Correlation-Based Content Adaptation for Mobile Web Browsing

Iqbal Mohomed, Adin Scannell, Nilton Bila, Jin Zhang, and Eyal de Lara

Department of Computer Science
University of Toronto
{iq,amscanne,nilton,delara}@cs.toronto.edu,
jinyaozhang@utoronto.ca

Abstract. The resource impoverished environment on mobile devices results in a poor experience for users browsing the World Wide Web. Proxy-based middleware that transform content on the fly to better suit the resource conditions on a user's device provide a promising solution to this problem. A key challenge in such systems is deciding how to adapt content, especially when the same content has multiple uses that have varying adaptation requirements. In this paper, we show that it is possible to provide fine grain adaptation of multi-purpose content by detecting correlations in the adaptation requirements of past users across multiple objects on a web site, and using this history to make adaptation predictions for users encountered subsequently. To evaluate our technique, we built prototype page layout and image fidelity adaptation systems, and used these to gather traces from users browsing multi-purpose web content in a laboratory setting. Our experimental results show that using correlations to make adaptation predictions can significantly reduce bandwidth consumption, browsing time, energy usage and user effort required to adapt content.

Keywords: Content Adaptation, Mobile Devices, Customization, Web Browsing, Experimentation.

1 Introduction

The severe resource constraints on mobile devices make browsing the World Wide Web an unpleasant experience for users. At present, the majority of content on the Web is targeted towards use on desktop computers with ample displays and high-speed connections to the Internet. These assumptions do not hold in a mobile environment, where devices have small screens, low-bandwidth, limited battery capacity, processing capabilities, I/O facilities and storage. The problem of mobile web access is further complicated due to the considerable heterogeneity among different classes of devices (laptops, PDAs, cell phones, pagers, etc.). Also, as users move about naturally during the course of their activities, the mobile computers they carry with them experience significant variability in wireless connectivity – at one moment the user may be in range of an accessible well-connected, lightly loaded 802.11g access point, whereas at other times, she may only have access to a WWAN service (such as GPRS, CDMA 1X, etc.) that charges her based on the number of kilobytes that are transferred over the link.

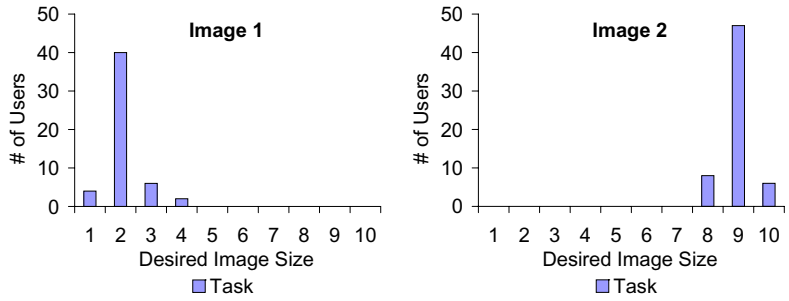
R. Cerqueira and R.H. Campbell (Eds.): Middleware 2007, LNCS 4834, pp. 101–120, 2007.

A promising solution to these problems is adaptation middleware, interposed in the network path between the client and web server, which automatically tailors content for individual mobile devices [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. For example, images on web pages can be served to the user at a reduced fidelity in order to conserve bandwidth and energy, and improve download times. Also, the layout and size of content objects (such as images) can be changed to better fit on a small display. However, a key challenge in such systems is *how* to identify appropriate adaptations. This is a difficult problem because optimal adaptation depends on the usage semantics of content (the user's purpose vis-à-vis the content) as well as the user's context (characteristics of the user's device as well as their surroundings).

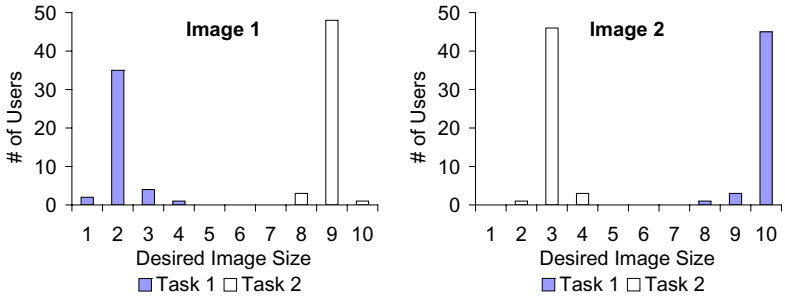
In previous work [11, 12, 13], we introduced Usage-Aware Interactive Content Adaptation (URICA), an automatic adaptation technique that customizes content for mobile devices based on the content's usage semantics and the user's context. URICA learns how to adapt content from implicit feedback provided by users carrying out their tasks. This is achieved by having the system make an initial adaptation decision, and allowing users who are unsatisfied with the system's adaptation decision to take control of the adaptation process and make changes (e.g., increase the fidelity of a transcoded image or change the layout of a page). The successful adaptation is recorded and used in making future adaptation decisions for the same and other users. URICA works well when users utilize content in a similar manner. For example, Figure 1(a) shows histograms of image display sizes that satisfied users for two distinct images in a system that scales the dimensions of images to fit on a small screen. Here, making predictions using the history of individual objects works well; we see that presenting Image 1 at size 2 and Image 2 at size 9 will satisfy the majority of users. However, URICA is less effective for multi-purpose content, where objects on a web page are used for different tasks with varying adaptation requirements. Figure 1(b) illustrates the case when users can perform one of two tasks on a page. For the first task, they require a small version of Image 1 and a large version of Image 2, while these requirements are reversed in the second task. Here, if we only consider the history of the object that is being adapted, there is no single adaptation that will satisfy all users.

Fortunately, typical web tasks involve more than one object. This paper shows that for web tasks that involve multiple objects, it is possible to leverage the feedback provided by the user on a few initial objects to narrow the history used to make subsequent predictions to include only those users who have similar adaptation requirements. This is achieved by finding correlations in adaptation requirements *between different objects* on a web site using the history of previously encountered users. Once these correlations are uncovered, the interactive feedback provided by the user to adapt some objects can be used to adapt other related objects on the page or site. For example, for the content depicted in Figure 1(b), we can see from the adaptation history of the two images that the sizes of Image 1 and 2 are inversely correlated. Once this determination is made, if a user increases the size of Image 1, the system can automatically decrease the size of Image 2.

Correlation-based prediction works well for multi-purpose content because, while users can utilize the same content in different ways, it is quite likely that there are *at least some users who use the content in each of the different ways*.



(a) Single-Purpose Content



(b) Multi-Purpose Content

Fig. 1. Histograms of image display sizes that satisfied past users. The vertical axis shows the number of users who desired each of the adaptations on the horizontal axis. When content is single-purpose (*a*), one adaptation decision works well for most users. For multi-purpose content (*b*), there may be correlations in the adaptation requirements of users across objects.

