

A calendar based Internet content pre-caching agent for small computing devices

Andreas Komninos
Glasgow Caledonian University
70 Cowcaddens Rd.
Glasgow, G4 0BJ, UK
+44 141 331 3095

andreas.komninos@gcal.ac.uk

Mark D. Dunlop
University of Strathclyde
26 Richmond St.
Glasgow G1 1XH
+44 141 548 3497

mark.dunlop@cis.strath.ac.uk

ABSTRACT

We described in earlier publications the principles of a system where internet content would be pre-cached, based on contextual information obtained from a user's electronic calendar. The model for such a system envisioned a set of cooperating agents, distributed on a user's desktop and mobile device, which would be responsible for making decisions on the context and preferences of the user, and downloading the relevant internet content through a land-based broadband connection and storing it on the mobile device. This paper presents and discusses established pre-caching techniques and their suitability for use on mobile information access scenarios. It proceeds in describing the implementation details of an alternative approach, a calendar-based pre-caching system and presents the findings of tests that were made with human subjects on such a system.

Keywords

Mobile Information Access, Electronic Calendars, Internet content Pre-Caching.

1. INTRODUCTION

Motivated by the disparity of desktop and mobile Internet access, both in terms of available bandwidth and in terms of cost, this paper presents research into an alternative method of making Internet content available for mobile users. This method is based on the extraction of contextual information regarding the user's activities and interests using their electronic calendar as a main source, and pre-loading their mobile device with Internet content, using a land-based connection.

The main aim of the research presented in this paper is to investigate whether calendars can indeed provide information that can be used to pre-fetch useful Internet content for mobile users. While it is expected that such an approach cannot fulfil the entirety of Internet content needs for a user, the work presented here provides evidence to the extent to which a mobile cache can be populated with relevant documents that the user could find of interest.

Further to this, this research is concerned with the potential of calendar entries to be used as sources for web query generation, independently of the entry brevity and without the direct involvement of the user. This is an essential step for the investigation of the aforementioned aims, given that an appropriately formulated web query would have a better chance of retrieving relevant documents and thus populate the mobile cache with more appropriate results.

Finally, this paper shows that it is indeed possible for a predictive pre-caching system to efficiently adjust itself to the preferences and circumstances of the user as an individual, in order to obtain optimal retrieval performance.

While not directly related to the main aims of this research, we report further results and findings which concern the usability and interaction patterns within electronic calendars, the document reading behaviour on mobile devices and the suitability of implicit interest indicators for information retrieval on mobile devices.

2. MOTIVATION AND CURRENT PRE-CACHING TECHNIQUES

One of the basic issues in the problem of effectively pre-caching internet content needs, whether on a large scale, such as in servers or proxies, or on a personal level, is the determination of exactly which documents should be pre-cached. Taking this decision at a level that allows for maximum personalization, involves the observation of a user's behaviors, in order to create suitable models that would encompass these behaviors and allow accurate predictions to be made.

One of the earliest attempts at the automatic prediction and retrieval of internet content was described by Balabanovic et al. in 1995 [1]. The system described there forms a model of each user's preferences and continuously adapts itself to reflect the user's opinions of the content that is prefetched. The user is presented with a collection of hyperlinks to documents that the system has identified as potentially interesting. There is an option for the user to explicitly rate each link (from +5 to -5), therefore providing the system with simple relevance feedback. Even though the scheme employed by the researchers is a relatively straightforward approach, they succeeded in proving that there are significant gains that can be made through the personal profiling of users.

Wang and Crowcroft [2] discuss the some tradeoffs between pre-fetching and the improvement of latency in the WWW. Also, they present an implementation of a deterministic pre-fetching approach, called Coolist. Their system is layered between the client and the proxy server and organises websites in folders. These folders can then be assigned three methods of pre-fetching. Batch pre-fetching is the first method, where a site is scheduled for downloading at a given date or time. Another method of pre-fetching is described by the term "start-up" and means that a site will be pre-fetched when Coolist is invoked. Finally, their third proposed method is pipeline pre-fetching, where sites are grouped

for pre-fetching. When the first page in a group is requested, the next one will be automatically pre-fetched, regardless of the fact that a user may have not requested it.

Another discussion of the advantages of pre-fetching was carried out by Cunha and Jaccud [3], who proposed two algorithms for the prediction of the user's next action while browsing the web. Their first algorithm, using Random Walk approximation, projects the long-term interaction trend, while a second algorithm focuses on the short term trends. Using a model described by Thiebaut in 1989 [4], which relates the accumulated number of cache misses to a program's random walk range, the researchers show that it can be successfully applied to characterise users' strategies, under the hypothesis that these relate to an infinite browser's cache. This model is mathematically described as follows:

$$N(r) = Ar^{1/\theta}, r \gg 1, \theta \geq 1$$

In this equation, r is the number of references, $N(r)$ is the accumulated number of misses, θ sets the curve growth pace, and A is a constant. A second method is described within the same report, which uses an algorithm of two phases: Firstly, a *preparation* phase computes the first order difference of the envelope of the user's profile curve, displaced by a factor of 0.5 (for ease of detecting behaviour changes). Secondly, the *prediction* phase determines how conservative the user was in the last t accesses. Also, a determination of how much history is made, based on that count, in order to compute the desired set of coefficients that minimise the short-time prediction error, around a vicinity of size n , for a sample at virtual time r . A routine, based on Durbin's method to calculate the linear prediction coefficients is then called, and lastly, the predicted value is computed as a linear combination of the past $NCOEF$ terms. The authors show that both user models manage to achieve a degree of accuracy around 85%, which can be applied in conjunction with pre-fetching techniques.

In his technical report, Palpanas [5], investigated the feasibility of using a model based on the partial-match prediction algorithm, for pre-fetching documents from the web. In his model, a pre-caching agent acts as an intermediary between the client and the server(s) that a user is connected to in a session. Having taken into consideration the special characteristics of the Web and after tailoring the algorithm to accommodate those, the author concludes that his proposed scheme's implementation is feasible and that it would be assistive to users who "consistently follow regular access patterns, when searching for information". This conclusion is reached through simulations, run on the access log files of the web server of the department of computer science, at the University of Toronto.

Jiang and Kleinrock [Jian98a] presented in 1998 a system in which pre-fetching is decided by the client, based on usage statistics about embedded HREF tag attributes. In their work, the client monitors its available bandwidth continuously and pre-fetches web content, choosing however not to pre-fetch images, in order to save bandwidth. An algorithm to decide which pages should be pre-fetched is used, based on the client's access history combined with the server's access histories for each file they hold. Further filtering on the decision process is placed by placing an upper bound on the pre-fetch threshold, which is a function of

the system load, capacity and cost of a time unit and a system resource unit. This two-tier decision process allows the system to maximise the performance gaining that can be achieved through pre-fetching.

Further application of Jiang and Kleinrock's work is found in another paper that investigates pre-fetching for mobile users [Jian98b]. Interestingly, in this paper, the authors extend their prediction algorithm to achieve higher hits, by assigning users to a category (such as those interested in database research), amongst other things. The second component of their scheme is a server threshold model, which judges whether a page should be pre-fetched based on:

- The amount of time that may be saved by pre-fetching a file that may be needed
- The amount of bandwidth that will be wasted if the file is not used;
- The impact of the pre-fetch request on other users whose normal requests may be delayed by the pre-fetch request.

The latter is a necessary consideration in order to ensure that the overall system performance can be improved by pre-fetching the file. The authors proceed to conclude that their approach is well suited to mobile users, who may need to switch between different network connection methods (modem, broadband, satellite, wireless). This is because the separate server threshold module allows the adaptability of the prediction algorithm, ensuring the best possible performance under each network condition.

A simpler implementation than the one by Ziang and Kleinrock was proposed with the WCOL system, by Chinen and Yamaguchi in 1997 [8]. Their system is a research prototype that pre-fetches embedded hyperlinks top-to-bottom without regard to likelihood of use. Embedded images of pre-fetched pages are also pre-fetched. Bandwidth waste can be capped by configuring WCOL to pre-fetch no more than a certain number of hyperlinks, and no more than a certain number of images embedded within pre-fetched hyperlinks.

