Using Data Item Relationships to Adaptively Select Data for Synchronization

Oskari Koskimies

Nokia Research Center, Itämerenkatu 11-13, 00180 Helsinki, Finland
oskari.koskimies@nokia.com

Abstract. Though synchronization of email and PIM data to mobile devices has been a major practical concern for a long time, there has been relatively little advance in making synchronization work adaptively. We examine in this article the possibility to adapt synchronization to bandwidth and resource constraints by only synchronizing the items that are currently relevant to the user, and present initial results suggesting that item relationships are helpful for accomplishing this task.

1 Introduction

Adaptive synchronization has been studied at Nokia Research Center in connection with SyncML [1]. Instead of the traditional approach of protocol and content optimization, the idea is to intelligently partition the user's data between terminal and network, choosing what to synchronize and when so as to maximize efficiency. The task is complicated by the unreliable nature of wireless connections – if user data resides in the network, access to it will be slow when network conditions are poor, and impossible when in disconnected state. It is therefore imperative to ensure that data important to the user is available locally on the mobile terminal.

Selection of data must take into account usage context. For example, the data needed on a business trip is usually quite different from that required on a holiday. Synchronization should also adapt to the network, allowing more data to be synchronized when network conditions are favorable, and to the terminal, storing more files locally if there is ample space on the terminal.

2 Related Work

When examining prioritizing synchronization systems, the venerable CODA system, presented by Kistler and Satyanarayanan e.g. in [2], must of course be mentioned. To facilitate disconnected operation, the CODA distributed file system attempts to ensure that critical files are available in the cache. The process utilizes both access history as well as explicit configuration information provided by the user.

General-purpose email classifiers used to identify spam [3][4] could be applied to our problem as well. Instead of looking for email similar to known spam, the classifier would look for email similar to emails known to be important (e.g. by

L. Kutvonen and N. Alonistioti (Eds.): DAIS 2005, LNCS 3543, pp. 220 – 225, 2005.

observing user email reading behavior). The problem with this approach is that the importance of emails depends on current tasks and may change on a daily basis; traditional classifiers do not adapt fast enough.

The closest related work is actually related to email prioritization. The main difference in this case is that when prioritizing email for presentation, you only care about the unread emails. This is reasonable since the user knows about the already read emails and can easily view them if desired. In synchronization, however, also read emails are important, since they may contain information that the user needs while mobile. For example, an old email about a soon-to-be meeting might contain instructions for driving to the meeting venue. Examples of static rule systems include CLUES [5] and Bifrost [6]. CLUES uses information from a user's work environment (calendar entries, sent emails, etc.) to extract clues about his short term-interests for the purpose of prioritizing the messages. The Bifrost system organizes a user's inbox into groups (messages related to calendar events, personal messages, etc.), which enables the user to more easily spot important mails. PRIORITIES [7], on the other hand, is a learning system which classifies emails according to criticality based on sender identity and relation to user, number of recipients, textual content, etc.

3 Adaptive Synchronization

Adaptive synchronization provides a way for autonomously determining what are the most important data items for synchronization in a given context. The algorithm utilizes application-specific a priori knowledge about data items the user is certainly going to need (e.g. new mails) and follows relationship paths between data items to find other data items related to the needed items. The underlying basic assumption is that if the user needs a certain data item, then he is also likely to need related data items. The use of relationships is necessary to identify data items that may be quite old and of little apparent importance by themselves, but are relevant to the user because they are closely related to other, currently important items.

The idea behind our approach is that relationships between data items describe the probability that if the user needed the first item, he will also need the second item. For example, the relationship between an email and an attachment of the email might have the weight 0.9, signifying that if the user needed the email, with 90% probability he will also require the attachment. We refer to this weight as the *relevance* of a relationship. The *utility* of an item quantifies the benefit of the item to the user.

An initial set of data items, which are certain to be needed by the user, is assumed to be known (for example, new mail and today's calendar entries). Determining the initial set is application-specific. The expected utility (overall importance) of an item depends then on both its utility and on the relevance of the relationship path(s) to it from the initial set, which represents the probability that the user will in fact need the item. The expected utility of an item can be penalized in the final calculation with the size of the item, so that large data items are synchronized only if they have very high expected utility (optimizing "utility per kilobyte").

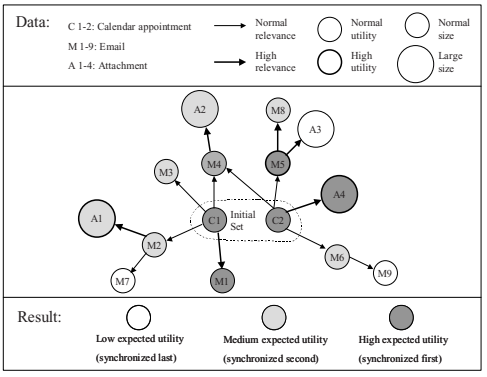

Fig. 1. Adaptive synchronization example

In Figure 1, an example relationship graph is shown. The figure also shows an overview of relevancies, utilities, item sizes and calculated expected utilities. Note that in general, the relationship graph will contain loops. However, as it is our purpose to explain the general idea, the handling of loops is an unnecessary complication.

Calendar appointments C1 and C2 in Figure 1 make up the initial set. All relationship paths are calculated from that set. The email messages M1-M6 are from people participating in one of the meetings, the sender of M4 participates in both. The email M5 has high utility because the sender had set the high importance flag on. Email M7 is an earlier mail from the same sender as M2, similarly for M9 and M6. Email M8, on the other hand, is an email from both the same thread and sender as M5. Attachments A1, A2 and A3 belong to emails M2, M4 and M5 respectively, and attachment A4 belongs to calendar appointment C2. The attachments are much larger than emails. Attachments A1 and A4 have high utility because they are PowerPoint files which can be viewed on a mobile terminal relatively efficiently, unlike A2 and A3 which are MSWord documents. Relationships M2A1, M4A2, M5A3 and C2A4 have high relevance since an attachment is usually highly relevant to the containing email/appointment. Relationship C1M1 has high relevance because the organizer of the meeting has sent M1. Relationship M5M8 has high relevance because M8 has both same sender and thread as M5.