Correlation-based predictions can also be useful in the case of single-purpose content when users have different context. If the context in question affects adaptation requirements, the adaptation history of individual objects will be noisy just as in the case of multi-purpose content. That is, a single adaptation will not satisfy users with different context. In such situations, adaptation based on correlations will also be beneficial.

We experimented with two well-known machine learning techniques that enable correlations between objects to be uncovered automatically: Decision Stumps, which directly encodes relationships between the adaptation requirements of objects, and the Gaussian Mixture Model, which finds correlations implicitly by clustering users with similar adaptation requirements. Our experience showed that these techniques only perform well when users have an incentive to fix incorrect adaptation decisions made by the system. For instance, in a system that adapts the dimensions of images, users have a clear incentive to correct images that are larger or smaller than what they require. However, such incentives may not always exist. For instance, in a system that adapts image fidelity, if the initial set of images on a site is served to the user at a fidelity that is greater than that required, there is little incentive for the user to interact with these images to lower their fidelity given that the bandwidth to transfer the images would have already been spent. For these situations, we developed an algorithm called *all-in* that

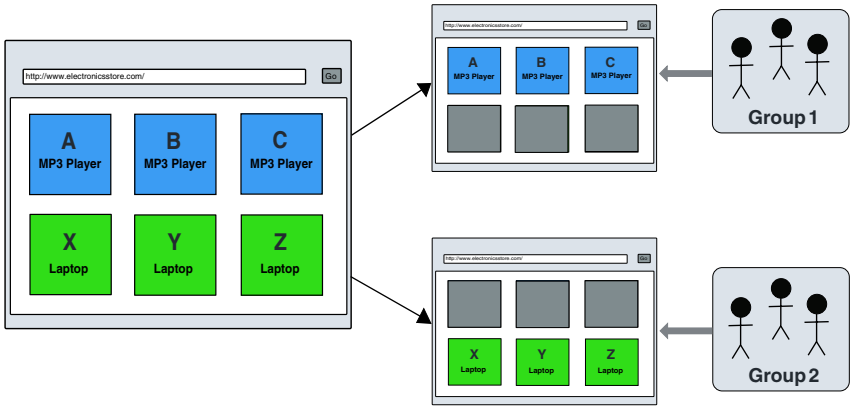


Fig. 2. A schematic of a web page with images of MP3 players and laptop computers. Users in group 1 are shopping for MP3 players, whereas those in group 2 are shopping for laptop computers. If the user improves the quality of one image of an MP3 player, it is likely that they will want the other MP3 player images at high quality as well (the same holds for laptop images).

clusters together the histories of past users with similar adaptation requirements, and as a user provides feedback, it rapidly narrows down a cluster of users with similar adaptation preferences.

We built two prototype adaptation systems for the purpose of evaluation. One scaled the dimensions of images on web pages and the other adapted their fidelity. We collected traces from users browsing multi-purpose content using our prototypes in a laboratory setting, and used these to evaluate the performance of alternative algorithms for making adaptation predictions. We found that making adaptation predictions using correlations results in significant performance improvements. For image scaling, we observed that using correlations to make predictions required 66% fewer user interactions. In the case of image fidelity adaptation, we observed that correlation-based predictions reduce bandwidth consumption by 63%, user interactions by 48%, energy consumption by 17% and time to completion by 20%.

The rest of this paper is organized as follows. Section 2 describes algorithms for multi-purpose content adaptation. Section 3 provides a description of our experiments. Section 4 presents the result of our evaluation. Finally, Section 5 discusses related work, and Section 6 concludes the paper and suggests avenues for future work.

2 Adapting Multi-purpose Content

When the same content can be used for multiple purposes, users may have varying adaptation requirements for the same object based on the particular task they are trying to perform on a web page or site. For example, Figure 2 shows the schematic of a web page that contains six images and represents the front page of an online retailer that sells MP3 players and laptop computers. The first three images on the page show MP3 players and the next three show laptop computers. A user who pays for downloads by

the kilobyte will most likely not want to see all images at high quality - users shopping for MP3 players will want to see the top three images at a higher quality than the others (and vice versa). In this case, the varying adaptation requirements of different users leads to a noisy history for every object - no single adaptation will satisfy all users.

Fortunately, typical web tasks involve more than one object, and when there is correlation in the adaptation requirements of these objects, the feedback provided by users on a few objects can be used to adapt others. Continuing our previous example, the system can determine that users who want any one of the MP3 player images at high quality will likely want the other two at a high quality as well (the same holds for laptop images). The adaptation system would initially serve all images at low quality, and as soon as the user requests an improvement for one of the images (say, an image of an MP3 player), the system can improve the quality of not just that image, but also the quality of other images whose quality levels are highly correlated (the other images of MP3 Players).

It is important to note that objects do not have to be tagged with any meta-data by the content creator for this approach to work. Nor are users required to explicitly specify the task they are performing. The only information required is the history of how users have adapted objects on the page in the past, which is gathered automatically by the system at run-time.

In this section, we start by describing the two types of implicit feedback that can be provided by users in interactive adaptation systems: *two-sided feedback* and *one-sided feedback*. The type of feedback available plays a crucial role in the design and performance of algorithms that predict adaptation requirements. We then describe three algorithms for taking advantage of correlations in user preferences.

2.1 Type of Feedback

The type of adaptation being performed influences the nature of feedback provided by users. In some cases, the constrained resource cannot be recovered when an adaptation decision results in overconsumption. In such situations, users have no incentive to provide additional feedback to the adaptation system. That is, users only provide feedback until the adapted object is "good enough". We call this *one-sided feedback*. For example, in image fidelity adaptation, if the system serves an image at a fidelity that is lower than what is desired by the user, we can expect the user to interact with the system to obtain a higher fidelity representation. However, if the system provides a representation that is of a higher fidelity than that which is required, the user has no incentive to provide feedback. This is because the cost of downloading the higher quality representation has already been incurred. Thus, if the system provides an image at some initial fidelity level (say, level 5, where there are 10 fidelity levels in total) and the user does not improve the object, we cannot say for certain that the user required fidelity 5. We only know that the user may have desired a fidelity between 1 and 5.

In other cases, where overused resources can be reclaimed, users are motivated to keep interactively adapting an object until it has been appropriately customized. We call this *two-sided feedback*. For example, in image screen size adaptation, if the system overuses the screen real-estate resource, it can be reclaimed. Users have an incentive to shrink and enlarge images until they are suitable for their purpose.

