

# Automatically Classifying Emails into Activities

Mark Dredze  
Department of Computer and Information Science  
University of Pennsylvania  
3330 Walnut St.  
Philadelphia, PA 19104 USA

Tessa Lau  
IBM Almaden Research Center  
650 Harry Road  
San Jose, CA 95120 USA

Nicholas Kushmerick  
Computer Science Department  
University College Dublin  
Belfield, Dublin 4, Ireland

mdredze@cis.upenn.edu, tessalau@us.ibm.com, nick@ucd.ie

## ABSTRACT

Email-based activity management systems promise to give users better tools for managing increasing volumes of email, by organizing email according to a user's activities. Current activity management systems do not automatically classify incoming messages by the activity to which they belong, instead relying on simple heuristics (such as message threads), or asking the user to manually classify incoming messages as belonging to an activity. This paper presents several algorithms for automatically recognizing emails as part of an ongoing activity. Our baseline methods are the use of message reply-to threads to determine activity membership and a naïve Bayes classifier. Our SimSubset and SimOverlap algorithms compare the people involved in an activity against the recipients of each incoming message. Our SimContent algorithm uses IRR (a variant of latent semantic indexing) to classify emails into activities using similarity based on message contents. An empirical evaluation shows that each of these methods provide a significant improvement to the baseline methods. In addition, we show that a combined approach that votes the predictions of the individual methods performs better than each individual method alone.

**Categories and Subject Descriptors:** H.5 [Information Interfaces and Presentation]: Misc

**General Terms:** Algorithms, Experimentation.

**Keywords:** Activity management, email, machine learning, text classification.

## 1. INTRODUCTION

The goal of activity-centric research is to provide tools for people to manage their activities more effectively. We view activities as a representation of collaborative work practice. Activities involve a set of people, who may each play dif-

ferent roles in the activity (e.g., the coordinator vs. participants). Activities often have state (e.g., completed vs. in-progress). They typically have a goal (e.g., producing a report or making a decision), and they might contain formal or informal processes for reaching that goal. Activities may be related to other activities in ways such as composition (one activity being a part of another) or dependency (one activity must be completed before another can proceed). Examples of activities include: organizing a conference, reviewing papers, purchasing equipment, managing candidate interviews, and making design decisions, for example.

Several activity-centric systems have emerged recently [5, 17, 7, 16]. Most of these systems have focused on the cross-application nature of activities, and provide a means to gather together an activity's documents, emails, people, chat transcripts, and related information into a single view, so that when working on the activity, all of the artifacts related to that activity are near at hand.

At the same time, email has become the primary communications mechanism for people to coordinate collaborative work [8, 23], particularly as workplaces are distributed across multiple geographies. Yet despite email's importance, none of the activity systems to date provide a compelling story for integrating email with activity management. Our goal in this research is to provide better tools for people to manage activities more effectively, focusing on email as the medium people use to communicate about activities.

In this paper, we focus on the problem of classifying emails into activities, in order to automatically populate activities with the emails related to them. Our approach leverages two characteristics of activities: the observation that activities connect groups of people together and the observation that activity-related email tends to center around particular topics. In addition, we have found that a combined approach, which votes together the predictions of the base models, can perform better than each individual model alone, resulting in 93% overall accuracy and an F-measure of 0.81 on a dataset of 1146 emails.

Specifically, this paper makes the following contributions:

- The SimSubset and SimOverlap algorithms for email activity classification that compare the people involved in an activity against the recipients of a message;
- The SimContent algorithm for email activity classifica-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IUT'06, January 29–February 1, 2006, Sydney, Australia.  
Copyright 2006 ACM 1-59593-287-9/06/0001 ...\$5.00.

tion based on content similarity using iterative residual rescaling [2], a form of latent semantic indexing [6];

- An empirical evaluation of our algorithms showing that each one performs better than baseline approaches of either message threads or a naïve Bayes classifier; and
- Empirical evidence showing that a simple voting method for combining the results of the individual algorithms performs better than each algorithm alone.

## 2. PRIOR WORK

Classifying email into activities is conceptually similar to the traditional problem of email classification. Most work in email classification has focused either on spam detection [19, 20] or automated foldering [21, 1, 12]. The vast majority of email classification systems have employed text classification techniques such as naïve Bayes, rule learners, and support vector machines. Our work is different from traditional email classification in a number of ways. First, most assume that the set of folders is static and known ahead of time (with the notable exception of SwiftFile [21]), whereas we are investigating the more challenging problem where the set of class labels (activities) grows over time. Second, most email classification systems examine only the words in the message body and headers to predict its folder classification. Since our focus is on activities, which involve features such as a set of people with specific roles to play, rather than folders, which are only collections of messages, we can leverage activity-specific features in order to improve performance. Finally, foldering often assumes every message is placed in a folder, whereas in our data we have found that the vast majority of emails are not part of a managed activity.

