CutOnce - Recipient Recommendation and Leak Detection in Action

Ramnath Balasubramanyan
Language Technologies Institute
Carnegie Mellon University
rbalasub@cs.cmu.edu

Vitor R. Carvalho
Language Technologies Institute
Carnegie Mellon University
vitor@cs.cmu.edu

William Cohen
Machine Learning Department
Carnegie Mellon University
wcohen@cs.cmu.edu

Abstract

Email has become increasingly ubiquitous in recent times bringing with it new problems. In this paper we revisit two such problems, namely *information leak detection* and *recipient recommendation*, and study the impact of previously proposed solutions on real email users. Previous work addressing these problems showed a lot of promise on static email corpora tests. In this paper, we implement these solutions in an integrated interface as an extension to **Mozilla Thunderbird**, a popular open source email client. By capturing data about the usage patterns in the extension, we evaluate the performance of the proposed algorithms against baseline methods and address issues related to user interface design and implementation. Preliminary results from a user study show promising usage patterns, indicating that these features can benefit a large number of email users.

Introduction

The network of contacts maintained via email has been growing steadily over the years. Until relatively recently, email was used primarily for work-related contacts by most people. Now email is used for many contact with many overlapping social circles, including friends, neighbors, and relatives, as well as social circles associated with various on-line communities. Moreover, the number of people with multiple email addresses (eg separate work and personal emails) has grown. All of these changes make it harder for users to choose the right place to send a message they have composed.

In past work, several specific problems have been explored. Carvalho and Cohen (2007) considered the problem of *email leaks*, where the problem of detecting and preventing messages from being sent to unintended recipients was addressed. This problem was also addressed by Boufaden et al. (Narjs Boufaden 2005). Another task considered by Carvalho and Cohen (Carvalho and Cohen 2008), Pal and McCallum (Pal and McCallum 2006) and others (Byron Dom 2003) is *recipient recommendation*, where the task consists of finding persons who are potential recipients for a message under composition given its current contents. Recipient recommendation is related to the problem of address auto-completion. The work by Carvalho and Cohen improves

upon the text similarity to address book entries method used in many email clients currently, by using the message content to predict recipients.

Prior work on these tasks has evaluated the effectiveness of various algorithms in experiments with static email corpora. Such studies cannot address a number of crucial issues. What is the best way to present predictions to end users (e.g., the prediction that an outgoing email address is a "leak")? How accurate do predictions need to be in order to be considered helpful to users? Are learning methods for these tasks efficient enough to be used in current email clients? How do the tasks of recipient recommendation and leak detection interact?"

To address these questions, we build an extension, CutOnce[1] that adds these capabilities to Mozilla Thunderbird, a popular and open source email client. The extension is designed to elicit the impact of the proposed algorithms by toggling between baseline predictions and predictions using the algorithms on a per user basis. All aspects of the usage of the extension are logged and periodically emailed (if express permission is given by the user) to us enabling us to study the impact of the algorithms on users who use email as a part of their day to day activities.

The paper is organized as follows. We introduce both message addressing problems, and explain the choice of email client in the next section followed by a section with the details of the proposed algorithms. We then discuss the issues with implementation in Mozilla Thunderbird, followed by a section that explains the logging scheme. We end by describing our learnings from the analysis of user logs and finally presenting our conclusions.

Recipient Recommendation and Leak Detection

The widespread use of email has raised serious privacy concerns. A critical issue is how to prevent email information leaks, i.e., when a message is accidentally addressed to non-desired recipients. This is an increasingly common problem that can severely harm individuals and corporations — for instance, a single email leak can potentially cause expensive

---

[1]Derived from a quote in *Neuromancer* by William Gibson where Case says "Easy, mon. Measure twice, cut once, wise man put it ... "

law suits, brand reputation damage, negotiation setbacks and severe financial losses.

Another important message addressing problem is recipient recommendation, i.e., recommending persons who are potential recipients for a message under composition given its current contents, its previously-specified recipients or a few initial letters of the intended recipient contact. This task can be a valuable addition to email clients, particularly in large corporations, where negotiations are frequently handled via email and the cost of errors in task management is very high. A system with intelligent message addressing can prevent a user from forgetting to add an important collaborator or manager as recipient, preventing costly misunderstandings, communication delays and missed opportunities.

In this paper, recipient recommendation is achieved by displaying a ranked list with all recipients from the user's address book. As described below, different algorithms can be used to rank recipients.