In 1999, Dan Duchamp [9] presented his own work on pre-fetching hyperlinks, based on a predictive algorithm at a client level, which is however able to communicate usage statistics from the server. Because the client is unable to form an objective view of the usage for a given web page, unless that page is visited often by the user, it passes on to the server the current usage statistics it has obtained, but also demands the aggregated statistics for that page, as held by the server. The performance results obtained by an implementation of the aforementioned idea were strongly encouraging. For example, of all the pre-fetched pages, a figure of 62.5% was eventually used. An improvement in latency of the order of 52.3% was observed, while notes for the consideration of network overhead due to the usage reports are being addressed. Further concerns regarding the size of the modifications necessary to the browser (Mozilla) and the execution time overhead due to these, are eased, since these do not appear to be significant.

Continuing on the theme of document pre-fetching on a personalised level, more related work was carried out by Fan et al. [10] in 1999. They propose pre-fetching at a proxy level, under the argument that because proxies can collect access histories for limited numbers users, this would present significant advantages over server-level pre-caching, since servers would be able to only collect access histories for the entire WWW population. An investigation of prediction by partial match algorithms follows in their work, followed by a simulation which shows an improvement in latency between low-bandwidth clients and proxies, of the order of ~23%

Pitkow and Pirolli [11] explore predictive modelling techniques that attempt to reduce model complexity while retaining predictive accuracy. The techniques merge two methods: a web-mining method that extracts significant surfing patterns by the identification of longest repeating subsequences (LRS) and a pattern-matching method that embodies the principle of weighted specificity. Their work is largely motivated by previous studies by Schechter, Krishnan, and Smith [12], who utilized path and point profiles generated from the analysis of Web server logs to predict HTTP requests. Also, much reference is made to the work by Padmanabhan and Mogul [13], who describe the efficiency of Markov Models for pre-fetching. The authors use the definition of Longest Repeating Subsequence by Crow and Smith [14], which contains the following terms:

- Subsequence means a set of consecutive items
- Repeated means the item occurs more than some threshold T, where T typically equals 1
- Longest means that although a subsequence may be part of another repeated subsequence, there is at least one occurrence of this subsequence where this is the longest repeating.

Two models are proposed, firstly a hybrid LRS model which extracts LRS patterns from a training set and uses them to estimate a 1st order Markov Model. This model is compared against a 1st order Markov model estimated from all the paths in the training data set. The second hybrid model proposed is one that decomposes the extracted LRS subsequences into all possible n-grams of various lengths. This is called the All Kth Order LRS model, as all orders of k are able to make predictions. This model is compared against an All-Kth-Order Markov model derived from all the possible subsequences, decomposed into varying length n-grams. Further simplification is added to their web surfing model, by assuming that surfing paths have an average branching factor b. Surfers may start in b places and from each page, they move on to one of b pages on average. By assuming that surfing paths of length S can be divided into S/k subpath partitions ($0 < k \leq S$), the complexity cost C(k), in terms of the number of patterns as a function of k can be described as

$$C(S) = \sum_{i=1}^S (S / i) b^i$$

Through analysis of the applied methodology, the authors showed that in the case of modeling paths under the first hybrid model, the reduced LRS model was able to match the performance accuracy of the 1st order Markov model while reducing the complexity by nearly a third. They also then showed that overall hit rates could be raised by including the principle of specificity, with the All-Kth-Order LRS model almost equaling the performance of the All-Kth-order Markov model while reducing the complexity by over an order of magnitude. Finally, within their findings, it was further shown that increasing the prediction set has a dramatic impact on predictive power, with the predictive power of each method nearly doubling by increasing the set size to four elements.

Swaminathan et al. presented in 2000 [15] a study of web pre-fetching which is based on the characterisation of the web client alone, without depending on server or proxy side algorithms.

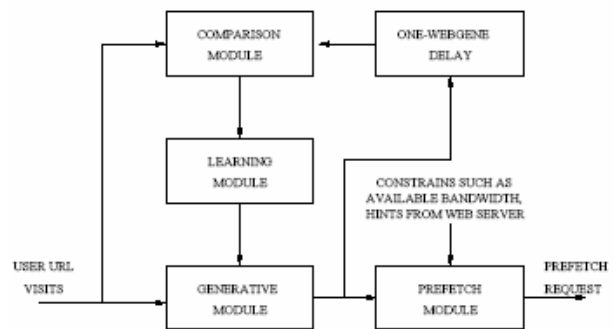


Figure 1: Operational principles diagram (Swaminathan et al.)

The above figure shows the principle of the system as described by the authors. The input stream consists of symbols representing the URLs. The learning module learns the trend in visit counts associated with the past n URLs and the order of URLs visited and it is implemented using genetic algorithms. The comparison module compares the predicted and the actual streams and provides feed back of the error in prediction to the learning module. The generative module attempts to predict the next k URLs that might be visited for possible pre-fetching. Finally, the pre-fetching module decides which URLs to actually pre-fetch based on constrains such as available bandwidth, recommendation from the server and the state of the web client. In their research, the authors highlight the problem of dynamically generated Internet content, which renders pre-fetching approaches useless. Indeed, given the trend to deliver highly customisable content to users and the generic structure that is “populated” by dynamic articles, which now forms the basis of many major sites, the validity of the pre-fetched content becomes an important issue. An interesting point in this research was that the authors manage to prove, through simulation on actual client traces, that their proposed pre-fetching technique allows the maintenance of client

cache hit ratio of around 13% on average, even when all the visited URLs are dynamic.

Interesting research on proxy cache was also presented by Foygel and Strelow in 2000 [16]. The authors propose a system of hierarchical proxy caches, where their algorithm observes requests to a cache and its ancestors, before initiating pre-fetching for the predicted future requests. This would only happen if the pre-fetching action is deemed likely to reduce the overall latency experienced by the cache's clients. Their algorithm is based on the continuous evaluation of the usefulness of each document in the cache, but also of documents that are not in the cache, but are likely to be needed in future requests. The set of documents (fetched or un-fetched) which has the greatest esteemed value is kept in the caches. In their conclusions, the authors argue that a hierarchical cache network structure is the ideal foundation on which pre-fetching can yield significant performance gains. Again, however, they highlight the concern over increased network utilisation, although they argue that it is often the case that because traffic is added to under-utilised networks, the performance gains can be obtained without significant cost.

Brian Davidson [17] presented an article in 2002, which relates to predicting web actions from HTML content. In his work, he compared the simplistic approaches so far taken for pre-fetching based on HTML content, with an information retrieval-based one that ranks the list of links using a measure of textual similarity to the set of pages recently accessed by the user. These simple approaches vary, but examples are namely pre-fetching all hyperlinks in a page, or pre-fetching all hyperlinks in a serial manner, as time allows.

The algorithm used for measuring the similarity between two text documents (D_1, D_2) is

$$\text{TextSim}(D_1, D_2) = \sum_{\text{all}_w} TF(w, D_1) * TF(w, D_2)$$

($TF(w, D_1)$ = the number of times term w appears in D_1)

Having compared text-similarity-based ranking methods to simple original link ordering and a baseline random ordering, the author found that similarity-based rankings performed 29% better than random link selection for prediction, and 40% better than no pre-fetching in a system with an infinite cache.

The ideas of Davison are implemented in Mozilla, an open-source web browser and a technical report is made in 2003 by Zhang et al. [18], who incorporated the aforementioned content-based prediction algorithm with the history-based prediction described in another work by Davison [19].

Further related research, however not immediately a document pre-caching technique in its own right, was presented by Cohen and Kaplan [20] in 2000. Their proposal is that in order to overcome potential problems in the validity of cached documents, and other problems that relate to the increased network utilisation, which in turn might cause clients to experience even more latency, one could pre-fetch (rather, pre-execute, one might add) the *means* of getting a document, rather than the document itself. This is because of the observation that the actual steps required for the setup of a connection, are relatively costly in terms of time. A suggestion is made that this *pre-transfer prefetching* could be accomplished by

- Pre-resolving, which means that the browser or a proxy could perform a DNS lookup before a request to a server is issued, therefore eliminating the DNS query time from user-perceived latency
- Pre-connecting, where the browser or a proxy establishes a TCP connection to a server, prior to a user's request. This should address the problem of connection establishment time, which is significant compared to HTTP request response times.
- Pre-warming, or, in other terms, sending a dummy HTTP HEAD request prior to the actual request, in order to address the problem of start-of-session latency at the server, which tends to be larger for first-time requests than follow-up requests (the server is referred to as being "cold" or "hot", once a request has already been issued).

The tests conducted in this research show a significant decrease in average latency times, proving that pre-fetching the means for getting a document is a useful technique that can be applied to the problem of reducing overall latency.