A few email classification systems go beyond message content in classifying email. For example, Kiritchenko *et al* [11] mine temporal patterns in email such as the fact that messages to a mailing list all tend to arrive at the same time. Based on their results, we plan to leverage temporal patterns in activity-related email in the future. As another example, emailSift [1] uses subgraph detection to find patterns that characterize all the messages in a folder, to build a model of the folder. We imagine applying their techniques to characterizing the email associated with an activity.

Recent research has investigated challenges in activity-oriented email. Kushmerick *et al* [13] examined email activities that represented structured business processes such as purchasing books online, and reverse-engineered the structure of processes by learning them from representative email transactions. While those techniques worked well for fairly structured activities, this work addresses the more general problem of less-structured activities that are not necessarily generated by formal business processes.

Khoussainov *et al* [10] inferred relationships amongst messages in less-structured activities. However they assume a batch-processing model in which activities are to be discovered within an archive of past messages; in contrast, here we attack the more challenging problem of tracking an evolving set of activities as they unfold. Similarly, others have attempted to discover activities in a user’s inbox using clustering [9, 22]. However, these systems are not dynamic. In contrast, we build and evaluate our models in a way that better approximates real-world, ongoing usage rather than a one time clustering.

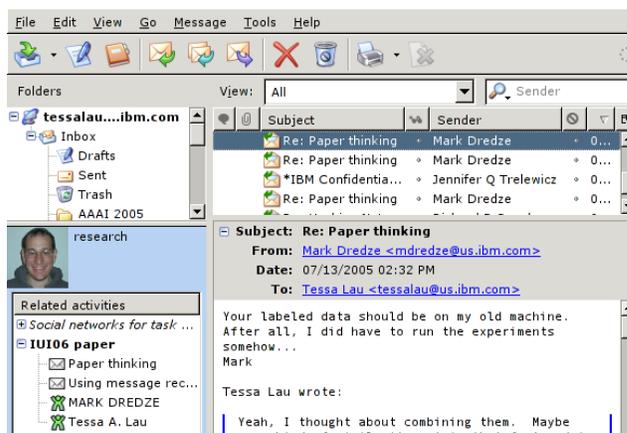


Figure 1: EAM Thunderbird extension displaying the contextual activity pane in the lower left.

Others have explored activity-oriented systems that help users manage their workload more effectively. For instance, Bellotti’s Taskmaster system [3] enhances an email client to function as a task management system. It assumes as a starting point a correspondance of tasks to message threads. In our work we have found that this does not sufficiently capture the entire activity. Muller’s ActivityExplorer system [17] enables people to collaborate around shared artifacts, but does not integrate collaboration with email, despite the observations made by many email researchers [23] that email is used extensively for collaboration and activity management.

Finally, semantic email, in which an email message contains an embedded query allowing a system to process the message as part of a larger task, pursues the same goals as our system [15]. Our projects are complementary in that semantic email seeks to utilize existent meta-data while we attempt to generate meta-data given the emails. Even if semantic email applications are adopted, however, they will still need to interact with legacy systems and this is where a system such as ours could prove useful.

## 3. EMAIL-BASED ACTIVITY MANAGEMENT

In order to motivate our work on email activity classification, in this section we describe several different user interfaces we have prototyped to illustrate the benefit of integrating email with activities.

### 3.1 Activity pane

Figure 1 shows an extension to the Thunderbird email client that displays an activities pane in the lower-left corner. Each activity in this display can contain a set of email messages and a set of people involved in the activity; in the future, this representation could incorporate documents (such as mail attachments), instant messaging transcripts, and other artifacts that may be related to an activity.

The pane shows all the activities that involve both the current user and the sender of the currently-selected email message. The activity list is prioritized by relevance, using the SimOverlap metric described later in this paper; activities whose members are most similar to the set of recipients in the message are displayed at top of the list.

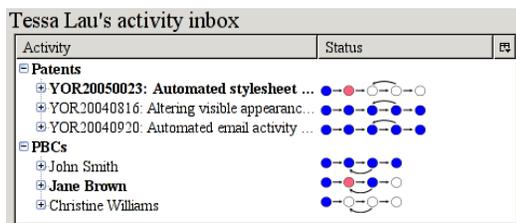


Figure 2: An activity inbox displaying six instances of two activities, along with the process structure of each (bubble diagrams) and the current state of each (light grey dots). Highlighted messages indicate “unread” progress through the activity.

A new activity can be created by right-clicking on an email message. The user is prompted to enter the name and description of the activity, and the set of people involved in the activity; the name and people are automatically extracted from the subject and recipients of the target email. New messages can be associated with an activity by dragging and dropping them onto the activity pane.