The same algorithms can also be used to provide email leak prevention. However, instead of ranking all entries in the address book, only the already-specified recipients of a message under composition need to be ranked (with the least likely address on the top).

Email Clients

Selecting an email client in which the recipient recommendation and leak detection algorithms could be implemented depended on several factors such as the ease with which it could be modified to incorporate new features, popularity of the client, whether or not the client is open/source, ease with which the modifications can be distributed to other users, operating system interoperability etc.

The options we considered included Mozilla Thunderbird, a new standalone email client which we would have to develop from scratch, GMail etc. Developing a new email client had the disadvantage that it would take a long time for it to be used widely if at all. Morever considerable effort would have to be put into engineering efforts which was peripheral to the issue at hand. GMail has the advantage of being widely used especially in the academic community, however the API offered by GMail was inadequate for our needs. Mozilla Thunderbird on the other hand is very popular, has a well established mechanism to add extensions and is open source which makes it an excellent platform to implement new features on. Extensions that we develop can be distributed easily in a zip-based format and installed using Thunderbird's Extension Manager.

Mozilla Thunderbird extensions are developed primarily in Javascript. User interfaces are specified using a Mozilla specific XML based file format called XUL.

Algorithms

The algorithms chosen for implementation in the Mozilla Thunderbird extension have to be computationally inexpensive since Javascript is a slow interpreted language. Expensive operations in Javascript tend to bog down the email client and make it virtually unusable.

Recency and Frequency based ranking

Recency ranking ranks candidate recipients by ranking recipients to whom messages were sent more recently higher using an exponential decay function.

$$recency(r) = \sum_{doc \in D(r)} e^{\frac{-timerank(doc)}{\tau}} \qquad (1)$$

$timerank(doc)$ is the chronological rank of doc with $timerank(doc)$ being 1 for the most recently sent message. $D(r)$ is the set of messages in the Sent folder that were sent to the recipient r. Based on preliminary tests, τ is set to 0.01 in CutOnce, highlighting the importance of the 100 most recent messages.

Another baseline is based on the frequency to whom previously sent messages were addressed. Frequency ranking orders candidate recipients by the number of messages addressed to them in the *sent* folder, i.e.,

$$frequency(r) = \sum_{doc \in D(r)} 1 \qquad (2)$$

TFIDF method

We can treat recipient recommendation as a multi-class classification problem and use the TFIDF classifier using the Rocchio algorithm as described elsewhere (Joachims 1997; Carvalho and Cohen 2007). The centroids for each recipient, represented as TFIDF vector over terms, is first computed by iterating through the Sent folder in the email client.

$$centroid(r) = \frac{1}{|D(r)|} \sum_{doc \in D(r)} tfidf(doc) \qquad (3)$$

For a new message new_doc that is being composed, we compute its cosine distance with each recipient centroid and rank the recipients according to this distance.

Aggregating Rankings with Data Fusion

The ranks obtained by the recency, frequency and TFIDF methods can be combined using data fusion techniques based on the Mean Reciprocal Rank (MRR) (Aslam and Montague 2001; Craig Macdonald 2006; Ogilvie and Callan 2003) of the baseline rankings. Results from (Carvalho and Cohen 2008) showed that MRR ranking works better than a pure TFIDF ranking for tests using several users from the Enron email collection.

The MRR ranking score can be expressed as:

$$mrr(r) = \frac{\alpha}{recency\_rank(r)} \qquad (4)$$
$$+ \frac{\beta}{frequency\_rank(r)} + \frac{\gamma}{tfidf\_rank(r)}$$

, i.e., the final aggregated ranking of a recipient is a function of the ranking of this recipient on the base rankings (TFIDF, frequency and recency).

Based on preliminary tests, we set α and β to 1.0 and γ to 2.0 by default.