The approach takes context into account in two ways. Firstly, the initial set contains currently relevant items (e.g. recent mails and appointments), which causes the algorithm to automatically select items related to the user's current context. Secondly, relationship and utility weights can be adjusted based on context. For example, if the user expects to have low-bandwidth connectivity available continuously, utility of large items might be increased since small items can be fetched on-demand using the low-bandwidth connection. Similarly, the age of an item might affect its utility.

4 Interim Results

To evaluate the correctness of our assumption that being related to another important item is a good predictor for item importance, we need to have a priori knowledge of items the user actually requires next. We are implementing a tracker application for Microsoft Outlook, which enables us to compare the items selected by the algorithm to the items that were actually required by the user. In the meantime, we have done some initial tests using a simple HTML form generated from the contents of the user's Microsoft Outlook data. The users were asked to assume that they were about to go on a one-week business trip, and then grade in that context the importance of each item on a scale from 1 to 5 using the form. The rather laborious form-filling aspect of the test setup prevented the gathering of a statistically significant set of data; however, the initial results are already encouraging, and when the Outlook Tracker application is finished, collection of user data will be much simpler.

With user-provided importance data available, we tested the hypothesis for each relationship type R and item type A by examining the probability that type A items, targeted by a type R relationship originating from an important item, were important only randomly. The probability was calculated using the Hypergeometric distribution. In testing with two large email sets from two different users, we got the results shown in Table 1. If the probability P("importance is random"), denoted as P(rand), is very small (<0.05), then we can with high confidence say that type R relationships are indeed good predictors for importance. The entries that signify over 95% confidence

Table 1. Relationship predictive power

Relationship	Source item type	Target item type	P (rand) Dataset1	P (rand) Dataset2
Previous email by same sender	Email	Email	**0.00046**	**0.02095**
Next email by same sender	Email	Email	**0.00001**	**0.00578**
Last email from same sender	Email	Email	**0.00071**	0.10272
Email from organizer	Calendar entry	Email	0.05029	0.86095
Email from required participant	Calendar entry	Email	**0.02661**	**0.04725**
Contact information for sender	Email	Contact information	**0.00000**	**0.00000**
Contact information for receiver	Email	Contact information	**0.00000**	**0.00068**
Contact information for organizer	Calendar entry	Contact information	**0.00000**	0.09370
Contact information for required participant	Calendar entry	Contact information	**0.00055**	**0.00109**

in the predictive power of the relationship are in bold. For the most part, the results are quite intuitive, but it is surprising that emails from meeting participants are more likely to be important than emails from the meeting organizer. This is likely because the organizer usually sends information related to the meeting as updates to the calendar entry, which are processed by Microsoft Outlook into the original calendar entry and are thus not visible as independent email entries. Other participants, on the other hand, have to rely on normal email for issues related to the meeting.

Note that there are no relationships in the above table that would point to calendar entries. Several such relationships were in fact examined, but none were good indicators, with even the best one (meeting participant is email sender) having a value of around 0.38 for dataset 1, i.e. a 38% probability of having no predictor value. We conclude that while relationships *from* calendar entries make good predictors, the same is not true for relationships *to* calendar entries. In other words, calendar entries are very good candidates for the initial set.

The selection accuracy of the algorithm was slightly above 90% for the two datasets; reasonable considering that no machine learning was used, but not yet good enough for practical use. The percentage of important items was 13% for Dataset 1 and 26% for Dataset 2.

5 Summary and Future Work

We have developed a relationship-based approach for autonomously selecting emails for synchronization based on their importance. The interim results indicate that relationships are good predictors for email importance, but the algorithm would need machine-learning features to perform well enough. We plan to combine the relationship-based approach with a traditional classifier approach to achieve this.

We are in the process of instrumenting Microsoft Outlook so that it records user activity such as the order and speed in which emails are read. From this information, an importance classification will then be extracted and used to more rigorously test our hypothesis of item importance and to evaluate the performance of the selection algorithm. The Outlook Tracker will also provide us a method for gathering data from which the selection algorithm can learn and adapt to user behavior.

References

[1] Open Mobile Alliance: “SyncML Data Synchronization Specifications”, Version 1.1. Available electronically from http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html.

[2] Kistler J. J. and Satyanarayanan M.: “Disconnected Operation in the Coda File System”. ACM Transactions on Computer Systems, Vol. 10, No. 1, pages 3-25, February 1992.

[3] Graham, P.: “A Plan for Spam”, August 2002. Available electronically from http://www.paulgraham.com/spam.html

[4] Yerazunis W.: “Sparse Binary Polynomial Hashing and the CRM114 Discriminator”, in Proceedings of the 2003 Spam Conference, January 2003. Available electronically from http://spamconference.org/proceedings2003.html.

[5] Marx M. and Schmandt C.: “CLUES: Dynamic Personalized Message Filtering”, in Proceedings of CSCW `96, pages 113-121, November 1996.

[6] Bälter O. and Sidner C. L.: “Bifrost Inbox Organizer: Giving users control over the inbox”, in Proceedings of the Second Nordic Conference on Human-computer Interaction, pages 111-118, October 2002.
[7] Horvitz E., Jacobs A. and Hovel D.: “Attention-Sensitive Alerting”, in Proceedings of the 15th Conference on Uncertainty and Artificial Intelligence, pages 305-313, July 1999.