2.2 Prediction Algorithms

We initially investigated two standard techniques from machine learning that enable correlation-based predictions: Decision Stumps, which directly encodes relationships between the adaptation requirements of objects, and the Gaussian Mixture Model, which finds correlations implicitly by clustering users with similar adaptation requirements. We observed these techniques to perform well when users provide two-sided feedback. However, when only one-sided feedback is available, these algorithms can perform badly if the system over-predicts on the initial set of objects on the page. For example, in a system that adapts image fidelity, if the initial set of images on a site is served to the user at a high fidelity, there is little incentive for the user to interact with these images to lower their fidelity, and the system cannot accurately gauge the user's adaptation requirements. This problem can be overcome by under-predicting on the initial set of objects as a way of *probing* for the user's true adaptation requirements. However, this can require the user to frequently interact with objects. To address this problem, we developed an algorithm called *all-in* that under-predicts without causing an excessive number of interactions.

Decision Stumps: In order to investigate the effectiveness of directly correlating the adaptation requirements of different images, a method using decisions stumps [14] for predicting adaptation requirements was implemented. A decision stump is a decision tree with only a single branch. In reference to the motivating example, it encodes a decision of the form: Was the required fidelity for image $X < 5$? If yes, then a fidelity of 5 is sufficient for image Y; otherwise 5 is not sufficient for Y. Several decision stumps (alternatively, they may be thought of as *rules*) are weighted and combined into a final model, which is used to make predictions. Each decision stump in this model represents some relationship between the object whose adaptation requirement is being predicted and some other object. The weighting of the decision stumps is calculated during training in order to minimize error; it may be thought of as specifying the relative predictive ability of each relationship. Due to the multiple decision stumps that compose a single model, more than one relationship can be captured and the multi-purpose nature of any given object preserved.

For each object (call this the target object) and every subset of the non-target objects, a distinct prediction model is generated. This model is generated by feeding the history of all user adaptation requirements for the given set of non-target objects along with the corresponding adaptation requirements for the target object into a training procedure. This training procedure uses boosting [15] which generates a set of decision stumps and weights that predict adaptation requirements with a low error rate. A model for each subset must be generated because, as we are encoding correlations directly, a prediction for an image X may be based on different images, depending on the set of objects for which the user has provided some feedback.

Predictions are made by selecting the appropriate model for the target image, and providing as input the already-specified set of required fidelities. For example, suppose we are generating predictions for the electronics retailer used in the motivating example of this section. Suppose the user has seen and possibly interacted with two images, X and A, and we must now predict a fidelity for the image Y. First, we retrieve our model

that was trained with adaptation requirements for X and A and predicts Y. Based on the current user's requirements for X and A, we predict an appropriate requirement for Y. Since this model is a combination of decision stumps that involve rules regarding the required fidelities of X and A, we can simply evaluate them all and determine a final score. This final score corresponds to the predicted required fidelity. This process is repeated with all remaining images other than Y, as predicted images are loaded and the user provides feedback.

In the evaluation section, the use of this model with both one-sided and two-sided feedback is explored. This method may encode many complex relationships between objects, and requires no specification of parameters in advance (such as number of clusters). Unfortunately, the cost of training and generating the large number of models for this method may be high, although there could exist optimizations to alleviate this problem. Another disadvantage of this method is that it may also have a tendency to over-fit training data, especially for users with non-typical adaptation requirements. This may manifest itself as a single out-of-character requirement given by a user throwing off several predictions due to over-emphasis on a particular image.

For our implementation of this method, we used the MultiBoost [15] algorithm implemented by the Weka [14] toolkit. The MultiBoost algorithm combines AdaBoost [16] with wagging, and it was shown to be more effective in reducing error than either of its constituent techniques [15].

Gaussian Mixture Model: In a Gaussian mixture model, all sets of adaptation requirements are assumed to be sampled from a set of Guassian distributions spread throughout the space of all possible adaptation requirements. Given a set of training data, the parameters of the distributions are set by running an expectation-maximization (EM) algorithm in order to maximize the likelihood that the given data was sampled from the mixture of distributions. As input for this training procedure, all available history of user adaptation requirements is provided. The number of distributions must be selected a priori, however.

For prediction, based on a user's currently specified set of adaptation requirements and the training distributions, a candidate distribution for the user is selected by computing the likelihood of her belonging to each distribution, then selecting the most probable. The mean of this candidate distribution is used to provide any missing adaptation requirements. If this mean is insufficient for the user for some particular object, the most probable distribution with a higher adaptation requirement is selected for that object instead. Eventually, these adaptation requirements which are not well-represented may lead to the selection of a better candidate distribution.

Similar to the scenario given for decision stumps, suppose that we are serving images for users browsing the online electronics retailer. The user has seen and possibly interacted with the images X and A and we must now predict image Y. Based on their required fidelities for X and A, the probability of the user belonging to each Gaussian distribution d, $p(d)$, is computed. This is a calculation over only the images which the user has seen (X and A), in this case given by

$$p(d) = \alpha_d \prod_{i=\{X,A\}} \frac{1}{\sigma_d(i)\sqrt{2\pi}} exp(-\frac{(x_i - \mu_d(i))^2}{2\sigma_d(i)^2})$$

for each distribution d with means $\mu_d(i)$, standard deviations $\sigma_d(i)$ and prior α_d (all set by the EM training procedure). The distribution with the highest $p(d)$ is selected, call this d_{best}, and the means of d_{best} are used to provide a prediction for the adaptation requirements of other images. In this case, since we need to predict Y, we would use $\mu_{d_{best}}(Y)$, the mean of the distribution d_{best} for the image Y.

Logically, these distributions can be thought of as the center of clusters. During the training procedure, they will tend to each cover different groups of users' adaptation requirements. This model is representable in a compact way, and has the advantage that each distribution, or cluster, has an explicit variance for each object. This is useful for intentional overprediction and underprediction, based on user preference. For example, the user may favour underprediction in order to conserve bandwidth. Also, since the identification of a candidate cluster is based on all objects, a single odd requirement from a user is likely to have less of an impact on other predictions than in the case of decision stumps.

The *all-in* Algorithm: We designed the *all-in* algorithm[1] for use when only one-sided feedback is available, and investigate it in the context of image fidelity adaptation. The algorithm starts off by using the standard K-means clustering algorithm to partition users into multiple groups. The idea is that users within a group share similar adaptation requirements - not just for a single object but rather across all objects on a web page or web site. Once we have a set of clusters, the system transitions into prediction mode. The system then uses an online classification algorithm to make predictions.

The goal of the *all-in* algorithm is to rapidly classify the user into a single cluster. For each user, the adaptation decisions made by the system early on are aggressive in that they may be wasteful. However, once the system is able to correctly classify the user, it starts making moderate predictions, such as serving the mean of the image fidelities that was requested by other users within a cluster.