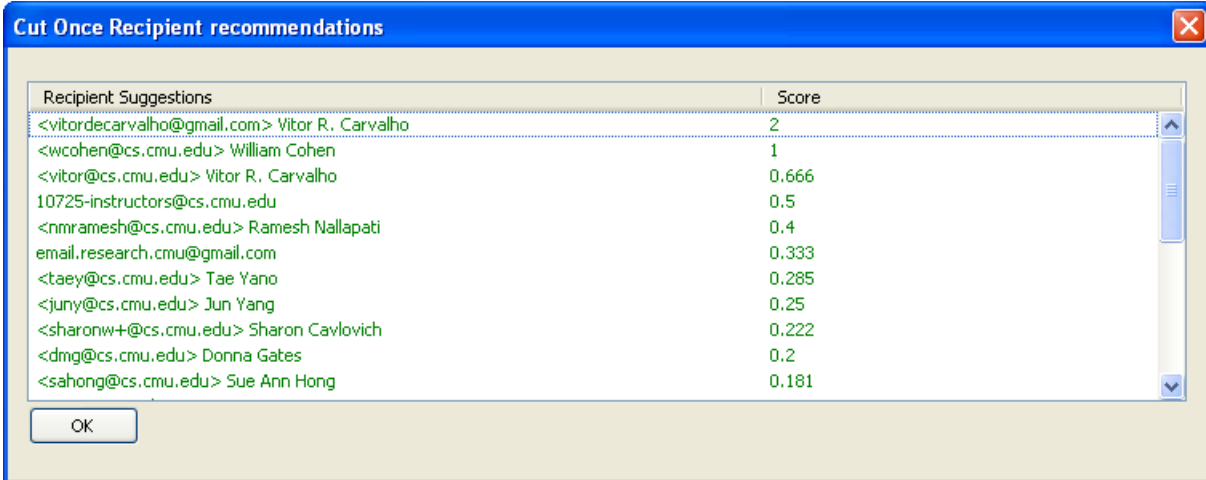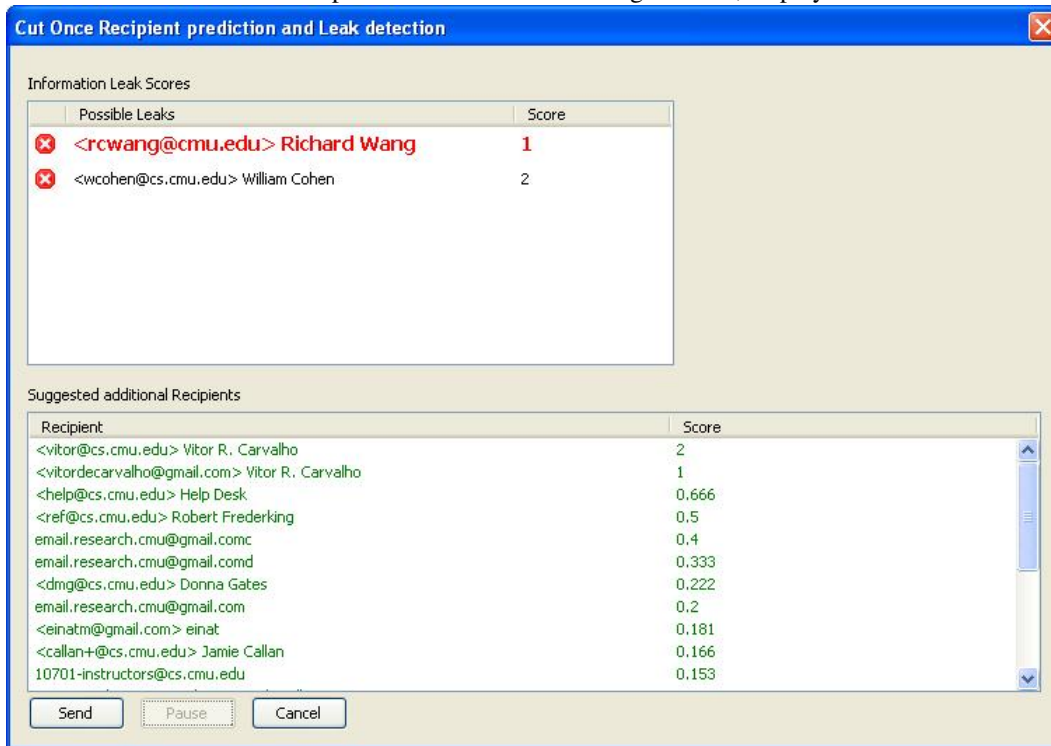Figure 1: The recipient recommendation dialog window

Figure 2: The information leak and recipient recommendation dialog window; displayed when Send button is pressed.

Implementing the Training in Thunderbird

Since the algorithms are implemented in Javascript, scalability and computation time are significant factors. The memory available to the extension is also limited since computation occurs on client machines. Keeping this in mind, steps were taken to keep the training time in check to limit the impact on user experience. Firstly, all words with a document frequency lower than a fixed threshold (set at 5) were eliminated from the TFIDF representation. Secondly centroids for recipients to whom the number of messages sent was below a threshold (set at 5), were not calculated. The procedure used to compute the centroids is described in Algorithm 1. After the model is trained, the parameter values are stored in a text file on the user's computer. When the client is restarted, this model file is read in thus preventing the need to retrain the system each time the client is started.