3. LIMITATIONS OF CURRENT RESEARCH

The techniques presented in these research papers describe the process of pre-caching documents in a dynamic manner, whilst however presuming that the user has a currently active connection and is using (surfing) the Internet. Although all of the concepts are largely relevant and indeed of great interest, they suffer from the disadvantage that network connectivity is a pre-requisite and is essential to their operation.

In mobile devices such as PDAs and Smartphones, network connectivity is not a transparent service which is normally available under typical operating circumstances, such as on modern broadband-connected desktop computers. Indeed, network connectivity depends heavily on the location of the user and the strength of wireless network signals, the interference of nearby radio-wave emitting devices with network antennae, the subscription to co-operating networks, the present load of the wireless network and the type of wireless network connectivity supported by the mobile device hardware. All of these factors are commonly present in everyday situations and make the likelihood of network unavailability rather high.

Further to this, most of the aforementioned pre-caching models make no, or very little, consideration of the bandwidth used to pre-fetch files which will never be used and assume a practically unrestricted local (or proxy) cache. With mobile devices, one has to consider that the available bandwidth is generally very limited, as well as extremely costly, as the charging mechanism is per unit of data (typically KB), rather than time, when using connections that provide acceptable surfing speeds (GPRS, 3G). Additionally, the devices themselves offer very little memory capacities, thus placing significant constraints on the size of caches that can be created for web surfing. It can therefore be generally concluded that current predictive pre-fetching models cannot be directly applied

Given the high probability of network unavailability in the daily environment, internet content pre-caching seems a logical solution to partially solving the problem of mobile information access. However, pre-caching, as described in aforementioned research, is a technique that is largely inapplicable (or even unacceptable), due to the significant cost and the charging mechanism for accessing the Internet over a wireless connection. In fact, given the charging per Kilobyte that most service providers adopt for GPRS and 3G, it would seem obvious why a user might want to initiate data transfer from the Internet as only they deem necessary. Wi-Fi networks on the other hand are a lot less expensive but are very limited in range, thus preventing true mobility.

Another type of solution to pre-caching Internet content is common to several types of mobile devices are not equipped with wireless connection (actually even to devices that have integrated wireless capabilities), but have software for browsing the web already installed. Users of devices such as the ones we described, tend to resort to commercial programs such as AvantGo, that provide a means of pre-caching selected or interesting pages based on declared user preferences and storing them for off-line browsing. However, such services offer static, rather than dynamic personalization, and depend heavily on explicit user instruction on the (limited) type of content that should be pre-cached.

4. A PREDICTIVE PRE-CACHING SYSTEM FOR MOBILE DEVICES

Information access relates strongly to the individual user's interests, which rise from general, always valid (permanent) preferences (e.g. aviation news will always interest an aviation enthusiast, regardless of their profession). Information access also is strongly dependent on preferences which can vary, according to the task the users are, or anticipate to be engaged in. While the knowledge of both contexts under which information access might be desirable are necessary for pre-caching useful information, in our research, we focus more on the automatic prediction and comprehension of the activities a user might be engaged in, in the near future.

Thus we hypothesized that it would be possible for a personal pre-caching algorithm to examine a user's electronic calendar as a source of information that it could use, in order to make dynamic informed predictions about the user's future tasks and provide useful content to the user. While it may be unrealistic to expect a calendar-based pre-fetching algorithm to describe all of the user's preferences and content needs or desires in a perfect manner, we expect that content that is relevant to the user's daily interests can be picked up. Further more, we were interested in investigating whether permanent user preferences can be captured in order to tailor the quantity and quality of the content better to suit the user.

5. THE PRE-CACHING SYSTEM

5.1 Overview

To explain the principles of operation behind the system we propose, it might be useful to present an example scenario. Let us consider the following example: Each entry in the user's calendar should contain at least one keyword, which will help us understand the nature of the activity denoted by the entry. If a calendar entry contains a keyword such as the name of a city (e.g.

Edinburgh), of which the user is not a resident, it can be assumed that the user will be present at that location for some purpose which is possibly described elsewhere in the entry. Typical information about cities could then be downloaded through web searches such as "Edinburgh map", "Edinburgh accommodation", "Edinburgh Museums" etc. Also, further and more specialised searches, formed through combination with other information retrieved from the calendar entry, can be conducted, e.g. "Edinburgh Holiday Inn" or "Edinburgh HCII2004 conference".

The predictive system can then be enhanced over time through use of implicit or explicit feedback to learn and remember a users' preference for different categories of information, e.g. more interested in transport links than hotels. Such a predictive system should be able to obtain information from the user directly and indirectly. The majority of information should be obtained through indirect means, in order to minimise interference with the user's other activities. However, the system should maintain its ability to directly interact with the user, in order to resolve any possible uncertainties.

In previous work [21], we described the overview principles for our pre-caching system, whose software comprises of two core modules, each of which encompasses a number of cooperating agents. The first module resides on a desktop computer and works in order to extract entries from the user's electronic calendar. It then uses the desktop's connection to pre-cache documents. The second module is responsible for presenting the pre-cached content on the user's mobile device and therefore resides therein. The second module is also responsible for evaluating the user's interactions with the pre-cached content and passes back this information to the desktop module, which then uses it to make more informed guesses at what it needs to pre-cache.

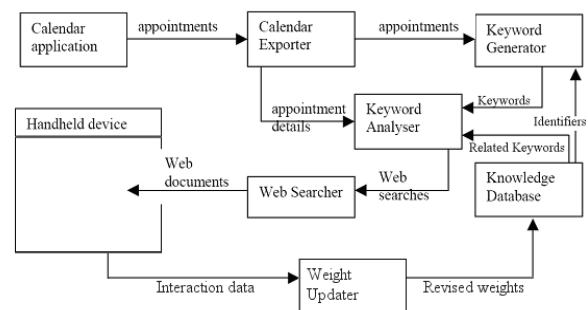


Figure 1. Overview model of the system components

5.2 Identifying keywords

One of the most important problems that the system needs to solve is the identification of keywords within the calendar entries, which can be used to initiate the process of web query formulation. To achieve this, a database of known keywords (identifiers) could be kept, against which the system would compare the content of the calendar entries.

In order to train the system with the ability to recognise potential keywords, it was decided that an analysis of the contents of a sample of real world calendars would be necessary. The scope of this analysis would be to determine firstly which suitable words and of what types, would be frequently encountered in a calendar.

Subsequently, in order to confirm previous early research on calendars, an attempt would be made to identify calendar entry categories, so the list of keywords for these could be supplemented with other words that belonged to similar contexts. For example, if the analysis was to show that placenames (“Glasgow”, “Edinburgh”) were common occurrences, then an appropriately extensive list of placenames would need to be compiled. These keywords in essence are *identifiers* for the categories they represent.

Although early research by Kincaid and Dupont [22] had highlighted some of the categories of entries commonly encountered in calendars, it was felt necessary to re-investigate this matter, firstly due to the age of the preceding research (which was not based entirely on electronic calendars), and secondly to tune the system performance for use by individuals in an academic context. Because the subjects of our final experiments were expected to be from an academic environment, it appeared reasonable to attempt to gear the system towards the particularities of the academic community. The system could have similarly been geared towards other professional areas or could be tailored towards a general population. This would require the selection of appropriate test subjects and since the comparison of performance on different target groups is not part of the scope of this research, the academic sample and subjects were judged to be appropriate and adequate.

From our research in the categorization of entries, it immediately became clear that the calendar entries tend to fall within specific categories. The categorisation of the entries was done manually, with guidance from the original entry authors, where ambiguity made it necessary. This categorisation comes as a confirmation of the findings of previous research, and especially those of Kincaid and Dupont, who discovered that users tend to use their calendars mainly to keep a record of meetings, appointments, events, travel, reminders, notes and as “to do” lists. This, in turn, suggests that there is a consensus to the items that form appropriate calendar entries and that there is a common mental representation model for the organization of these entries amongst humans. There is a slight variation in the categorisation of entries that was made with these new findings, although this can be considered natural, due to the particularities of the academic environment. However, the categories with the highest frequency are the same as those in previous research. The table below shows the categories and entry frequencies as derived from this research’s sample.

Category	Frequency
Meeting (Group)	53
Meeting (another person)	25
Reminder	18
Travel	13
Social event	13
Work task	10
Class (to attend)	10
General Task (to-do)	7
Miscellaneous	7
Birthday	5

Table 1: Calendar entry categories and their frequencies

It is important to stress here that these categories reflect the opinions of the calendar users, who are highly familiar with the context of their entries. One can observe that there is some overlap between the calendar entries, for example, Birthday could be considered a subset of Social. However, where such overlap is maintained, it is because there was a strong indication from the users that such a low-level category is significant and should exist separately from its high-level parent.

