An Experimental Framework for Email Categorization and Management

Kenrick Mock
University of Alaska, Anchorage
3211 Providence Dr.
Anchorage, AK 99508
(907) 786-1956
afkjm@uaa.alaska.edu

ABSTRACT

Many problems are difficult to adequately explore until a prototype exists in order to elicit user feedback. One such problem is a system that automatically categorizes and manages email. Due to a myriad of user interface issues, a prototype is necessary to determine what techniques and technologies are effective in the email domain. This paper describes the implementation of an add-in for Microsoft Outlook 2000 ™ that intends to address two problems with email: 1) help manage the inbox by automatically classifying email based on user folders, and 2) to aid in search and retrieval by providing a list of email relevant to the selected item. This add-in represents a first step in an experimental system for the study of other issues related to information management. The system has been set up to allow experimentation with other classification algorithms and the source code is available online in an effort to promote further experimentation.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information Filtering, Retrieval Models*. H.4.3 [**Applications**]: Communications Applications - *Electronic mail*

General Terms

Experimentation, Human Factors

Keywords

Email management, filtering, classification

1. INTRODUCTION

Email overload has become a growing problem as more users embrace online technologies. Time Magazine estimated that 776 billion email messages were sent in 1994, 2.6 trillion sent in 1997, and 6.6 trillion sent in 2000 [2]. In one study, recipients averaged around 30 email messages per day [1]. To address this problem, researchers initially designed systems to automatically classify incoming email into categories or folders using various machine learning techniques.

This early work focused on the classification accuracy of the algorithms on sets of test data. While classification metrics on test sets can provide valuable information as to the effectiveness of a classifier, an email classifier must deal with a multitude of user design issues. As one example, many users leave email in the inbox as a reminder regarding some task [5]. This mode of operation is disrupted by a system that automatically files email.

As another example, many classification algorithms are accurate but require minutes or even hours to train. However, users will typically only be willing to wait a number of seconds, not minutes [4]. Furthermore, what kind of errors are users willing to tolerate? Error rates lower than 1% may be enough to warrant discarding the entire system if the error is made on crucial email.

To address these issues, recent work has focused on experimental systems. For example, SwiftFile used shortcut buttons to file messages into folders, but only when initiated by the user. The system also incorporated an incremental learning algorithm [5]. Other projects such as Relevance Categories, Enfish Onespace, or Metastorm's infowise product use information retrieval techniques to provide relevance to folders or individual messages [4]. Other companies such as Abridge, Plumtree, and Tacit use rules or user-supplied categories to group email.

2. PROPOSED APPROACH

This project is a first step toward building an experimental system that may be used to test different ideas for categorization and management within a real email environment. The project includes utility-like routines such as extracting features from email, assigning numeric values to features, stop listing, putting email into categories, or detecting when email arrives. These routines are used as the basis for the system described in the rest of the paper.

SIGIR '01, September 9-12, 2001, New Orleans, Louisiana, USA.

Additional classification algorithms or email access paradigms may be built on the existing framework.

The initial prototype is based on expected user behavior. Whittaker and Sidner identify three types of email users: 1) those that use no folders and rely on search tools to find mail in their inbox, 2) those that file frequently into folders, and 3) those that file intermittently into folders every few months [6]. This work describes two tools that are expected to help users in these categories. The first is a tool that automatically groups inbox email within categories, and the second is a tool to aid searching for relevant email.

2.1 Automatic Grouping of the Inbox

On one hand, we would like to preserve email in the inbox so that users can keep them available as reminders for action items. On the other hand we would like to file the email to keep the volume manageable. A middle ground is to classify email into groups, but leave them in the inbox until filed or deleted by the user.

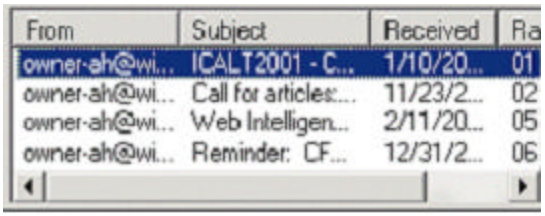
The email add-in prototype addresses this middle ground by building classifiers based on user-created folders. Each classifier is constructed by scanning through all email the user has placed into the folder. The features from the emails are extracted and added to the classifier. The system is currently capable of extracting as features terms from the subject, author, recipient, and body of the text. Stop-list terms are removed and weights are assigned to each term based on their frequency in the email.