At the outset, the algorithm assumes that the user can belong to any cluster. Also, it computes an upper and lower threshold for each object in every cluster. These thresholds correspond to the range of values where an object's desired fidelity may lie, for users belonging to this cluster. We take the highest and lowest fidelity that was previously observed as the upper and lower threshold, respectively[2]. When serving an image initially, the algorithm makes an aggressive prediction: it serves the image at the lowest upper threshold across all clusters. If the user is not satisfied with this adaptation, she will request an improvement and the system will remove the cluster whose upper thresholds are violated. The process is repeated until the user no longer requests improvements to an object, and moves on to a different page or object. In this case, the system checks if there exist any clusters whose lower thresholds are violated, and removes them. Once the system has classified the user into a single cluster, the algorithm behaves less aggressively, and serves objects at the mean of the image fidelities that were requested by other users within the cluster.

[1] The phrase "all-in" is taken from poker where a player bets his entire stake on a hand. When there are only two players, this move forces the opponent to evaluate her hand and make a decision on whether to accept the bet ("call") or give up the hand ("fold").

[2] Other alternatives are possible, such as taking the endpoints of the 5-95 percentile range. This would help eliminate outliers in a production system.

It is possible for the system to reach a point where there is no cluster that the user can belong to. This can occur for two reasons: first, there may be no cluster that captures the current user's preferences or second, we may have removed the user from a cluster that she would otherwise have fit into because her adaptation preference on some prior object was radically different. With regards to making adaptation predictions for this particular user, we can do nothing about the first possibility. However, we can address the second by making all clusters valid again for the user. For the particular object under consideration, we give up, and serve it without any adaptation. In a production system, we can take the first possibility into account as well. Any time the system runs into a large number of users who cannot be classified, it can transition into training mode again, and regenerate the clusters.

2.3 Practical Considerations

Parameters, such as the number of clusters to use in the *all-in* algorithm, can be automatically determined in a production system. Once the system has encountered some number of users (say T, specified by the operator of the adaptation proxy), it can run profiling experiments that compare the performance that would have been experienced by the previously encountered users in different conditions. The experiments may compute a variety of performance metrics for different parameter settings, and the system can set parameters to be the values that result in the best performance. A single metric or a composition thereof can be used for this purpose, based on the goals of the proxy operator or the preferences of users. Indeed, if users specify different goals to the adaptation system, it can provide them with varying predictions tailored to their requirements based on the same history.

3 Experimental Methodology

To evaluate our prediction algorithms, we considered two types of adaptation: page layout and image fidelity. For each type of adaptation, we created a prototype that allows users to interactively adapt content. We used the prototypes to perform experiments in which participants adapted content in a laboratory setting. The traces of the user's adaptation decisions were then used to evaluate the prediction algorithms.

In this section, we first describe our trace gathering experiments. We then discuss the methodology used to evaluate the prediction algorithms on the collected traces.

3.1 Gathering User Traces

We conducted our experiments in a laboratory at the University of Toronto. For the experiments we recruited three groups of participants from the general student population. The first group adapted the layout of web pages, while the second and third groups adapted the fidelity of images on web pages. Table 1 summarizes the setup of the experiments. During the experiments, the prediction component of the adaptation system was disabled so that participants would have to interact with images in order to achieve an appropriate adaptation. That is, the system did not take advantage of past

interactions of the current or previous users. This forces participants to reveal their true adaptation preferences as well as avoiding any effects arising from the ordering of participants in our study.

Page Layout Adaptation Experiment: The goal of our first experiment was to investigate a scenario in which users naturally provide two-sided feedback. We created a prototype page layout adaptation system that allowed users to increase and decrease the screen dimensions of images on a web page. While all of the participants in this study were given the same task to perform, we varied the device used to browse the web across individuals. Thus, the primary source of variation in the adaptation requirements of users is the difference in device context.

The experiment consisted of four web pages, each containing three images of postage stamps. For each page, participants were asked to modify the dimensions of the images in a manner such that it would be easy to identify differences between two images and find details on a third image. We obtained traces from 30 participants who were randomly divided into three sub-groups which used different simulated displays: a PocketPC SmartPhone, a PocketPC PDA and a Toyota GBook vehicular terminal. The setups for these traces are referred to in Table 1 as SmartPhone, PDA and GBook, respectively.

Image Fidelity Adaptation Experiments: The goal of our second and third experiment was to consider a case where users are only motivated to provide one-sided feedback. To this end, we created an image fidelity adaptation system in which the images on a web page are initially served at low fidelity (for faster download), and users can click on individual images to improve their fidelity. In these studies, different participants were given varying tasks. However, all of the participants performed their assigned tasks on the same device. As such, variations in the adaptation requirements of users stem from differences in their assigned task.

For these experiment, we designed two image-rich sites. The first, a movie posters site, had images of popular movie posters. The second, a map site, had a map of the University of Toronto's campus represented in a grid of 6 x 6 images. For each site, we designed three tasks, and each participant performed only one of those tasks. For the movie posters site, each task consisted of detailed questions pertaining to a different subset of the posters. For example, participants were asked to identify the director, title and release date of some of the movies. For the map site, participants were asked to provide directions from one given building to another within the university's campus. To accomplish these tasks, participants had to increase the fidelity of relevant images until sufficient details were visible. Participants were able to adjust image fidelities on a scale between 1 and 10. The tasks were designed so that participants would find some images in a web page relevant while others not as much.

For these experiments, participants used a laptop equipped with our adaptation system and an available network bandwidth of 56kbps, which is a reasonable approximation of a GPRS WWAN connection. We recruited 231 participants who were divided in six sub-groups of 37 to 40 individuals. Our setup is described in Table 1.

Table 1. Summary of experiments

	# of Pages	Images per Page	Total Images	Setup	# of Users
Page Layout					
Postage Stamps	4	3	12	SmartPhone Display	10
				PDA Display	10
				GBook Display	10
Image Fidelity					
Movie Posters	9	1	9	Task-1	37
				Task-2	37
				Task-3	37
Map	1	36	36	Path-1	40
				Path-2	40
				Path-3	40

3.2 Trace-Based Evaluation

In order to determine the effectiveness of each algorithm considered, we evaluated them using the traces collected from the participants in our experiments. For the page layout experiment, we collected for each participant, their required image dimensions for every image. From the fidelity experiment, we obtained for each participant their minimum required fidelity for every image.

To test each algorithm, we used leave-one-out cross-validation. That is, each algorithm was trained with the traces of all users except one. The algorithm was then used to predict the dimensions or fidelities of the images served to the user, depending on the experiment.

For this testing, we created a user simulator. At the start, the prediction algorithm provides an adapted version of each image on a page. The simulated user, based on the collected traces, goes through each of the images on the page in turn and provides an "interaction" for the first image it finds that is not properly adapted. When the simulated user provides an interaction, the prediction algorithm recalculates an appropriate adaptation for all of the images on the page and presents it to the simulated user once again. This process is repeated until all of the images are adapted according to the user's preferences.

