

Results – What Users Want

In this experiment after performing a set of tasks that required some amount of web browsing, users went back and cleaned out their web browsing history list for each task to remove not useful or unimportant pages fore revisits. We assumed that if a user chose not to delete a webpage, they had made an explicit decision that they might want to return to that page some day. Thus we are measuring perceived future revisit value of web history records from the user's viewpoint.

Participants	Task	total # of pages	# of pages deleted	% deleted
P1	1	33	7	21.21
	2	74	15	20.27
	3	36	17	47.22
P2	4	16	11	68.75
P3	5	34	23	67.65
	6	38	24	63.16
	7	32	23	71.88
P4	8	42	7	16.67
	9	30	12	40.00
P5	10	50	21	42.00
	11	57	26	45.61
	12	47	20	42.55
	13	43	10	23.26
P6	14	25	9	36.00
	15	16	3	18.75
	16	35	3	8.57
	17	63	6	9.52
	18	30	2	6.67
P7	19	69	5	7.25
P8	20	48	29	60.42
	21	29	17	58.62

Table 1. Summary of datasets.

Our experiment does not measure the effects of the heuristics, because webpages filtered out by the heuristics were not shown to the when they edited their web history. Also, all the accuracy numbers represent accuracy after the most obviously non-useful webpages have been removed. We expect that the accuracy numbers would increase if were able to include the level of pre-heuristic filtering.

A summary of users' history cleaning behavior is shown in Table 1. On average, users deleted 34.3% of the webpage resources in their history lists. However, deletion behavior was highly user- and task-specific. For example, deletion rate per user ranged from a high rate average of 68.75% (participant P2), to a low rate average of 7.25% (participant P7). Furthermore, there was variation of deletion behavior for users according to task. For example, participant P6 had a deletion rate of 36% on a certain task, whereas on a different task this dropped to a low rate of 6.67%.

Results – Classifier Accuracy

The boxplots in Figure 2 show the accuracy achieved when each classification method is used, both with and without feature selection. The average prediction accuracy for Naïve Bayes was 68.4%. The application of SVM (third boxplot) improved this result, raising average accuracy levels to 74.7%, not considering feature selection. That represents a relative improvement of 9%.

The addition of feature selection did increase the overall accuracy. By using the top 100 features as determined by information gain, accuracy was improved to 71.05% (NaiveBayes) and 78.02% (SVM), respectively. These represent relative gains of 4.5% and 3%. Furthermore the reduced feature spaces lead to reductions in space and online computation time.

Accuracy varied considerably across users. As examples accuracy for data collected from participant P1 on average achieved 84.39%, whereas accuracy for participant P5's

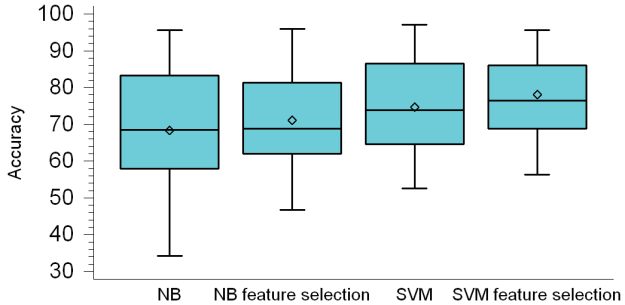

Figure 2. Average accuracy by classification method and use of automated feature selection.

data was 70.92% (SVM using feature selection).

There was also significant variability in accuracy of predictions across tasks. Figure 3 shows accuracy broken down by task and grouped by user. As an example, participant P4 completed two tasks. On one of the tasks, accuracy of 83.3% was achieved, whereas on the other task only 66.7% accuracy was achieved (SVM using feature selection)

Moreover, as we can see from Figure 3, the application of prediction algorithms and feature selection was not consistent in performance in all kinds of situations. For some tasks for some users, Naïve Bayes was as good as (or better) than all other approaches.

There was a large imbalance between pages kept and pages deleted from the history. In spite of that, we found that the prediction errors were largely equally balanced between false positives and false negatives, with a slight leaning towards falsely predicting that a user would want to remove a page from their history, when they actually did not remove that page. This can be seen in the confusion matrices shown in Table 3.