Currently, the classifier is a simple nearest-neighbor (NN) classifier. Given a target message to classify, its features are extracted and compared to all messages in the classifier using the cosine coefficient. The top three matches are averaged as the similarity measure for the classifier. The message is then put into the classifier with the largest similarity measure. While ad-hoc, the initial focus is to create the infrastructure and then experiment with new classifiers in the future. However, this classifier does satisfy the necessary criteria of speed and supports incremental updates. NN classifiers may even be more effective than classifiers based on global information since some users create generic folders (e.g. "Projects") encompassing multiple sub-categories [4]. Once a message is classified, it is grouped within the inbox by category. This allows the user to view messages by category, by date received, by author, or any other field.

2.2 Finding Relevant Email

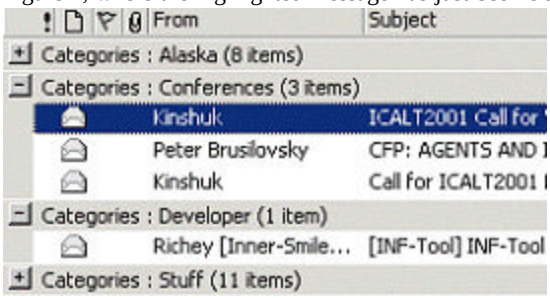
To aid users that wish to search for email, the add-in provides the capability to quickly display a list of messages ranked by relevance (using the similarity metric) to the selected message. In this manner, other messages in the same thread or in the same topic will be displayed at the top of the list.

This feature is currently implemented by simply scanning through all messages in the classifiers, comparing the selected message to each and saving the top matches. While brute-force, the prototype scans about 300 messages per second using a 400Mhz Pentium II.



3. IMPLEMENTATION

This project was implemented as an add-in for Microsoft Outlook using Visual Basic and Visual C++. Outlook's "Categories" field is used to store the classification and the object model exposes the necessary interface to access email messages and events such as the arrival of a new message. Outlook's interface already supports a view that groups email by category. This is illustrated in Figure 1, where the highlighted message has just been classified



and grouped into the category "Conferences."

Finding relevant messages was implemented by adding a "Similar Messages" button to the toolbar. The ranked results are shown in a new window, depicted in Figure 2.

While optimization was not an early emphasis of the project, one step was taken to increase performance. All string-based terms are hashed into 32 bit values [3]. This greatly increases the speed required to compare terms and cuts memory use in half. The current code for the project is available on the web at: http://www.math.uaa.alaska.edu/~afkjm/emailaddin/

Figure 2. Finding Relevant Email.

4. FUTURE WORK

The next phase is to perform user testing and to gather feedback on the effectiveness of the methods described here. Another area of work involves integrating thread information within the relevance view. Additional work also needs to be done to select an effective classifier – e.g. one that is incremental and can handle multiple sub-categories. Much work remains to be completed in code enhancements such as latching into additional Outlook events, database integration for classifiers, or .NET upgrades. Finally, new experiments that integrate classification and information retrieval techniques across email and into calendaring, notes, or other types of data may also be explored.

5. REFERENCES

[1] Balter, O. and Sidner, C. Bifrost Inbox Organizer: Giving users control over the inbox. Lotus TR 00-08, 2000.

[2] Gwynne, S. and Dickerson, J. Lost In The E-Mail. Time Magazine, April 21, 1997.

[3] Jenkins, B. Algorithm Alley: Hash Functions. Dr. Dobbs 22, 9, September 1997.

[4] Mock, K. Dynamic Email Organization via Relevance Categories. In Proceedings of the International Conference on Tools with Artificial Intelligence ’99 (Chicago IL, Nov 1999).

[5] Segal, R. and Kephart, J. Incremental Learning in SwiftFile. In Proceedings of the Seventh International Conference on Machine Learning (June, 2000).

[6] Whittaker, S. and Sidner, C. Email Overload: Exploring Personal Information Management of Email. In Proceedings of CHI 96: Human Factors in Computing Systems. New York, NY, 1996. ACM, p 27