For the page layout adaptation experiment, the primary metric used to evaluate the different algorithms is the number of user interactions. However, for the fidelity experiments, a number of metrics are used for evaluation: the number of user interactions required, fulfillment time, wasted bandwidth and energy consumed. Number of interactions is the number of times a user had to interact with the images in order to achieve her desired adaptation. Fulfillment time is the aggregate of interaction time (the time users spend interacting with images until their fidelity requirements are met) and download time. Wasted bandwidth is calculated as the amount of bandwidth used beyond what would be required by the user if all images were served at their exact required fidelity immediately. Energy consumed is the energy measure, in Joules, consumed by the device for viewing and downloading content.

In order to compute fulfillment time and energy consumption with our simulator, we measured the average interaction time from one of our user studies (2388 milliseconds).

We then ran several experiments on an HP iPAQ h6325 PDA in order to measure download speeds and energy characteristics of real hardware. With a GPRS connection, we observed effective download speeds of approximately 33kbps. When the device was idle, it consumed 0.67 Joules/second (with GPRS radio and backlight on) and when the device was downloading, it consumed 1.59 Joules/second.

For any particular algorithm, there is a clear trade-off between wasted bandwidth and the number of interactions: under-predicting the fidelity required for an image will lead to more user interactions and over-predicting the fidelity will lead to wasted bandwidth. However, good algorithms can perform well at both simultaneously. Indeed, a perfect prediction algorithm that knows the exact adaptation required by users (we call this *oracle*) would not waste any bandwidth, nor would it require any interactions by the user.

4 Experimental Results

In this section, we provide the results of our evaluation. We start by considering the case of two-sided feedback, which occurs naturally during the course of our page layout experiment. Next, we consider the performance of different algorithms when only one-sided feedback is available, as is the case in our fidelity adaptation experiments.

All of the results presented in this section are mean results, averaged across individual users over the entire web site for any given experiment. The algorithm that we use as our baseline for performance is the *single object history* (SOH) prediction algorithm from our previous work [12,13]. This algorithm makes adaptation predictions for each object by considering its adaptation history in isolation. For image fidelity adaptation, the SOH algorithm initially serves an image at the mean value of the fidelity that was desired by previously encountered users. If this is not satisfactory, the SOH algorithm provides a subsequent prediction by ignoring the desired fidelities below that which was just served, and recomputing the mean. For page layout adaptation, the initial prediction of the SOH algorithm is computed in the same way (taking the mean of the desired image sizes of previously encountered users). However, when a user decides to increase or decrease an image, the algorithm removes from the history all of the desired image sizes of previous users that are less than or greater than the size that was just provided, respectively. SOH makes the next prediction by computing the mean value from the remaining history.

4.1 Two-Sided Feedback

We tested both the decision stump algorithm and the Gaussian mixture model algorithm on the postage stamp experiment, where users were required to adapt images by resizing them. For this experiment, feedback was provided for predictions that were too high or too low. Because the images were already downloaded, there was no notion of bandwidth wasted for this experiment. Using the SOH algorithm, the mean number of interactions required of a user during the experiment was 15. By leveraging correlations between adaptation requirements however, the decision stumps algorithm achieved a mean of 5.1 interactions, while the Gaussian mixture model achieved 5.9 (with six distributions), both demonstrate a vast improvement over using only SOH.

Table 2. The performance of several variations of decision stumps on the movie poster dataset. We observe that significantly more bandwidth is wasted with one-sided feedback (*line 2*) than in the hypothetical case of perfect feedback (*line 1*). We also observe that under-prediction greatly reduces wasted bandwidth but comes at the cost of more interactions required of the user (*line 3*).

Variation	# of Interactions	Bandwidth Wasted (KB)
Perfect Feedback	2.50	200.74
One-Sided Feedback	0.32	830.19
Under-prediction on First Image with One-Sided Feedback	6.71	113.67

These results demonstrate that using correlation based prediction methods for cases where two-sided feedback is available is an excellent idea, and that standard machine learning techniques work well. After all, this a very straight-forward prediction problem.

4.2 One-Sided Feedback

One-sided feedback introduces a twist to the prediction problem. We compare the performance of our different algorithms and explore the effect of under-prediction on the movie posters experiment. We show that the *all-in* algorithm leverages this effect and provides strong performance across all studies where only one-sided feedback is available. Finally, we evaluate the algorithms on the movie posters experiment using two metrics of practical interest: fulfillment time and energy usage.

Without any adaptation, 2.70MB are transferred to download the 9 images in the movie posters experiment. However, if an oracle were to exist such that we were able to provide users with their desired fidelity, only 1.29MB would have been downloaded on average. That is, without adaptation, an average of 1.41 MB of bandwidth is consumed needlessly. Making predictions using single object history results in an average wastage of only 378KB of bandwidth; this occurs at an average cost of 5.4 interactions. When we consider how interactions are distributed across images, we observe that the users must interact with approximately two-thirds of the images on the web site.

In the case of image fidelity adaptation, only one-sided feedback is available. However, in order to establish the validity of the methods in general, we first consider the performance they achieve if users provided perfect feedback. For perfect feedback, we assume the algorithm knows by how much each image was over-predicted, without incurring any additional interactions (under-predictions still result in interactions). We then show results for the case where users provide one-sided feedback.

The first line of Table 2 shows the result of making predictions using decision stumps when perfect feedback is provided. We see that, beyond the single object history case, the amount of wasted bandwidth is reduced by 47%. In addition, the number of interactions is decreased from 5.4 to only 2.5. However, for image fidelity adaptation as the problem is made manifest (only one-sided feedback is available), the wastage increases significantly. The second line of Table 2 shows the result of making predictions under these conditions. Although the number of interactions required is minimal, the wasted bandwidth is significantly higher (830KB) than using the predictions generated by the single object history method (378KB).

Figure 3(a) shows the performance achieved when making predictions using a mixture of Gaussians when perfect feedback is available. The x-axis in the graph indicates the number of distributions that are created based on the observed training data. The y-axis on the left indicates the mean number of interactions required by each user and the y-axis on the right provides the mean wasted bandwidth per user. We observe that after about four distributions, the algorithm achieves consistent performance. With six distributions, users waste 212KB with 1.4 interactions. Like the decision stumps method, this represents a significant improvement over the predictions generated with only single object history which wastes 378KB with 5.4 interactions. When only one-sided feedback is given, the number of interactions remains consistently low, however, the wasted bandwidth climbs above 700KB. Figure 3(b) shows the performance of the Gaussian Mixture Model in this case.