Naïve Bayes (NB)

Predicted	
Keep	Delete
49%	16%
13%	21%

NB w/Feature Selection

Predicted		
Keep	Delete	
50%	15%	Kept
12%	22%	Deleted

SVM

Predicted	
Keep	Delete
53%	12%
12%	23%

SVM w/Feature Selection

Predicted		
Keep	Delete	
56%	10%	Kept
11%	23%	Deleted

Table 1.Confusion matrices illustrating relatively similar numbers of false positives and false negatives.

Discussion

This study shows that we can get on average 78% accuracy in predicting what pages the user wants to keep in their history list. We expect that this is an underestimate of the true accuracy that is possible to achieve, for several reasons. First, we measured accuracy after we applied the heuristics to remove the records that are obviously not useful. Assuming that our heuristics are correct, we should see a significant increase in accuracy if they are included. Second, we have not aggressively tuned the classification algorithms. Third, we have not fully explored the space of possible features. As examples, we have neither explored features extracted from the link structure of the web nor from user navigation paths. Finally, the empirical study examines only short histories – for long running tasks, we would expect hundreds of visits to accumulate over time, providing substantially more training data.

The diversity of results across tasks and users suggests that it could be valuable to learn separate models for each task. However, we need to examine the tasks with poor prediction performance closer to understand how to choose algorithms that can consistently adapt to diverse situations.

We believe that the results are promising, although further user studies are necessary to determine exactly what level of accuracy is necessary to build a usable and effective intelligent history user interface.

In this work, there are several sources of potential error other than the standard limitations of the Naïve Bayes and SVM methods. We have relied on users to supply us with binary judgments as to which pages should appear in the history. We chose this route, because it allowed us to account for subjective differences between users in the ways that they might use history lists. We considered tracking what webpages were actually revisited over time, but that has the significant problem that there are many pages that users may visit repeatedly, but are not valuable to return to, such as a form submission URL for a login page. With our

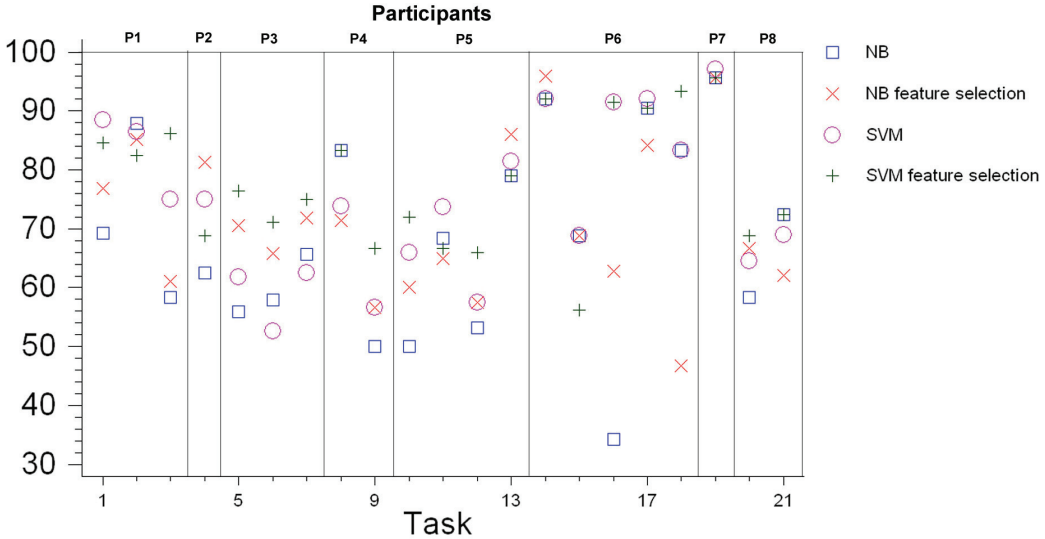

Figure 3. Accuracy per task, grouped by user.

approach, we must keep in mind that users may perceive the value of pages for revisit differently than what their actual behavior suggests.

Another source of error comes from the fact that the technique that we applied assumes that all mistakes have equal cost. The reality is that some pages in the history are much more likely to be revisited than others. An alternative approach would be to ask users to assign weights or multi-valued ratings to each item in their history lists. Then we could better identify the features that distinguish webpages that *must* be kept versus those that it would be nice to keep.