Activities are published to a central server, and every participant in an activity will see the activity displayed in her activity pane when relevant. Shared activities are a collaborative artifact: any changes to an activity are made visible to all members of the activity. Thus all participants share responsibility for updating the activity, and all members benefit from the organizational work of other members. We have implemented this interface in Thunderbird, and it is currently in use by a handful of researchers.

While the contextual activity pane automatically displays all activities that may be relevant to a particular selected message, the system does not automatically associate messages with activities. Currently this is only accomplished manually though drag-and-drop. In order to build up a more complete view of the email exchanges that contribute to an activity, users must collectively expend a lot of effort to manually associate messages with activities.

This paper addresses this problem of automatically classifying messages into activities. The algorithms we have developed here could be used to extend this prototype to automatically populate activities with relevant messages.

### 3.2 Activity inbox and Activity manager

One of our future research goals is given a set of messages in an activity, automatically extract the structure of that activity. For example, an activity might include a formalized process, so the structure could include the steps in the process as well as the user’s current state in completing that process. Alternatively, an activity might have a submission deadline, with messages leading up to that deadline acquiring more importance. The vision is to provide activity-centric tools rather than message-centric tools, and let users manage activities as a whole, rather than individual mail messages.

Figure 2 shows a conceptual UI for email-based activity management. Suppose that instead of checking one’s email inbox for new messages, one could check one’s activity inbox to see what has happened on each activity since the last time it was checked. This interface groups messages together by activity, and shows that (for example) there is action on one

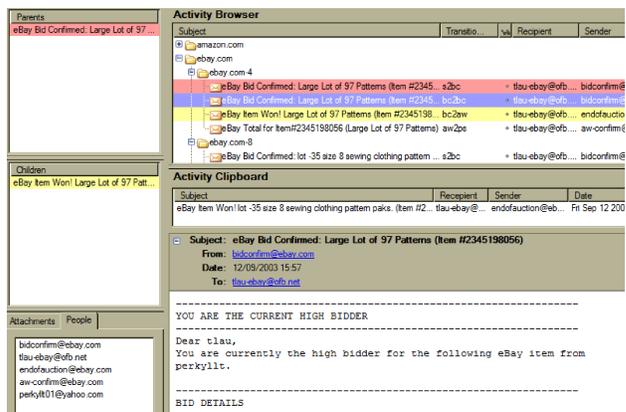


Figure 3: Activity manager with a list of activities and associated emails. The corner pane shows people and attachments related to the current activity.

of the “patent” activities and one of the “PBC” activities. The colored-dot diagrams show a finite-state-machine representation of the structure of each activity; the light grey dot shows where the new message occurred, within the context of the rest of the activity.

Figure 3 shows a third activity-centric email client. It models activities as instances of finite-state *process models* in which messages correspond to transitions between states [13]. This model is general enough to handle both structured computer-human activities (e.g., e-commerce transactions) and informal human-human activities (e.g., meeting scheduling, collaborative document editing). This client seamlessly integrates both ends of this spectrum: the algorithm of [13] is used to automatically label structured activities (“message 1 is an order confirmation”, “message 2 is a delay notification”), and the algorithm of [10] is used to label informal activities with speech acts [4] (“message 1 is a meeting request”, “message 2 is a meeting confirmation and a commitment to send an updated document”).

As these examples show, our larger goal is to infer structure from the unstructured email that make up these activities, and to build tools that make this structure accessible to users to help them manage their activities more effectively. As manually categorizing messages is tedious and infeasible given the volumes of email users receive, automated email activity classification is the first step towards this goal.

## 4. EMAIL ACTIVITY CLASSIFICATION

All of the UIs described in the previous section would benefit from a system that can identify, given a new message, which activity it belongs to. Thus, we define the *email activity classification* problem as follows:

Given a set of existing activities  $\mathcal{A} = A_1, A_2, \dots, A_N$ , the null activity  $\epsilon$ , and a message  $M$ , output a probability distribution  $P$  over activities such that  $\sum_{i \in \mathcal{A} \cup \{\epsilon\}} P(i|M) = 1$ .

Note that this is an incremental learning problem — the set of class labels can (and will) change over time as new activities are created by the user. However, at a particular point in time, the classifier is only expected to be able to predict activities that have previously been created. The intent is that  $\epsilon$  is a distinguished activity label meaning the message is not associated with any activity.

Correct label	Predicted label			
	No activity	Activity A	Activity B	
No activity	(A) Correct	(B) False positive		
Activity A	(C) False negative	(D) Correct	(E) Incorrect	

Table 1: Confusion matrix showing possible predicted outcomes.