We conclude that when only one-sided feedback is available, the two standard techniques that we considered suffer from poor performance.

Effect of Under-prediction: In situations where users only provide one-sided feedback, the performance of prediction algorithms that use correlations can be improved by purposely under-predicting on the initial set of images on a web page. We now show the performance of the decision stumps and Gaussian mixture model algorithms for the movie posters dataset when we under-predict on the first image.

The third line of Table 2 shows the result of making predictions using decision stumps, but with a purposeful under-prediction on the first image served. We see that the amount of wasted bandwidth is reduced by nearly 70% compared to the case where only single object history is used. However, this comes at the cost of more interactions, 6.7 versus 5.4 in the case of single object history.

Figure 4(a) shows the performance of the Gaussian mixture model with one-sided feedback for the case of six clusters. Due to the nature of the model, it is natural to under-predict on images by some standard deviation of the required fidelities of the object. The x-axis indicates the amount of under-prediction (in terms of the number of standard deviations). Similar to Figure 3, the y-axis on the left and right indicate the average number of interactions required per user and the average amount of

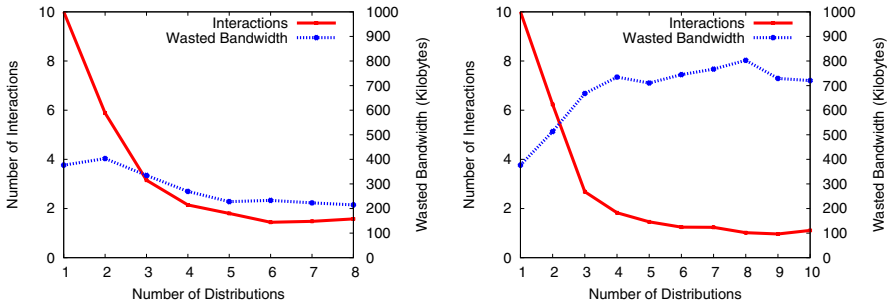
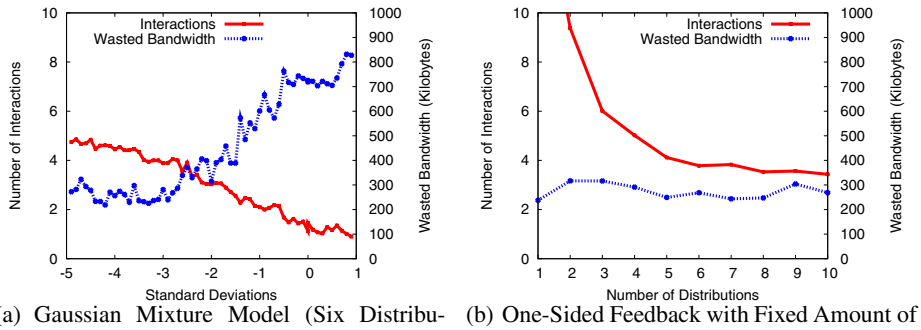


Fig. 3. The performance of the Gaussian Mixture Model on the movie posters dataset with perfect and one-sided feedback. We observe that significantly more bandwidth is wasted with one-sided feedback (*b*) than in the hypothetical case of perfect feedback(*a*).



(a) Gaussian Mixture Model (Six Distributions) with Under-Prediction

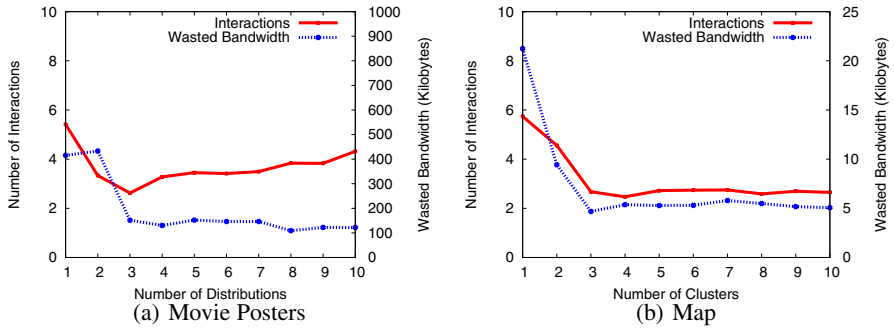
(b) One-Sided Feedback with Fixed Amount of Under-prediction (3.3 std dev)

Fig. 4. The effect of under-prediction on the Gaussian mixture model on the movie posters dataset. We observe that wasted bandwidth decreases as we under-predict with more standard deviations (*a*). For e.g., -4 on the x-axis refers to under-predicting the mean by four standard deviations. Alternatively, for a fixed amount of under-prediction, we observe that GMM has far less wasted bandwidth (*b*) compared to GMM with no under-prediction.

wasted bandwidth per user, respectively. If the algorithm under-predicts by 3.3 standard deviations, compared to using single object history for predictions, users waste 45% less bandwidth and require 1.4 fewer interactions. Figure 4(b) shows the performance of the Gaussian Mixture Model when under-predicting by 3.3 standard deviations for various numbers of distributions.

From these results, we conclude that under-prediction results in a significant reduction in the amount of wasted bandwidth. However, doing so may result in more interactions required of the user.

Performance of the *all-in* Algorithm: Figure 5(a) shows the performance achieved when making predictions using the *all-in* algorithm for the movie posters data set.



(a) Movie Posters

(b) Map

Fig. 5. The performance of the *all-in* algorithm on the studies where only one-sided feedback is provided. We observe that *all-in* performs consistently well on all datasets for both wasted bandwidth and number of interactions.

The x-axis indicates the number of clusters into which users may be classified. These clusters are created using the observed training data. Similar to the previous figure, the y-axes on the left and right indicate the mean number of interactions required by each user and the mean amount of wasted bandwidth per user, respectively. With respect to the number of interactions required, the algorithm performs optimally when there are three clusters. In this case, users waste only 138KB (a reduction of 63% compared to using single object history) at the cost of just 2.6 interactions (2.8 interactions less than single object history).