The database of category identifiers which would be compiled, would not exclusively contain words, but also short keyphrases, such as “travel to”. The reason for this choice was that it would be almost impossible to compile a list of every possible associated identifier for every category. Therefore by including keyphrases, we could infer that an unknown word which would follow these, might actually be a good identifier candidate. For example, “trip to Garnethill¹” is a sentence where a human can immediately identify Garnethill as the name of a place, even though they might never have heard of it. It is also very unlikely that the name of Garnethill would come up in any general placename list. It was desired that same functionality for inferred knowledge should be implemented for the system as well.

It might be argued that using pre-compiled lists and rules as a basis might not be an optimal solution. However, the process of recognising keywords and inferring their meaning based on pre-compiled rules and lists is well-established practice in the field of Information Extraction. This practice is one of the two approaches generally available for solving the problem of machine text analysis and understanding, the other one being automatically trained systems that do not rely on hand-crafted rules and databases. Given the recommendations of Appelt and Israel [23] on the choice of approach and based on the good availability of resource lists (e.g. name and place lists) and the lack of extensive training data (calendar entries), the manual approach was elected.

5.3 Formulating queries based on keywords

With the appropriate keywords identified by the Keyword Generator module, the next step the system should take is to generate appropriate queries for these keywords. Sometimes a keyword on its own might be a good candidate for a query. However, in order to obtain information that is closer to the needs or preferences of the user, the keyword will have to be combined with other keywords to form longer queries. Those additional keywords can be obtained from the calendar entry itself, or could be retrieved from a separate database of additional keywords which are known to be typically included in queries that are based around the original keyword.

To be able to adapt the system to the user’s preferences and needs, it becomes apparent that all additional keywords will need to be weighted in order of importance to the user. Because some of the additional keywords that will initially be provided might be irrelevant to the user’s context, the system should be able to either negatively weigh these, or increase the weight of additional keywords for which the user was presented queries that retrieved “good” documents, while leaving other weights intact. A combination of positive and negative weighting can also be used.

¹ Garnethill, a region in the Glasgow City Centre area (UK).

One approach to the problem of constructing an additional keyword database would be to create a list of potential associations for each keyword contained in the identifier database. In this manner, the level of relevance between identifiers and additional keywords would be high, although this would come at a considerable cost. The costs of this approach would firstly be the large degree of duplication of data, as an additional keyword might be relevant to a multitude of identifiers. Secondly the construction of appropriate additional keyword lists for every single keyword would have to be done manually and there is no guarantee that every single identifier would have an additional keyword to reflect all context scenarios. Finally, because of the specificity of the additional keyword associations, a sudden or temporary change of context for the user might be completely ignored or take a long time to adapt to, which is of course undesirable.

An alternative approach would be to cluster the additional keywords into smaller databases, which will reflect the association of each keyword category, rather than each keyword individually, with additional keywords. The term of *clustering* is borrowed from the Information Retrieval field, where it is defined as follows:

*'...We define the organisation as the grouping together of items (e.g. documents, representations of documents) which are then handled as a unit and lose, to that extent, their individual identities. In other words, classification of a document into a classification slot, to all intents and purposes identifies the document with that slot. Thereafter, it and other documents in the slot are treated as identical until they are examined individually. It would appear, therefore, that documents are grouped because they are in some sense related to each other; but more basically, they are grouped because they are likely to be wanted together, and logical relationship is the means of measuring this likelihood...'*²

This is a more natural step to take, considering that calendar entries are already clustered into categories by nature. Therefore web queries will be made by combining the original keyword (category identifier) with either all or the top n additional keywords that are associated with the category. In fact, as the system becomes increasingly accustomed to the preferences the user, the user-defined n -sized window of additional keywords ranks should reflect the true preferences of the user by returning a smaller amount of keywords within the same window.

The additional keywords can initially be set to have the same score, which would indicate an equal opportunity for each additional keyword to rise up in the ranks and therefore reflect a user's preference, independently of that preference's commonality. Alternatively, additional keywords can all be given different initial scores which would reflect the general preferences and assumptions that could be made for an average user. This would impede the training process for users that have very particular requirements, although for the majority of the users this

method should provide adequately promising performance even from the early phases of use. We employed the second option in our implementation.

In order to obtain additional keywords for each category, three different methods were used in conjunction:

- Interviews where people were asked to give details of typical searches they might have performed for calendar entries of each category (during the calendar entry sample collection)
- Calendar entry samples were given to independent subjects who were asked to produce as many web queries as they could for each entry sample
- Google's keyword suggestion tool³, where category identifiers were entered and a list of potential associated keywords was returned.

In our implementation, the keyword analyser module forms the following two types of web queries:

- Original keyword (or keywords, e.g. name+surname) only
- Original keyword + additional keyword

The system can currently be set-up to fetch the top N rated queries for each user, according to their rank value, although for the main experiment, the amount of generated queries was limited to the top 5 rated items. When two or more queries happen to be ranked equally, they are both included in the generated query list.

5.4 Pre-caching documents

Having described the methodology for formulating queries, the next logical step in the sequence of operation events for the system is to submit these queries into one or more search engines on the Internet, which in turn should be able to retrieve links to relevant documents.

Those links should be followed in order to retrieve the documents. From within the documents, further links can be followed to documents within the same web site, or to documents that are external to the site. If those external documents were fetched and analysed, the system would encounter yet more links, which could also be retrieved. The retrieval process might be likened to a multi-tier tree, where a document from the original query can be considered a root, further linked from it are second-level nodes and links to further documents from these form edges that lead to even lower-level layers of nodes.

² HAYES, R.M., 'Mathematical models in information retrieval', in Natural Language and the Computer (Edited by P.L. Garvin), McGraw-Hill, New York, 287 (1963).

³ Google Adwords keyword suggestion tool:

<https://adwords.google.com/select/main?cmd=KeywordSandbox>

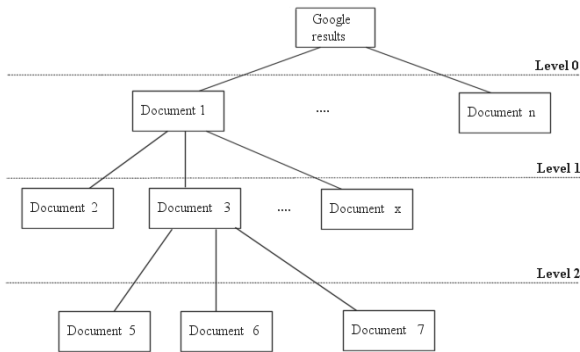


Figure 2: The multi-tier retrieval tree

It becomes apparent that the system could be locked into a fetching mode, which will exponentially increase the number of documents retrieved, without guarantees that all of the lower-level nodes will be relevant to the original query. In fact, it is logical to expect that the degree of relevance will reduce with the increase of the depth of the retrieval tree. Considering Jansen's [24] observation that only 50% of the original web documents retrieved from web queries are classed as "interesting" by users, the lack of any guarantee of relativity of all the supplementary retrieved documents is apparent. For this reason a control mechanism should exist, which will either limit the amount of levels the retrieval tree can have, or be able to intelligently predict whether a linked document might be worth pre-fetching. Based on the pre-fetching methods researched earlier for desktop computers and proxy caches, it becomes obvious that such decisions have to be based either on data from server request statistics or an analysis of the target document, in which case it needs to be fetched so it can be analysed. Such an implementation technique as the latter might be considered an interesting exploration for the adjustment of performance levels, however it is beyond the scope of this investigation. Therefore a logical solution for this problem under these circumstances would be to implement a mechanism that would limit the depth of the retrieval tree levels for each document.

Further considerations would have to include the type of Internet content that the system should be able to store. Further from HTML documents which contain formatted text, other elements such as images, .pdf or .doc files, audio and video, might be desirable to have. All of these however place a considerable stretch of the memory limitations of current handheld devices. For this purpose, we carried out interviews with users in order to discover exactly what kind of content should be prioritised for pre-fetching. The results of the interviews showed an overwhelming preference to text (HTML, Word, PDF), while other media such as images, audio and video were considered largely irrelevant, unless a user was specifically looking for such types.