Prediction

The runtime components of CutOnce are triggered using two mechanisms

Algorithm 1 Training procedure

for all doc in Sent folder **do**
  for recipients r of doc **do**
    $centroid(r) = centroid(r) + tfidf(doc)$
    $n(r) = n(r) + 1$
  end for
end for
for all recipients r **do**
  $centroid(r) = centroid(r)/n(r)$
end for

i) When users send a message, a dialog box pops up with potential leaks and a list of recommended recipients. Items on the list can be clicked on to add recipients from the recommendation list or to remove them from the recipient list for the leak list. This dialog box has a countdown timer that sends the message after 10 seconds if the user does not take any action thus ensuring that no additional action is needed to send a message. (See Figure 2 for screenshot)

ii) The compose window has a Recommend recipients button that pops up a window with a list of recommended recipients for the message being composed. Recipients can be added to the message by clicking on the suggested recipients just as in the Send dialog. (See Figure 1 for screenshot)

Evaluation against baseline methods

Ranking recipients based on TFIDF similarity was used as a baseline method. Studying the improvement offered by the proposed algorithms over the baseline method requires statistics about the performance of the baseline. To obtain this, the extension uses baseline predictions for roughly half the installations (chosen randomly at installation time). When the logs returned by users are studied, indicators in the log specify whether the baseline method was used enabling comparisons. The extension also allows user to change the relative weight assigned to the TFIDF ranking method in relation to the weight assigned to frequency and recency.

Logging

CutOnce logs information about all aspects of the usage of the extension. The following pieces of information are logged

- Whether the use used the explicit Send button or let the timer countdown

- whether the user deleted a recipient (possibly due to a potential leak)

- rank of the deleted recipient in the potential leak list

- the MRR score of the recipient deleted

- the amount of time elapsed before the recipient was deleted

- whether recipients were added from the recommendation list

- rank of the added recipient in the recommendation list

Figure 4: Log message
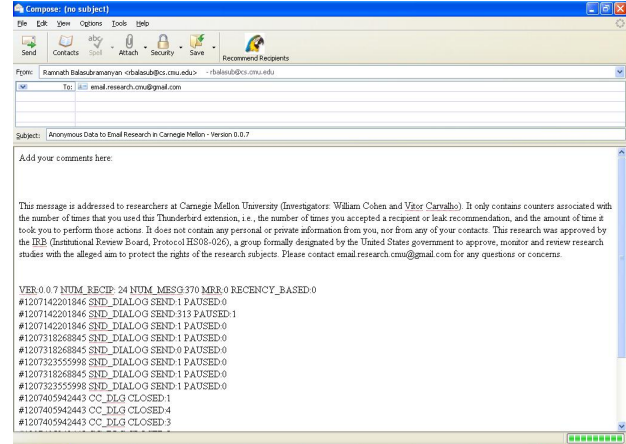
- time elapsed before recipient was added

- the MRR score of recipient added

The user is prompted to send log information every week. If the user acquiesces, a new email compose window is opened up with the log information pre-filled in the content section. Users are also encouraged to send in comments in a designated area in this email. (See Figure 4)

User study

Description

Here we describe a user study based on the CutOnce currently in progress at Carnegie Mellon University. Mozilla Thunderbird users from the Pittsburgh area were recruited on the web and via newsgroups for a four-week long study. These participants were told that the goal was to study how to improve the way people address email messages based on the intelligent addressing techniques (Carvalho and Cohen 2007; 2008).

We required the participants to be Mozilla Thunderbird users, writing email using Thunderbird on a daily basis, and being at least 18 years-old. The recruitment message also indicated that the task would be simple, with minimum or no interruptions at all.

After contacting the study researchers indicating their interest, participants would then be instructed to install CutOnce, train it on their *sent* folder, and keep on using Thunderbird as usual.