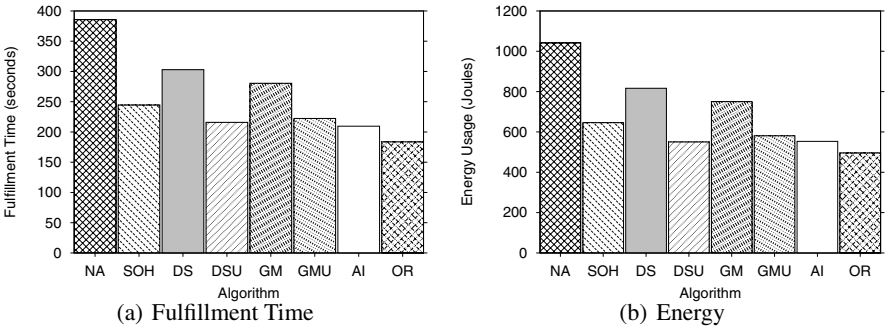
The *all-in* algorithm also performs well on the map experiment 5(b). When we compare the performance of the three methods, we find that the *all-in* algorithm has the best performance. One of the key features of the *all-in* algorithm is that until the user is isolated into a single cluster, the initial prediction made for each object is lower than the average fidelity required by users. When doing correlation-based adaptation with one-sided feedback, purposeful under-prediction for the first few objects provides significant benefit. This is because when the algorithm under-predicts, it forces the user to interact. This leads to an accurate history for a small set of objects, which can be leveraged to provide better quality predictions for the remainder of the objects on a web site.

Energy and Fulfillment Time: To characterize the exact benefits that the different methods may provide in practice, we evaluated all of them and several baseline and naive approaches with respect to fulfillment time and energy consumption. Figure 6 shows the fulfillment time and energy consumption for a number of adaptation policies: no adaptation (NA), single object histories (SOH), decision stumps (DS), decision stumps with under-prediction (DSU), Gaussian mixture model (GM), Gaussian mixture model with under-prediction (GMU), *all-in* (AI) and oracle (OR). Oracle, discussed earlier, is able to exactly predict the user's required fidelity, wasting no bandwidth nor requiring any interaction. It gives an upper bound on the performance of prediction algorithms.

For both fulfillment time and energy, we see that under-prediction results in significant improvement for both decision stumps and Gaussian mixture model. *all-in* performs the best for both fulfillment time and energy consumption, and performs close to oracle. In all cases, the correlation based approaches that use under-prediction offer both better fulfillment time and energy usage. Of all of them however, *all-in* also requires the fewest interactions.

4.3 Summary of Results

We first considered the performance of our prediction algorithms on an adaptation problem where two-sided feedback is available. We found that, in this case, all correlation-based techniques perform better than if we were to make predictions using only SOH. We then considered the algorithms' performance in the case where only one-sided feedback is available and found that for both the decision stumps and Gaussian mixture model algorithms, over-predictions on the initial set of objects lead to poor predictions for later objects. We modified these algorithms to perform under-prediction, which while reducing wasted bandwidth significantly, burdened the user by requiring more interactions. The *all-in* algorithm performed consistently well, even when only one-sided feedback was available, due to its aggressive under-prediction.



(a) Fulfillment Time (b) Energy

Fig. 6. The average fulfillment time and energy consumption per user for several adaptation techniques on the movie posters dataset

Finally, we considered the fulfillment time and energy consumed during the movie posters experiment by each of the algorithms. We found that the *all-in* algorithm outperformed all others, and provides effective fine-grain adaptation even in the case of multi-purpose content.

5 Related Work

There is significant research on content adaptation for mobile devices [1, 2, 3, 17, 18, 4, 5, 19, 20, 6, 8, 9, 10], and even a few commercial adaptation systems have been deployed [21, 1].

Content providers have traditionally adapted content manually, by offering device specific versions of their content. This approach places significant overhead on content providers as they need to maintain multiple versions of their content.

There has also been research on systems which automatically adapt content on-the-fly. Most automatic systems generate adaptation policies either based on rules [6, 7, 1, 22, 20] or constraints [7, 4, 23, 19]. In both approaches, adaptation policies are defined using high-level programming languages or mathematical formulas [23,22]. Rule-based systems rely on high-level rules to guide the adaptation process. When adapting an object, the system determines the subset of rules that apply and adapts accordingly (e.g., convert images larger than 50 KB to progressive JPEG images). Constraint-based adaptation extends rule-based adaptation to encode tradeoffs between possible adaptation strategies. A constraint captures, in a mathematical formula, the relationship between resource consumption and user satisfaction for a specific adaptation. An automatic solver adapts content by finding a solution that meets all constraints, minimizes resource consumption, and maximizes user satisfaction. Unfortunately, content providers cannot be expected to provide constraints or rules for every data object, as this imposes significant onus. As a result, small sets of rules apply to broad sets of content (e.g., all JPEG images are adapted the same way independent of their purpose or value to the user). Moreover, determining the relationship between user satisfaction and content metrics, such as resolution or frame rate, is hard and often

depends on the semantics of the content being adapted and the user's task, which is rarely taken into consideration in these approaches.

In contrast, in our approach, end-users provide feedback for only a small subset of the content of web pages (by clicking on the objects), and the system is able to correctly adapt the larger set of content by considering the correlation in adaptation requirements of users. Also, because the end-user has control over the degree of adaptation, the system is guaranteed to provide adaptations that are satisfactory to the user.

End-user adaptation is also explored in [24, 25], however, those systems provide solutions specific to the layout of web pages on small screens, and as such do not explore correlations in user adaptation requirements.

Our work is related to previous efforts on recommendation-based systems. Most recommendation systems [26, 27, 28, 29] use collaborative filtering, in which people collaborate to help one another perform filtering by recording their reactions to documents they read. Balabanovic et al. [30] add the ability to evaluate and provide feedback in order to learn and improve on the recommendations. A collection of histories [31] can be created and then mined to recommend to the user a set of candidate functions and to detect users' erroneous behavior. Semantics can be used to build a model of the user [32] such as that used by online retailers like Amazon.com, which can then be used to recommend other items in the same class of products.

In our previous work, we introduced the URICA technique, which adapts single-purpose content based on the history of previously encountered users [12], and considers the context of those adaptations [13]. In this paper, we have shown how to provide fine-grain adaptation in the more challenging case of multi-purpose content. This is achieved by finding correlations in user adaptation requirements *between different objects* on a web site, and leveraging a user's feedback across multiple objects.

6 Conclusions and Future Work

In this paper, we showed that correlations in user adaptation requirements across different objects can be used to provide fine-grain adaptation for multi-purpose content. We considered two techniques from machine learning that enable correlation-based predictions: decision stumps, which directly encodes relationships between the adaptation requirements of objects, and the Gaussian mixture model, which finds correlations implicitly by clustering users with similar adaptation requirements. These techniques do not perform well when users have no incentive to correct over-predictions made by the system. We provide an algorithm called *all-in*, which groups together users with similar adaptation requirements and then makes predictions in a way that rapidly classifies users into a single cluster. We showed that for one-sided feedback, the *all-in* algorithm performs significantly better than other techniques when considering key metrics such as bandwidth usage, number of user interactions, fulfillment time, and energy consumption.

In the future, we intend to do a large scale, real-world deployment of an image fidelity adaptation system. The goal of this endeavor is to learn about the behavior of users performing interactive adaptation on web content outside a lab environment, and over an extended period of time. A version of this system for devices that can run

the Firefox browser has already been made publicly available [33], and versions for the Minimo and Pocket Internet Explorer web browsers are currently being tested.