Of the search engines available today on the Internet, the most successful is the Google engine⁴. This engine was chosen for the submission of the formulated web queries. An observation by

Jansen and Spink [25] indicated that the majority of users (80%) only view between ten and twenty results for each query, i.e. no more than one or two pages of results. Because the Google search engine can retrieve thousands of relevant results, a choice was made to limit the returned results to the top ten, as ranked by Google. It was felt that the choice to restrict the results to the top ten was justified, bearing in mind the memory constraints that are present on handheld devices. Furthermore, because Google is widely considered to be the best Internet search engine currently available, the choice was made not to submit the queries to other search engines. The comparison between the performance of Google and other search engines was not of interest for the purposes of this research and beyond this, such an attempt would greatly increase the amount of retrieved documents without any expected significant increase in the amount of retrieved relevant documents. This would place a significant and unnecessary burden on the subjects of the main experiment. Although collective filtering could be applied to perhaps include only the top K results from a number of search engines, it should be kept in mind that the optimisation of the cache, while desirable, was beyond the purposes of this research. Thus no services other than Google were employed in this instance.

The Google result page is trimmed from all unnecessary HTML elements in order to remove graphics, advertisement and unnecessary links. Subdirectories are created to store all the documents of the desired tree depth levels. The retrieval procedure then works for each document of each tree level, including the Google result page, in the following manner: Firstly, the document is parsed and searched for links. Once a link is encountered, the linked document is retrieved and stored in the appropriate subdirectory. The link in the document under analysis is changed to reflect the local relative URL of the newly-fetched document. The process continues until no further links can be found in the document, in which case, the system moves on to the next document of the same depth level. In this manner, one can envisage a horizontal breadth-first traversal of the document traversal tree, in order to generate the lower levels of pre-fetched document. The entire process ends when the user-specified retrieval depth has been reached.

Once the system is finished retrieving the top ten documents for each Google page, further transformation of the Google page is made. A second-level parsing is performed in order to separate the document titles, summaries and URLs contained therein, and store them in an XML structure. This XML structure will later be passed to the handheld device, along with the relevant documents, and will be used to display the retrieval results to the user.

5.5 Mobile Component

5.5.1 Presenting Results

Once the documents have been retrieved and sent to the handheld device, a separate software component therein should be responsible firstly for displaying the documents to the user, and secondly for observing the interactions of the user with these documents.

It is clear that for the first task, the design needs to consider the physical characteristics of the handheld device and especially the constraints placed by the dimensions of the device screen. Taking in mind Nielsen's observations that users tend to dislike long pages which require lots of scrolling [26], an implementation of

⁴ <http://www.google.com>

the results browser should be considerate of this natural tendency and contain facilities that will allow the users to minimise the scrolling needed. A collapsible tree-structured list of calendar entries, identified keywords, searches and retrieved document titles/summaries is potentially a good way of addressing the scrolling problem. Unfortunately, due to the nature of web pages, scrolling to view their content is inevitable. However the quality of the browsing is beyond the system's control and will be fully dependent on the device's integrated web browser. As this issue is beyond the scope of this research, no attempt to write a dedicated web browser was made.

5.5.2 Processing relevance feedback

Observing the interactions of the users poses several questions that need to be answered. Firstly, which actions of a user do reflect interest and therefore should be monitored? Secondly, once such actions are identified, do they all indicate the same amounts of interest or should their importance be weighted with different measures? Based on previous related research, an instinctive negative answer to the last question is probably the right one. Indeed, as mentioned previously in chapter 2, it has been shown that not all kinds of interaction can provide dependable implicit information on relevance.

On the handheld device, a log is kept of the user interactions with the content. Based on this information, an attempt should be made to judge the relevance of a given keyword from the knowledge base to the user's context and determine which of these are likely to be wanted as part of a query in the future. Given the collapsible presentation structure, a log can be kept for:

- The viewing of the document index for a given search
- The viewing of the summary of a given document
- The viewing of the document
- The amount of time spent on a document that has been opened
- Any explicit feedback rating that a user might provide for the document.

The incorporation of further heuristics, such as pointer movement, highlighting of text, bookmarking and scrolling in the document, would also be desirable. However, given the implementation was on a PocketPC platform, programming Pocket Internet Explorer to trap such behaviours was not feasible. In addition, some of these heuristics (e.g. scrolling) would not be reliably applicable, given the lack of previous research on small screen devices for such measures.

Upon loading, the handheld component examines the XML structure passed to it and the XML-formatted Google pages, in order to load the necessary details for presentation to the user. The pre-fetching activity details are presented using a collapsible tree structure list, which gives details of the appointments, keywords identified, web queries formed and document titles of each search (see figure 3).

When a document title is highlighted, the user is given the option via two buttons to either launch a descriptive summary of the

document, or open the document with Pocket Internet Explorer. The system monitors the collapse of the document list for a web query, the viewing of a summary, the launch of Pocket Internet Explorer and the duration for which it is active, i.e. in the foreground. It is assumed that the user will be reading the document for that time. Also, when Pocket Internet Explorer exits and the user returns to the handheld software that is running in the background, an option is given to explicitly rate the quality of the document just read on a scale of 1-5 (figure 4).

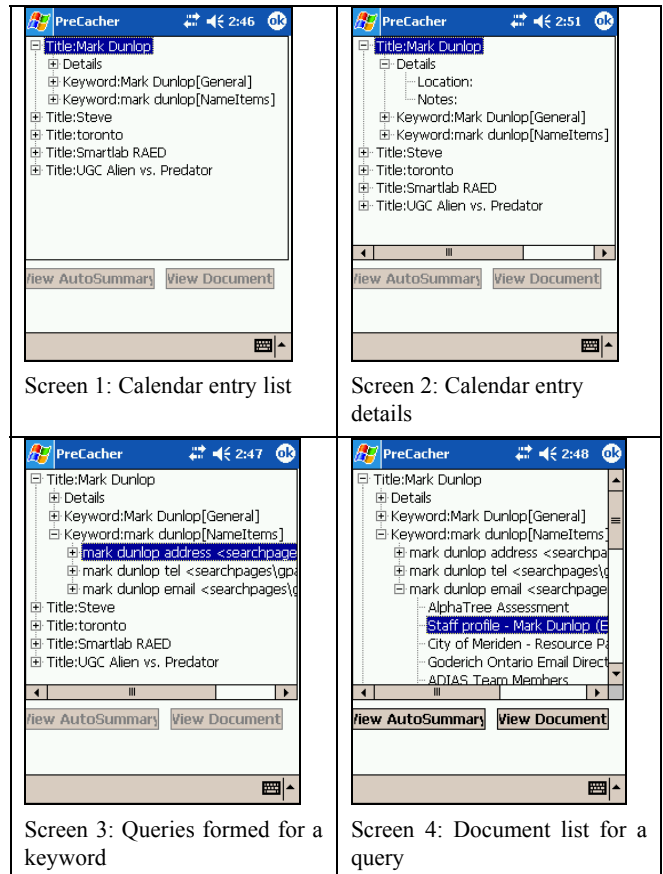


Figure 3: The mobile User Interface, showing the collapsible tree list. Each keyword under a given calendar entry (title), can be expanded to show the queries formed for it. For each query, a list of retrieved documents is provided.

To implement the document summary function, the short document narrative that Google provides directly under the document title in its results page is used. This is displayed as a pop-up dialog box to the user, upon request, therefore imposing an interaction cost on its viewing (figure 4).

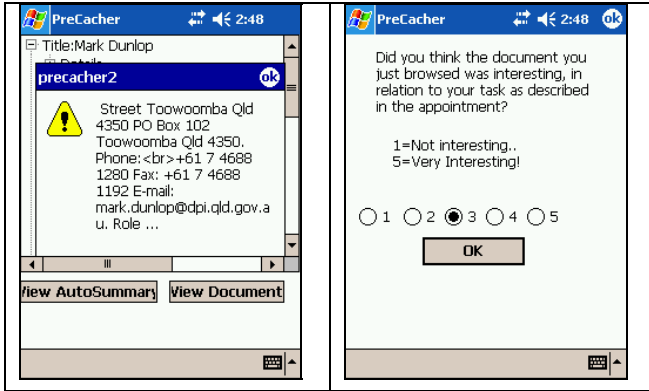


Figure 4: The auto-summary function and the explicit relevance feedback screens.