After successfully installing and training CutOnce, participants received a message explaining exactly what the extension does and what was expected to happen by the end of their first week using CutOnce. This message contained the instruction:

> Every time you compose a message, we collect statistics (timers and counters only, never private information) of your usage of the extension. In about a week from today, the extension will open a composition window for you, and we expect you to hit "send". This message is addressed to us at

Figure 3: Adoption statistics
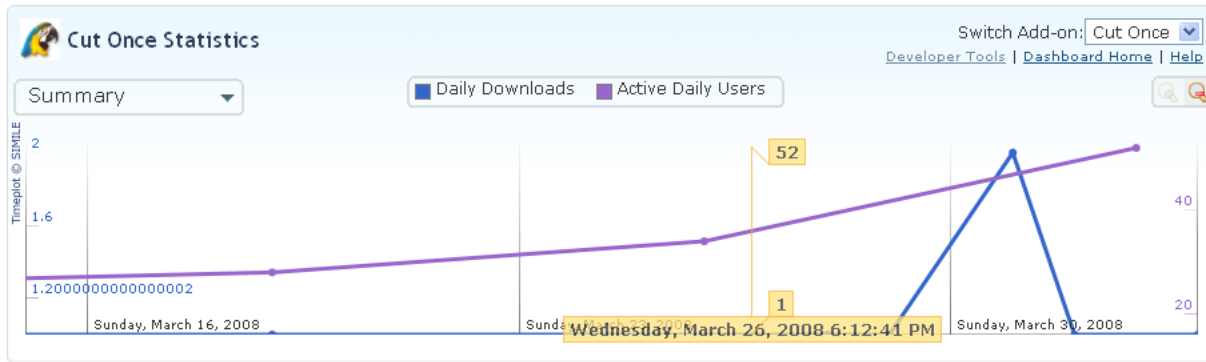
email.research.cmu@gmail.com and will contain the information we are collecting (counters and timers) associated with your usage of the extension. After receiving that message, we'll let you know about the next steps in the study.

Figure 4 illustrates the automatically composed message addressed to the researchers. Participants can read all contents of this message, before eventually allowing it to be sent. After this message is received and analyzed, qualified participants[2] are partially compensated (20% of total compensation) and invited participate in the second phase of this user study.

In the second phase of the study, participants will be compensated with the remaining 80% of the total compensation after three more weeks using CutOnce and completing a final questionnaire.

The questionnaire is concerned with the general CutOnce experience, quality of predictions, reactions of the participant and suggestions for improvement. We believe it can be a valuable source of information for improving CutOnce or designing these features for large scale email systems. The final questionnaire is detailed below.

1. What is your general impression of the extension (5(excellent) 4(good) 3 (neutral) 2(bad) 1(very bad))?

2. Where could it be improved? (open question)

3. How would you grade your overall experience (5(excellent) 4(good) 3 (neutral) 2(bad) 1(very bad))?

4. Would you recommend it? (yes or no)

5. Did the extension catch any email leak? (yes or no) If so, please tell us about it.

6. How often did you use the suggestions? (5(always) 4(frequently) 3(sometimes) 2(rarely) 1(never))?

7. In your opinion, what was the quality of the suggested rank? (5(excellent) 4(good) 3 (neutral) 2(bad) 1(very bad))?

8. Were the suggestions helpful? (5(very helpful) 4(helpful) 3 ("kind of") 2(marginally) 1(not at all))?

9. Were the suggestions annoying? (1(always) 2(frequently) 3(sometimes) 4(rarely) 5(never))?

10. Did the extension change the way you compose messages? (yes or no) If so, please tell us about it.

11. Would you keep on using this extension after this study? (yes or no)

12. Would you you recommend it (to your friends, etc.)? (yes or no)

13. What did you like and dislike the most?

14. Any suggestions or comments?

Analysis and Results

The user study is currently in progress and here we report results obtained from the logging messages obtained so far.

We were expecting that approximately 50% of the logged users would use TFIDF as ranking algorithm, and 50% using the MRR ranking scheme. However, approximately 72% of the users who submitted logging messages so far had their rankings based on TFIDF, while 28% based on MRR[3]. Proportionally also, the majority of the accepted leak predictions and recipient recommendations originated from TFIDF-ranked participants.

Although preliminary, this is an indication that users may be more satisfied with pure textual-based recommendations, or even that recommending the same frequently and/or recently addressed recipients may in fact annoy participants.

By the time this paper was submitted, 23 users had agreed to participate, but only 13 were qualified for the second stage of the study so far. Some users have not completed their first stage, and therefore cannot be considered for the next step on the user study. Other users were not qualified because either they did not have enough recipients (small address book), or they were not using Thunderbird on a daily basis.

---

[2]Qualification for the invitation to participate is based on how long a user has been using Thunderbird, the languages that most messages are written, how frequently he or she uses Thunderbird and general patterns of email use.