Acknowledgment

This research was supported Bell University Labs (BUL) under grant 480997, and by the Canadian Foundation for Innovation (CFI) and the Ontario Innovation Trust (OIT) under grant number 7739. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of CFI, OIT, BUL or the University of Toronto.

References

1. Britton, K., Case, R., Citron, A., Floyd, R., Li, Y., Seekamp, C., Topol, B., Tracey, K.: Transcoding: Extending e-business to new environments. IBM Systems Journal 40(1), 153–178 (2001)
2. de Lara, E., Wallach, D.S., Zwaenepoel, W.: Puppeteer: Component-based adaptation for mobile computing. In: Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, California (2001)
3. Fox, A., Gribble, S.D., Brewer, E.A., Amir, E.: Adapting to Network and Client Variability via On-Demand Dynamic Distillation. SIGPLAN Notices 31(9), 160–170 (1996)
4. Lum, W.Y., Lau, F.C.M.: A context-aware decision engine for content adaptation. IEEE Pervasive Computing 1(3), 41–49 (2002)
5. Noble, B.D., Satyanarayanan, M., Narayanan, D., Tilton, J.E., Flinn, J., Walker, K.R.: Agile application-aware adaptation for mobility. Operating Systems Review (ACM) 51(5), 276–287 (1997)
6. Smith, J.R., Mohan, R., Li, C.S.: Content-based transcoding of images in the Internet. In: Proceedings of the IEEE International Conference on Image Processing, Chicago, Illinois, IEEE Computer Society Press, Los Alamitos (1998)
7. Smith, J.R., Mohan, R., Li, C.S.: Transcoding internet content for heterogeneous client devices. In: Proceedings of the IEEE International Symposium on Circuits and Systems, Monterey, California, IEEE Computer Society Press, Los Alamitos (1998)
8. Sun, Z., Mahmud, J., Mukherjee, S., Ramakrishnan, I.V.: Model-directed web transactions under constrained modalities. In: WWW 2006. Proceedings of the 15th international conference on World Wide Web, ACM Press, New York, NY, USA (2006)
9. Borodin, Y., Mahmud, J., Ramakrishnan, I.: Context browsing with mobiles - when less is more. In: MobiSys 2007. Proceedings of the 5th international conference on Mobile systems, applications and services, pp. 3–15. ACM Press, New York, NY, USA (2007)
10. Zhuang, Z., Chang, T.Y., Sivakumar, R., Velayutham, A.: A3: application-aware acceleration for wireless data networks. In: MobiCom 2006. Proceedings of the 12th annual international conference on Mobile computing and networking, pp. 194–205. ACM Press, New York, NY, USA (2006)
11. Mohomed, I., Chin, A., Cai, J.C., de Lara, E.: Community-driven adaptation: Automatic content adaptation in pervasive environments. In: WMCSA 2004. Proceedings of the Workshop on Mobile Computing Systems and Applications, Lake District National Park, UK, pp. 124–133. IEEE Computer Society, Los Alamitos (2004)
12. Mohomed, I., Cai, J.C., de Lara, E.: Urica: Usage-aware interactive content adaptation for mobile devices. In: Proceedings of EuroSys 2006, Leuven, Belgium (2006)

13. Mohomed, I., Cai, J.C., Chavoshi, S., de Lara, E.: Context-aware interactive content adaptation. In: MobiSys 2006. Proceedings of the 4th international conference on Mobile systems, applications and services, pp. 42–55. ACM Press, New York, NY, USA (2006)
14. Witten, I., Frank, E.: Data mining: Practical machine learning tools and techniques (2005)
15. Webb, G.: Multiboosting: A technique for combining boosting and wagging. Machine Learning , 159–196 (2000)
16. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proc. of International Conference on Machine Learning, pp. 148–156 (1996)
17. Fox, A., Gribble, S.D., Chawathe, Y., Brewer, E.A.: Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. IEEE Personal Communications 5(4), 10–19 (1998)
18. Katz, R.H.: Adaptation and mobility in wireless information systems. IEEE Personal Communications 1(1), 6–17 (1994)
19. Narayanan, D., Flinn, J., Satyanarayanan, M.: Using history to improve mobile application adaptation. In: Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications, Monterey, California, IEEE Computer Society Press, Los Alamitos (2000)
20. Schilit, B.N., Trevor, J., Hilbert, D.M., Koh, T.K.: Web interaction using very small internet devices. IEEE Computer 35(10), 37–45 (2002)
21. iAnywhere Solutions: Avantgo, `http://www.avantgo.com`
22. Han, R., Bhagwat, P., LaMaire, R., Mummert, T., Perret, V., Rubas, J.: Dynamic adaptation in an image transcoding proxy for mobile web browsing. IEEE Personal Communications 5(6), 8–17 (1998)
23. Dotsenko, Y., de Lara, E., Wallach, D.S., Zwaenepoel, W.: Extensible Adaptation via Constraint Solving. In: Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications, Callicoon, New York, IEEE Computer Society Press, Los Alamitos (2002)
24. Bila, N., Ronda, T., Mohomed, I., Truong, K.N., de Lara, E.: Pagetailor: Reusable end-user customization for the mobile web. In: MobiSys 2007. Proceedings of the International Conference on Mobile Systems, Applications and Services, San Juan, PR, USA (June 2007)
25. Baudisch, P., Xie, X., Wang, C., Ma, W.Y.: Collapse-to-Zoom: Viewing Web pages on small screen devices by interactively removing irrelevant content. In: UIST 2004. Proceedings of the 17th Symposium on User Interface Software and Technology, Santa Fe, NM, USA (October 2004)
26. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(12), 61–70 (1992)
27. Terveen, L., Hill, W., Amento, B., McDonald, D., Creter, J.: Phoaks: a system for sharing recommendations. Commun. ACM 40(3), 59–62 (1997)
28. Balabanovic, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM 40(3), 66–72 (1997)
29. CiteSeer, `http://citeseer.ist.psu.edu/`
30. Balabanovic, M., Shoham, Y., Yun, Y.: An adaptive agent for automated web browsing. Journal of Visual Communication and Image Representation 6(4) (1995)
31. Ohsugi, N., Monden, A., Matsumoto, K.: A recommendation system for software function discovery. In: APSEC 2002. Proceedings of the 9th Asia-Pacific Software Engineering Conference, Gold Coast, Queensland, Australia (December 2002)
32. Ghani, R., Fano, A.: Building recommender systems using a knowledge base of product semantics. In: 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, Malaga, Spain (May 2002)
33. Chameleon Homepage, `http://adaptive.slup.cs.toronto.edu/`