5.6 Processing relevance feedback

For the heuristics mentioned above, we had to devise appropriate weights and a method to combine their values to form a score for each keyword. We define the importance $I(k)$ of a keyword k to be

$$I(k) = w_{E(k)} + \sum D(i)$$

where w is the weight associated with a viewed document index for the keyword $E(k) \in \{0,1\}$, and $D(i)$ is the importance of each document i that has been retrieved for keyword k . Further more, we defined $D(i)$ as follows:

$$D(i) = \alpha O(i) + \beta S(i) + \gamma F(i) + \epsilon T(i)$$

where $O(i)$ indicates the viewing of document i , $S(i)$ indicates the viewing of its summary, $F(i)$ is the explicit feedback given by a user to the document, $T(i)$ is the time spent reading the document and $\alpha, \beta, \gamma, \epsilon$ are the respective weights for each of these measures.

The weights w, α, β, γ and ϵ take positive or negative values. We wanted a mechanism that would promote the appearance of preferred keywords in the queries. However, the negative marking for undesirable keywords would not only allow the improved promotion of desirable keywords, but also, should one of the latter become undesirable, due to perhaps the change of context of the user, it would not take too long for it to start disappearing from the queries.

We experimented with several weights for our system and we arrived to the conclusion that it is not only important to consider the relationship between the weights, but also the bias towards positive or negative marking. Current research [8] shows that users will view only two or three (on average) documents per web query and the vast majority will visit at most 2 pages or results (approximately 20 results in all). The same research shows that an estimate of 50% of documents viewed from these results are expected to be relevant to the query. Therefore we decided that a bias of approximately 1:7 in favor of positive marking was reasonable. Because our queries will not generate more than 10 documents each, this means that two documents with a positive

overall rating will indicate a successful and relevant query was made. The weights that were used were as shown in table 2.

There was some confusion as to the weights ϵ which should be used for association with the reading time interest indicator. As mentioned in chapter 2, previous research highlights a possible correlation between average reading time and relevance of a document. However, all previous research had been conducted on desktop computers, where screen readability issues were not as much of a problem. The initial inclination was to apply a tiered weighting system according to the reading time averages reported by Claypool [27]. The weights for each metric were chosen on an ad-hoc basis, mainly through consultation with the main experiment's control group subjects and their observed interactions with the system. From the figure below, it becomes immediately obvious that there is quite some overlap between the distributions of reading times for each explicit rating, something which makes it difficult to accurately infer document relevance from reading time alone, in most cases. After experimentation, (described in section 5), it was decided also that these average reading times were completely irrelevant to the handheld device environment. In fact, it was discovered that no distinct correlation could be made between reading time and document relevance on a handheld device, so the decision was made to abandon this metric completely.

Weight	Value
α	-0.03 (document not opened) +0.03*7 (document opened)
β	-0.021 (summary not viewed) +0.021 (summary viewed)
γ	-0.15 (feedback=1) -0.075 (feedback=2) 0.00 (feedback=3) +0.15*6 (feedback=4) +0.15*7 (feedback=5)
ϵ	Inconclusive

Table 2: Keyword score adjustment weights as used in the system implementation

5.7 Limitations of the system

We would have liked to be able to incorporate further heuristics, such as pointer movement, highlighting of text, bookmarking and scrolling in the document. However, given the implementation was on a pocketPC platform, programming pocket Internet Explorer to trap such behaviors was not feasible. In addition, some of these heuristics (e.g. scrolling) would not be reliably applicable, given the lack of previous research on small screen devices for such measures. Indeed, further on, we describe how we were forced to also exclude reading time from our heuristics. Further more, we decided not to pre-cache images or other media (pdf, word), purely for reasons of storage space constraints.

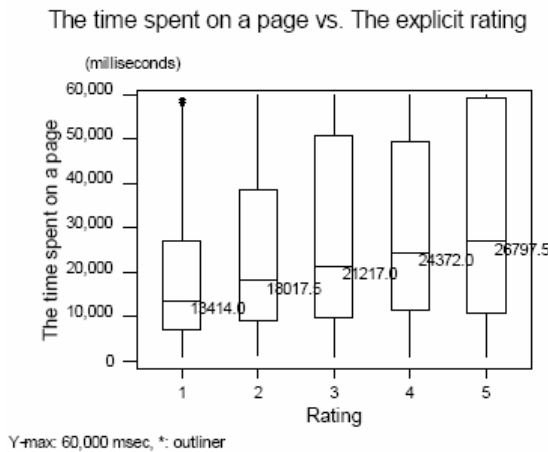


Figure 5: Claypool's findings on relevance and reading time correlation

6. EXPERIMENTATION

6.1 Experiment design

Having implemented a full system, as detailed in the previous section, it was time to test its performance and determine whether the hypotheses proposed by this thesis could actually be met. Ideally the system should be given to several users and they should be allowed to run it for a period of time which should be as long as possible. However, given the lack of volunteers that would be willing to run the experiment as part of their everyday routines and also given disproportionately large timescale the experiment would take, a decision was made to test the final system under supervised conditions which would simulate real world scenarios as closely as possible.

Two groups of users were given the same scenario with some details of their imaginary living location, job and a list of some names of people and how they would be related to them. Furthermore, over the duration of three weeks, the users would be given five tasks per week that form their hypothetical schedule of activities for that week. These activities were given in the form of a calendar entry that contained a title, location and notes for each one. Some activities did not contain items in the location or notes fields, as they were based on real-world entries that we had collected in previous studies. The users were also given clear instructions on the exact meaning of each entry, through the provision of accurate descriptions of the entries (see appendix 6).

The users were then allowed to freely navigate through the pre-cached content that was fetched for these hypothetical schedules, and try to locate content that they thought might be helpful to them. We would also ask the users to give an indication of whether they found the provided content for each activity useful.

It was decided that the users should not be told that their behaviours would be logged. Also, one of the groups would have their logs analyzed and we would attempt to provide them with content that was personalized on the basis of these logs. Again, the groups would not be made aware of this discrepancy until after the experiment had ended. The analysis of the logs was done automatically by the system, as described in section 3.9, for each

of the monitored subjects individually. Their respective profiles were maintained and updated at the end of each session, therefore influencing the retrieval process to personally match each of the monitored group's subjects.

Finally, another factor which was considered in the experiment, was that the physical storage limitations for the devices used led to a choice to pre-cache only HTML documents, and furthermore, these were restricted to the documents proposed by Google for each search. Effectively this meant that the retrieval tree was limited to just one level. For a first-level document that contains three hyperlinks, a two-order retrieval means a total of three documents retrieved. The amount of generated documents from just the one-level tree (table 16) was very large and this would only grow further with the implementation of additional levels. Therefore, in order to avoid overloading the user with documents and to overcome storage limitation problems, the choice was made to restrict the retrieval tree to just one level. Further to this, the focus was placed on the four most popular categories, according to the findings of the query test. Therefore the calendar sets given contained only entries of type Meeting, Travel and Social (including Birthday)

6.2 Initial experiment setup

An initial group of ten subjects volunteered to test the system before we proceeded with the actual experiment. All of the subjects were from a similar background and considered themselves computer literate, although most did not have previous experience with a PDA. The initial group was given different, but similar in context, data than those that would participate in the actual experiment; however, the rules of the experiment were the same, apart from the duration of the experiment, which would only encompass the virtual timeframe of one week. The goal of this initial experiment was to ensure the system ran smoothly with users that were unfamiliar with it. A further, and more important goal, was to observe the average reading times for the web documents and their relation to explicit feedback, as we planned to use this metric for implicit relevance feedback.

Having analysed the results of these initial groups, it was observed that the average reading times were not what we expected, and were certainly in contrast with previous research such as that mentioned in Section 4.

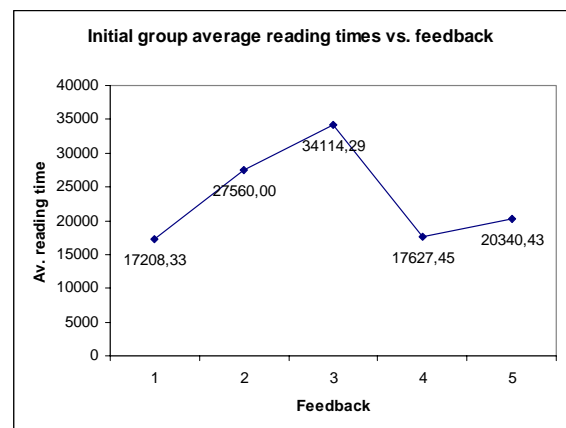


Figure 6: Initial group average reading times vs. feedback ratings

It is clear from this graph that the users take, on average, the same amount of time to distinguish between either relevant or largely irrelevant documents. Therefore it is apparent that the use of time as a metric is not a reliable source of information, since there is not much significant discrepancy between the average reading times for each feedback score. This observation brought about the decision to eliminate this metric from the weight recalculation formula, as it is in contrast with other findings, such as those by Morita [28] and Claypool [27], but seem to confirm Kelly's [29] conclusion that reading time is an unreliable source for implicit relevance feedback. While our research and that of Claypool's use scales of 1-5, Kelly uses a scale of 1-7. However what is more interesting than the direct comparison of observations is the fluctuation between these.