---

[3]This can be explained by the fact that anyone can download CutOnce from the web, and participants can easily un-install the extension (or disallow logging messages to be sent) at anytime.

Table 1: User Study Preliminary Results

User	|AB|	|train|	|test|	|leaks|	|recs|
u1	69	2181	15	1	0
u2	64	81	18	0	1
u3	78	1485	4	2	10
u4	12	87	9	1	3
u5	545	9728	25	6	0
u6	16	183	22	2	0
u7	15	516	24	1	2
u8	46	1048	50	1	0
u9	476	9262	4	2	0
u10	38	492	17	1	1
u11	86	1124	22	0	0
u12	47	471	4	0	0
u13	57	731	37	1	0

Preliminary results on these 13 users (u1...u13) are shown in Table 1. In this Table, $|AB|$ is the size of the user's address book (number of email addresses in the address book), $|train|$ is the number of sent messages used for training the leak prediction and recipient recommendation algorithms (see Section Algorithms), $|test|$ is the number of sent messages where predictions were displayed to the user (reported in the logging message), $|leaks|$ is the number of times a user removed a leak predicted by CutOnce from his recipient list, and $|recs|$ is the number of times the user accepted a recipient recommendation provided by CutOnce.

Table 1 suggests that most users utilized leak predictions at least once. Recipient recommendation was also used by 5 qualified participants on the study. Despite most data in Table 1 were collected from the first week of CutOnce usage — when some users may be still experimenting with the extension — we believe these are promising numbers and indicate that both intelligent message addressing techniques can potentially be adopted by a large number of email users.

This adoption will certainly depend on various factors, many not even addressed here, such as user interface, implementation speed and rank quality. Still, we believe the continuation of this user study can make key contributions to better understanding and design of intelligent message addressing systems.

Adoption

The Thunderbird extension developed was submitted to the Mozilla Addons webpage. At the time of writing this paper, there were 52 daily users of the extension (See Figure 3). More statistics are available at https://addons.mozilla.org/en-US/thunderbird/statistics/addon/6392. The official CutOnce website and links to the user study described above can be visited at http://www.cs.cmu.edu/%7Evitor/cutonce/cutOnce.html.

Conclusions

This paper introduced CutOnce, an extension of the open source email client Mozilla Thunderbird that provides email leak prediction as well as recipient recommendations for messages under composition. These techniques are desirable additions to email clients, particularly if the user is susceptible to high-cost errors such as accidentally addressing non-intended recipients (email leaks) or frequently forgetting to address intended recipients.

CutOnce implements several algorithms for these tasks, including as Recency and Frequency baselines, a Rocchio TFIDF classifier and a rank-based data fusion technique (Carvalho and Cohen 2007; 2008). Because it had to be written in Javascript, careful design decisions were necessary to optimize memory and processing resources on client machines.

We also described a user study based on CutOnce hosted at Carnegie Mellon University. Preliminary results indicate that leak prediction and recipient recommendation can potentially be adopted by a large number of email users. Although the various factors affecting the widespread adoption of these functionalities are still under investigation, we believe this study can provide important contributions to intelligent addressing systems.

References

Aslam, J. A., and Montague, M. 2001. Models for metasearch. In *Proceedings of ACM SIGIR*, 276–284.

Byron Dom, Iris Eiron, A. C. Y. Z. 2003. Graph-based ranking algorithms for e-mail expertise analysis. In *Data Mining and Knowledge Discovery Workshop(DMKD2003) in ACM SIGMOD*.

Carvalho, V. R., and Cohen, W. W. 2007. Preventing information leaks in email. In *Proceedings of SIAM International Conference on Data Mining (SDM-07)*.

Carvalho, V. R., and Cohen, W. W. 2008. Ranking users for intelligent message addressing. In *Proceedings of ECIR-2008 (European Conference on Information Retrieval)*.

Craig Macdonald, I. O. 2006. Voting for candidates: Adapting data fusion techniques for an expert search task. In *CIKM*.

Joachims, T. 1997. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of ICML-97*.

Narjs Boufaden, William Elazmeh, Y. M. S. M. N. E.-K. N. J. 2005. Peep - an information extraction base approach for privacy protection in email. In *Conference on Email and Anti-Spam (CEAS2005)*.

Ogilvie, P., and Callan, J. P. 2003. Combining document representation for known item search. In *ACM SIGIR*.

Pal, C., and McCallum, A. 2006. Cc prediction with graphical models. In *Conference on Email and Anti-Spam*.