Conclusion

Current web history mechanisms log an overwhelming number of webpages, significantly decreasing their utility for re-locating and revisiting information seen on previously visited pages. We believe that we can dramatically revive the utility of web histories by our approach of first segmenting webpage history by task, then applying machine learning to filter out pages for each task that will not be revisited. If we can reduce the costs of re-locating information, then we can substantially reduce the costs incurred while web browsing. The results from the experiments described in this paper provide evidence that if the user interface is designed to enable collecting or inferring training data from the user, we can successfully predict which pages they are likely to want to keep for revisits.

Acknowledgements

This project was supported in part by the National Science Foundation under grant IIS-0133994 and by the Defense Advanced Research Projects Agency under grant no. HR0011-04-1-0005 and contract no. NBCHD030010.

References

Badi, R., Bae, S., Moore, J. M., Meintanis, K., Zacchi, A., Hsieh, H., Shipman, F., and Marshall, C. C. 2006. Recognizing user interest and document value from reading and organizing activities in document triage. *In Proceedings of the 11th International Conference on Intelligent User Interfaces.* Sydney, Australia (2006). IUI '06. ACM Press, New York, NY, 218-225.

Bellotti, V., Ducheneaut, N, Howard, M & Smith, I. (2003) Taking email to task: the design and evaluation of a task management centered email tool. *In Proceedings of the SIGCHI conference on Human factors in computing systems*. Ft. Lauderdale, Florida, ACM Press.

Canny, J. (2004) GAP: A Factor Model for Discrete Data. *In Proceedings of SIGIR*. Sheffield, 122-129. United Kingdom, ACM Press.

Card, S. K. and Henderson, A. 1987. A multiple, virtual-workspace interface to support user task switching. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics interface* 53-59. Toronto, Ontario, Canada. J. M. Carroll and P. P. Tanner, Eds. CHI '87. ACM Press

Claypool, M., Le, P., Waseda, M., Brown, D. "Implicit Interest Indicators," *In Proceedings of ACM Intelligent User Interfaces*, 2001, pp. 33--40.

Dragunov, A., Dietterich, T. G., Johnsrude, K., McLaughin, M., Li, L. and Herlocker, J. L. Tasktracer: A Desktop Environment to Support Multi-Tasking Knowledge Workers. *In Proceedings of the International Conference on Intelligent User Interfaces*, 75-82. San Diego, Calif. 2005

Fenstermacher, K. & Ginsburg, M. (2002) A Lightweight Framework for Cross-Application User Monitoring. *Computer*, 35, 51-59.

John, G. H. & Langley, P.. 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. pp. 338-345. Morgan Kaufmann, San Mateo.

Jones, W.P., Bruce, H. & Dumais, S. T. (2001). Keeping found things found on the web. *In Proceedings of ACM's CIKM'01, Tenth International Conference on Information and Knowledge Management*, 119-126.

Kaptelinin, V. (2003) UMEA: Translating Interaction Histories into Project Contexts. *Proceedings of the SIGCHI conference on Human factors in computing systems,* 53 - 59 . Ft. Lauderdale, Florida, ACM Press.

Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K., Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 13(3), pp 637-649, 2001.

Kelly, D. and Belkin, N. J. 2004. Display time as implicit feedback: understanding task effects. *In Proceedings of the 27th Annual international ACM SIGIR Conference on Research and Development in information Retrieva.l* Sheffield, United Kingdom, 2004. SIGIR '04. ACM Press, New York, NY, 377-384.

Platt, J. 1998. Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in Kernel Methods - Support Vector Learning,* B. Schoelkopf, C. Burges, and A. Smola, eds., MIT Press.

Porter, M. An algorithm for suffix stripping. *Program,* 14(3):130-137, 1980

Robertson, G., Van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. & Gorokhovsky, V. (2000) The Task Gallery: A 3D Window Manager. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* 494 - 501. The Hague, The Netherlands, ACM Press.

Shen, J. , Li L., Dietterich, T. G., Herlocker, J. L.. A hybrid learning system for recognizing user tasks from desk activities and email messages. *In Proc. of International Conference on Intelligent User Interfaces*, 86 – 92. Sydney, Australia, 2006.

Tauscher, L. and Greenberg, S. (1997) How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies*, Special issue on World Wide Web Usability, 47(1), p97-138, Academic Press.

Witten I. and Frank E. (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005