MailCat: An Intelligent Assistant for Organizing E-Mail

Authors:
Richard B. Segal & Jeffrey O. Kephart

Presenter:
David Cantor

February 20, 2008
CS 525U – Intelligent User Interfaces

Overview

- Introduction
- MailCat
- Evaluation
- Related Work
- Conclusions and Future Work
- Questions?

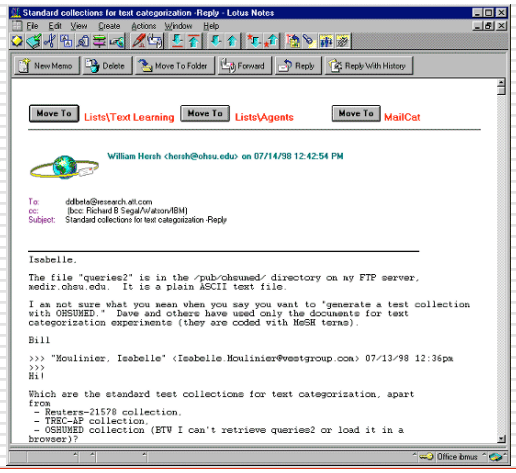
Introduction

- The Problem
  - Overwhelming amount of email
- The Old Solution
  - Hierarchy of folders
  - Move emails to appropriate folder
- New Problem
  - Must think about where each email belongs
  - People would rather leave email in Inbox

Introduction

- Solution
  - MailCat: An intelligent personal assistant to sort email
    - Extension to Lotus Notes
      - Extensive C++ API allows such an add-on
    - Determines three best folders
      - Buttons appear in each email
      - User can easily ignore
    - Takes burden away from user
      - Saves time and thought
    - Available as soon as installed
      - Based on user's previous sorting behavior

MailCat: Screenshot



MailCat

- Based on text classifier called AIM:
- *M* is a message
- *w* is a word
- *f* is a folder
- *F*(*M*, *w*) = how many times *w* appears in *M* (i.e. the *word frequency vector*)
- *F*(*f*, *w*) is *f*'s *centroid vector*:
  - = $\Sigma_{M \in f} F(M, w)$ or how many times *w* appears in *f*

MailCat (2)

- TF-IDF principle states: the *weight* assigned to a word is proportional to its frequency in the folder and inversely proportional to its frequency in other folders:
- $FF(f, w)$ is the *fractional frequency*:
  - $= F(f, w) / \Sigma_{w' \in f} F(f, w')$ or how many times w appears in f divided by the total number of words in f.
- $TF(f, w)$ is the *term frequency*:
  - $= FF(f, w) / FF(A, w)$ where A is the set of all messages
- $DF(w)$ is the *document frequency* or the fraction of folders where w appears at least once.
- $IDF(w)$ is *inverse document frequency* or $1 / DF(w)^2$

MailCat (3)

- $W(f, w)$ is the *weight* for word w in folder f:
  - $= TF(f, w)$ X $IDF(w)$
- What does this mean?
  - Basically, the chances a word belongs in a certain folder *increases* the more its already found there, and it *decreases* the more its found in other folders.

MailCat (4)

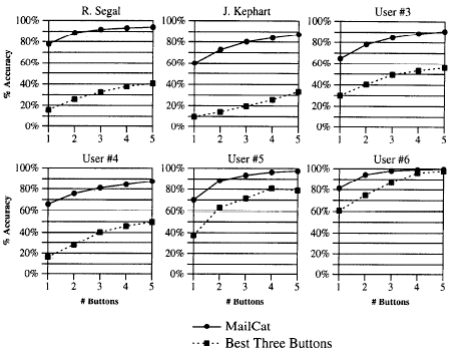
- Adding up all the words in *M*:
  - *SIM4*(*M*, *f*) or the *similarity* between message vector and the weighted folder vectors:
    - $= \Sigma_{w \in M}[F(M, w)W(f, w)] / \min[\Sigma_{w \in M}F(M, w), \Sigma_{w \in M}W(f, w)]$ (found using a variation of cosine distance)
    - The three folders with the highest results are chosen for the three suggestion buttons.
- Problem: As designed, AIM is not incremental.
  - Weights for words in all folders must be recalculated each time a message is moved
    - Excessive organizing will slow down system

MailCat (5)

- Solution: Modify AIM to ignore this recalculation.
- Whenever a message is moved to a folder, add its *word-frequency vector* to the folder's *centroid vector*:
  - $F(f, w) \leftarrow F(f, w) + F(M, w)$
- And whenever a message is removed from a folder, subtract:
  - $F(f, w) \leftarrow F(f, w) - F(M, w)$
- MailCat stores the *centroids* and recalculates the *weighted-word frequency vector* on the fly

Evaluation

- MailCat was applied to the mailboxes of six researchers IBM Thomas J. Watson Research Center (including the authors)
- Accuracy for 3 buttons was 80-98%
- How about 4 or 5?
  - Too many
  - 3 is an effective tradeoff between accuracy and non-intrusiveness

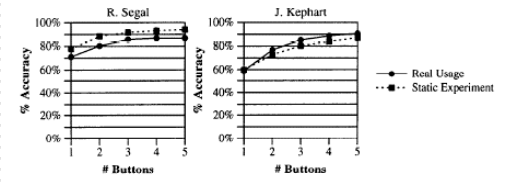


Evaluation (2)

- Problem with previous experiment:
  - Only ran static environment
  - Needed to see what happens with a real user
    - Creating
    - Deleting
    - Moving around
- Solution:
  - Segal and Kephart use MailCat for two months

Evaluation (3)

- Results of dynamic experiment:



- Overall, MailCat appears to be effective

Related Work

- CAP, a calendar scheduling program
  - Predicts best default values based on users' preferences
    - Shares MailCat's non-obtrusive nature
- Maxims, another mail organizer
  - Also deals with email sorting, but initial setup takes into account other users' mailboxes
    - Segal and Kephart argue this is inefficient because different users have different filing habits

Conclusions and Future Work

- MailCat contribution is simplicity
  - Takes the guesswork away from the user
  - User can start using as soon as installed
  - User can easily ignore suggestions as if MailCat wasn't installed
- Authors hope to expand MailCat to other sorting applications
  - Bookmarks, files, etc.
- SwiftFile: MailCat's successor
  - Experiments more in-depth
  - Would like to compare against other text classifiers

Questions?

References

- Segal, R. B., & Kephart, J.O. (1999). MailCat: An intelligent assistant for organizing e-mail. Proceedings of the Third International Conference on Autonomous Agents (pp.276-282). New York: ACM Press.
- Segal, R. B., & Kephart, J.O. (2000?). Incremental Learning in SwiftFile. http://www.research.ibm.com/swiftfile/dynlearn.pdf