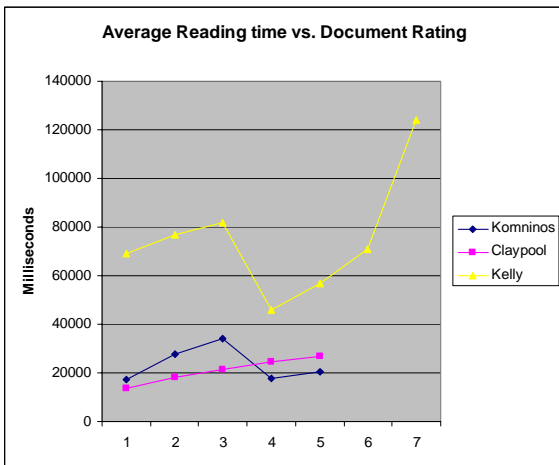


Table 3: Studies on average document reading times (msec) vs. perceived document usefulness

6.3 Actual experiment

For our actual experiment, two groups of ten people each were used. The rules were applied in full this time and we were able to obtain some interesting results at the end of the experiment. Unfortunately, due to data corruption on the logs of two members of one group, we were forced to exclude them from the analysis, removing also two random members from the other group, to make the figures directly comparable.

In the following graphs, a representation of the average reading times for both groups, over the three experiment weeks is depicted.

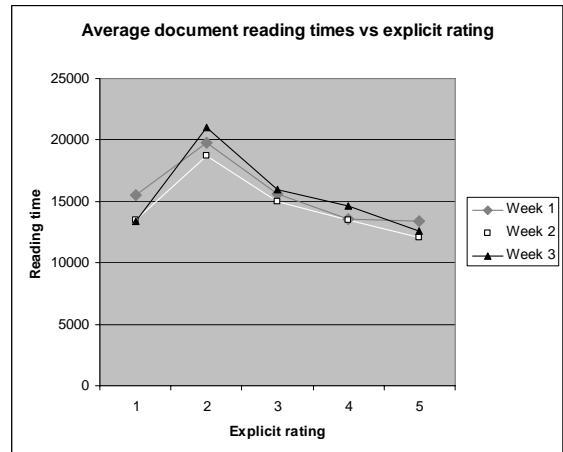


Figure 7: Experiment group average reading times vs. feedback ratings

The trend shown here is slightly different from the results of the initial group. This is expected as the data for the two groups were not the same. However, again from this graph one can clearly see that determining a relationship between feedback and reading time is not feasible. The average reading times for the top two and the worst mark are very close, making any secure distinction between the two almost impossible. Further to this, it is surprising that the group behaviour for all three weeks shows similar trends. This finding seems to confirm our suspicion that average reading times are strongly dependent on each individual test group characteristics and cannot thus be easily generalized.

Further analysis of the results should show whether there is a trend in the improvement of cache hits for the group whose data was tuned according to their previous logs. The following two tables show the numerical and percentile quantities of the opened documents (cache hits) vs. the total documents offered.

	Group 1	Group 2	Joint
Week 1	8,6%	24,4%	16.5%
Week 2	16.5%	18%	17.7%
Week 3	24.8%	30.5%	27.8%

Table 4: Total documents vs. Opened documents (percentage)

	Group1		Group2		Joint	
	Total	Opened	Total	Opened	Total	Opened
Week 1	1464	127	1464	357	2928	484
Week 2	1879	31	1880	357	3759	667
Week 3	1065	264	1248	381	2313	645

Table 5: Total documents vs. Opened documents (absolute values)

From table 4 above, one can clearly see an almost linear trend developing for group 1, who were the group that had their data adjusted according to their logs. This is a strong indication that for these users, the system provides an improvement in potential efficiency, if one considers that opening a document indicates the user's potential interest in it. For group 2, a solid conclusion cannot be made, as the percentage rates seem to fluctuate almost randomly, affecting of course the joint outcome as well. It would appear that for this group, the nature of the entries in their given schedules is the only determinant in the percentage of documents opened. For group 1 however, it appears that the application of interest indicators found in previous logs, has a restraining effect to the fluctuation of the variation in percentages and an overall effect that shows increased performance of the system.

As mentioned in section 4, a further indicator of interest in a document is its summary. A look at how many documents were judged by the summary reveals the following results:

	Group1		Group2	
	Immediate Open	After summary	Immediate Open	After summary
Week 1	90 (65.6%)	37 (34.4%)	255 (71.4%)	102 (28.6%)
Week 2	282 (90.9%)	28 (9.1%)	252 (70.6%)	105 (29.4%)
Week 3	209 (79.1%)	55 (20.9%)	205 (53.8%)	86 (46.2%)
Joint				
	Immediate Open		After summary	
Week 1	345 (71.3%)		139 (28.7%)	
Week 2	534 (80%)		133 (20%)	
Week 3	504 (78.1%)		141 (21.9%)	

Table 6: Summary viewing as a deciding factor for opening a document.

The percentages compare each figure with the total number of opened documents. From these trends we see that approximately only 1 in 4 times did the users consult the summary before making a decision. This suggests that a user will be inclined to navigate to a website based on the information contained in its title solely. A higher percentage was expected in this situation, especially since visiting a document is costly (in terms of loading and reading time) and also because in the implementation, explicit feedback was requested after each user had finished reading. According to Nielsen [26] also, users tend to like summaries and will read them before resuming with the rest of the text. However, it must be noted here that the summaries were only displayed on demand, while the document titles were immediately available. This was a necessary tradeoff in order to reduce the scrolling required for the retrieval results overview, although the action of opening a summary incurs an additional cost to the subjects.

Finally, the following table takes a look at the average document scores for each session:

	Group 1	Group 2	Joint
Week 1	2.26	2.37	2.44
Week 2	2.60	2.38	2.48
Week 3	2.49	2.36	2.42

Table 7: Average document scores (0-lowest, 5-maximum)

It would appear from this table that the scores remain at a constant level and in fact, at around the middle of the scoring table. This in turn is consistent with the results by Jansen [Jans03], where it is mentioned that users should expect 1 of 2 web documents they view to be relevant.

The choice was made not to measure the individual scores attained by each group in order to establish a trend. These would be actually just measuring the ability of Google to return relevant results, where as this research is only concerned with measuring the relevance of the web query in the context of the calendar entry and the user's needs.

6.4 Further discussion

6.4.1 Experimental environment

The experiment was performed in a quiet room. This setting might not appear to be realistic in the sense that there were no external distractions for the subjects, although they were given food and drink and were allowed to communicate and interact with each other. Mobile devices are used in both mobile and stationary environments and since the experiment shows that reading times are not long (around 25 seconds), it is expected that a mobile user could easily dedicate such small times to interact undisturbed. I perceive the notion of "mobile" to mean "out of office" rather than "walking" or "driving", therefore the setting seems adequate for the purposes of the experiment. In any case, a maximum time limit of 2 minutes, based on the observations from an initial test group, is imposed on the measuring to eliminate gross inaccuracies due to user distraction. Therefore it can be concluded that the environment settings for the experiment were appropriate for its purposes and did not deduct from its credibility.

6.4.2 Statistical Confidence

Further to the results of the test, a two-paired T-Test was conducted in order to investigate the statistical significance of the findings that were observed, in relation to the improvement of cache hit-rate improvement for the two groups. The t-test was the recommended approach as the experiment dealt with two groups of different subjects, who came however from a homogenous background, for one of which an external factor was applied and its effect was observed. This external factor was the monitoring and consideration of interaction and feedback behaviour, and its implication in the retrieval process.

The cache hit rates between week 1 and week 3 were measured for each individual subjects and their difference was analysed. The findings of the t-test are summarised in table 8 below. With a statistical probability of error of approximately 1.2% when considering whether the external factor was indeed responsible for the cache hit-rate improvement, the credibility of the results is further enhanced.

Mean— Meanb		t	df
0.1003		2.4985	14
P			
one-tailed	0.01277		
two-tailed	0.02554		

Table 8: T-test results

6.5 Summary of Findings

Several important conclusions were reached by this experiment. Firstly, the system shows that useful Internet content can indeed be pre-cached based on calendar information alone. This is shown by the cache hit rates, which rose close to 30%. Another important finding was that the reading behaviour of the subjects when faced with documents on a small screen, showed that the time spent on a document does not accurately reflect the quality of the document. A correlation between these two cannot be established, therefore the use of reading time for implicit relevance feedback on small screen devices is not recommended. Finally, and perhaps most importantly, the results of this experiment show a gradual, almost linear improvement of the retrieval performance for the group whose behaviours were taken into account. Although the duration of the experiment could have been longer, the statistical confidence is such that it can be argued that the results are solid enough to provide adequate confirmation of a promising learning curve performance.

7. Discussion and future work

We described a pre-caching system which is based on the information found in electronic calendars, in order to provide useful content for a user with a small mobile computing device. While such a system in its own right would not be able to completely satisfy all of a user's internet content needs or desires, we show that it can indeed provide useful content for the appropriate entry categories. Even in the case of entries where the information contained therein comprises of a single word, the automatic generation of web queries based on common knowledge and the users's preferences proves to be able to provide meaningful and useful content.

Given the opened document trends as described previously, we have reason to believe that our system is able to adapt accordingly to the individual preferences of a user. A further trial over an extended period of time would be able to show the fluctuation between improvement rates and whether a peak is reached, which would indicate the system's optimum performance level.

Further to this, the system currently lacks the ability to automatically add keywords to its knowledge base. Such an inclusion, we believe, would help dramatically in the improvement of the system's performance.

Apart from the findings that were part of our main target, we encountered several other interesting facts. The similarity between decision times for judging positively or extremely negatively against a document prohibits the use of such a metric from any further studies. Further to this, we were also impressed by the low reading times, which are in stark contrast with other studies that are concerned with the average reading time of a web document, such as [28], [27]. Other studies report average reading

times closer to the ones we experiences, but again higher [29],[30], although these were not based on web documents. However, all of these previous studies relate to documents viewed on a desktop, where a large monitor facilitates the viewing of documents. It is our assumption that the smaller reading times on the handheld indicate a tendency for users to "skim" through the document in order to decide on its usefulness. This should be considered normal, given that the need for immediate and full comprehension of the information in the text was not there (due to the virtual environment). Therefore the users would try to acquire a general "feel" for the quality of the searches, and refer to these later on when they have more time or immediately need the information. Nielsen argues that scanning the text in a web document is common practice. Further more, in his work, long pages that cause lots of scrolling are considered to be largely disliked by users. Since the limited size display on a handheld causes websites to appear unproportionately large and causes lots of scrolling, our findings of reduced reading times seem to be further supported by these statements.

8. References

- [1] Balabanovic M., Shohav Y., Yun Y., An Adaptive Agent for Automated Web Browsing, *Journal of Visual Communication and Image Representation* vol. 6 n.4, 1995
- [2] Wang Z., Crowcroft J., Prefetching in the World Wide Web, *Proceedings IEEE Global Internet Conference, London, 1996*
- [3] Cunha C., Jaccoud C., *International Symposium on Computers and Communication 97, Alexandria, Egypt 1997*
- [4] Thiebaut D., On the Fractal dimension of computer programs and its applications to the prediction of the cache miss ratio, *IEEE transactions on Computers*, 38(7), pp. 1012-1026, 1989
- [5] Palpanas T., Web Prefetching Using Partial Match Prediction, *Proceedings of the 4th International Web Caching Workshop, San Diego, CA, 1998*
- [6] Jiang, Z., Kleinrock L., An Adaptive Network Prefetch Scheme, *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 3, pp. 358-368, 1-11, 1998.
- [7] Jiang, Z., Kleinrock, L., Web prefetching in a mobile environment, *IEEE Personal Communications* 5, pp. 25-34, 1998
- [8] Chinen, K. Yamaguchi S., An Interactive Prefetching Proxy Server for Improvement of WWW Latency. *Proceedings of INET97, June 1997.*
- [9] Duchamp D., Prefetching Hyperlinks, *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, CO, 1999.*
- [10] Fan L., Jacobson Q., Cao P., Lin W., Web prefetching between low-bandwidth clients and proxies: Potential and Performance, *Proceedings of the Joint International Conference on Measurement and Modelling of Computer Systems (SIGMETRICS 99), Atlanta, Georgia, 1999*
- [11] Pitkow J., Pirolli P., Mining Longest Repeated Subsequences to Predict WWW surfing, *Proceedings of the Second USENIX Symposium on Internet Technologies and Systems, October 1999.*
- [12] Schechter S., Krishnan M., Smith M. D., Using path profiles to predict HTTP requests, *Proceedings of the Seventh*

- International WWW Conference, Brisbane, Australia, pp. 457-467, 1998
- [13] Padmanabhan V., Mogul J. C., Using Predictive Prefetching to improve WWW Latency, ACM SIGCOMM Computer Communication Review, 26 (3), pp. 22-36, 1996
- [14] Crow, D., Smith, B., DB_Habits, Comparing Minimal Knowledge and Knowledge-Based Approaches to Pattern Recognition in the Domain of User-Computer Interactions, Neural Networks and Pattern Recognition in Human Computer Interaction, pp. 39-63, NY, Ellis Horwood, 1992.
- [15] Swaminathan, N., Raghavan, S. V., Intelligent pre-fetching in WWW using client behaviour characterization. Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2000.
- [16] Foygel, D., Strelow, D., Reducing Web Latency With Hierarchical Cache-based Prefetching, Proceedings of the International Workshop on Scalable Web Services (in conjunction with ICPP0), Toronto, Canada, 2000
- [17] Davison, B., Predicting Web Actions from HTML content, Proceedings of the 13th ACM conference on Hypertext and Hypermedia, College Park, MD, pp. 159-168, 2002
- [18] Zhang, W., Lewanda, D. B., Janneck, C. D., Davison, B. D., Personalized Web Prefetching in Mozilla. Technical Report LU-CSE-03-006, Dept. of Computer Science and Engineering, Lehigh University, 2003
- [19] Davison, B., Learning Web Request Patterns, in A. Poulouvasilis and M. Levene, editors, Web Dynamics: Adapting to Change in Content, Size, Topology and Use, Springer 2004
- [20] Cohen, E., Kaplan, H., Pre-fetching the means for document transfer: a new approach for reducing Web latency, Proceedings of the 2000 IEEE INFOCOM conference, pp. 854-863, Tel-Aviv, 2000
- [21] Komninos, A., Dunlop, M.D. (2003): Towards a model for an Internet content pre-caching agent for small computing devices, Proceedings of the 10th International conference on Human Computer Interaction (HCI2003), Crete, Greece, 2003
- [22] Kincaid, C. M., Dupont, P.D., Kaye A. R., Electronic Calendars in the Office: An assessment of user needs and current technology, ACM Transactions on Office Information Systems 3(1), pp. 89-102, 1985
- [23] Appelt, D., Israel, D.J., Introduction to Information Extraction Technology, International Journal of Communications in Artificial Intelligence 12, pp. 161-172, 1999
- [24] Jansen, B.J., Spink, A., Bateman, J., Sarasevic, T., Real Life Information Retrieval: A Study of User Queries on the Web, ACM SIGIR Forum 32(1), pp 5-17, 1998
- [25] Jansen, B., Spink, A., An analysis of web documents retrieved and viewed, Proceedings of the 4th International Conference on Internet Computing, Las Vegas, Nevada, pp. 65-69, 2003
- [26] Nielsen, J., Morkes, J., Concise Scannable and Objective: How to write for the web, <http://www.useit.com/papers/webwriting/writing.html>, 1997, Link valid Jan 2005
- [27] Claypool M., Le, P., Waseda, M., Brown, D., Implicit Interest Indicators, Proceedings of the 6th International Conference on Intelligent User Interfaces (IUI '01), USA, pp. 33-40, 2001
- [28] Morita, M., Shinoda, Y., Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval, Proceedings of the 17th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 272-281, Ireland, 1994
- [29] Kelly, D., Belkin, N., Reading Time, Scrolling and Interaction: Exploring Implicit Sources of User Preferences for Relevance Feedback, Proceedings of the 24th annual international ACM conference on Research and Development in Information Retrieval, New Orleans, pp. 408-409, 2001
- [30] Kelly, J. D., Understanding Implicit Feedback and Document Preference: A naturalistic user study, PhD Thesis, State University of New Jersey, USA, 2004