## 4.1 Experimental methodology

Our evaluation was designed to simulate a single user’s experience with an activity management system. We assume that messages are received sequentially in temporal order. According to our model of system usage, when a user sees the first message that defines a new activity, she creates a new activity and associates that message with the activity. The user will then expect the system to correctly identify and associate any further activity-related messages. This model leads us to an incremental online learning evaluation.

We evaluate our system using a corpus of 1146 email messages gathered from one user’s email account, of which 149 were labeled with the activity to which they belonged.<sup>1</sup> The corpus spanned a period of 90 days. Since activities were manually labeled, we assume that the labels are correct but not complete; that is, messages that have an activity label have the correct label, but not all messages that belong to an activity may have been labelled as such. A total of 27 distinct activities were identified, ranging in size from one to 38 messages. Examples of activities included organizing an event at a conference, reviewing papers for a workshop, planning a visit to a university, brainstorming about new features to include in an upcoming product, ordering ergonomic equipment, and interviewing a job candidate.

We designed the following experimental methodology. First we sort the messages in temporal order, simulating the order in which the user sees them. We assume that messages not marked as belonging to any activity are labelled with an explicit “no activity” label  $\epsilon$  (i.e., the null activity). For each message, the system predicts which activity this message belongs to, including the null activity. We then incrementally train the classifier with this message, and repeat on each remaining message. One exception to this rule is the first labelled message of every activity; it would be impossible for the system to correctly predict the label on these messages because it has never seen an instance of that activity before. We expect the user to manually identify each new activity as it starts. So in our experiment, we detect the first message of each activity, assume that the user has intentionally labelled it, and do not test on this message.

Depending on how the classifier will be employed in an actual system, the straightforward approach of measuring accuracy does not always represent the quality of a user’s experience with the system. Table 1 shows a confusion matrix that categorizes different errors the system can make.

If the message is not part of an activity, our system can either label the message as no-activity, or as belonging to some activity. The former is correct, the latter is incorrect. If the message actually belongs to activity A, our system

<sup>1</sup>While in the general case messages may belong to multiple activities, we leave that as a topic for future work.

may either incorrectly predict no-activity, correctly predict activity A, or incorrectly predict activity B.

For instance, a recommendation-style UI might present a ranked list of the top 3 predicted activities given a message, and have the user explicitly select one of them. (This is the UI paradigm used in SwiftFile [21].) In this type of UI, the number of false positives (cell B) is largely irrelevant because if the system predicts an activity for a message that the user does not want to associate with an activity, she can ignore the prediction with no harm. On the other hand, false negative errors (cell C) are much more expensive for the user to recover from, because it means she cannot simply associate the message with one of system’s predicted activities, but must instead browse the complete list of activities in order to locate the correct one.

The *recommendation accuracy* of a system measures how useful it is at recommending the correct activity from a list of  $N$  activities. We define the recommendation accuracy of a system  $Acc_{rec,N} = \frac{D}{D+E}$  in terms of Table 1, where  $N$  is the number of recommended activities. For example,  $Acc_{rec,1}$  is the accuracy when the system only recommends a single activity, and  $Acc_{rec,3}$  is the accuracy of a system that recommends three activities and is correct when the correct activity is in the set of three recommendations.

Alternatively, in a UI such as Figure 1 where messages could be automatically added to activities, errors in box (B) might be less important than errors in box (C), under the assumption that a user could easily remove extraneous messages from an activity, but would find it difficult to locate “lost” messages that are not associated with any activity.

In a more extreme case, if a user comes to depend on an activity inbox view (Figure 2) to be notified of new happenings on his activities, then failing to categorize an incoming message as part of one of those activities could be disastrous. On the other hand, miscategorizing a message as belonging to one activity when it actually belongs to another seems to be less critical, because once a user has been notified of the incoming message, he can correct the system’s classification to place it in the correct activity.

The *recognition accuracy* of a system,  $Acc_{recg}$ , measures how useful it is at classifying messages into activities. It depends on the system’s performance on both messages that do belong to an activity, and messages that are not part of an activity. We define recognition accuracy as  $Acc_{recg} = \frac{A+D}{A+B+C+D+E}$ .

However, the accuracy does not necessarily reflect the true usefulness of a system. We also measure the precision  $Prec = \frac{D}{B+D+E}$  and the recall  $Rec = \frac{D}{C+D+E}$  as well as the F-measure  $F = \frac{(\beta^2+1)*Prec*Rec}{\beta^2*Prec+Rec}$  which is a combined precision/recall number that decreases the weight of no-activity examples and emphasizes the system’s performance on positive examples based on the parameter  $\beta$ . In our experiments we set  $\beta = 3$ . We believe that  $F$  is a more representative measure of the quality of a user’s experience with a recognition system since it favors activity completeness over accuracy. This reflects the higher user cost of missing activity messages rather than viewing several extra messages.

We use two baselines for evaluation of our methods. The first is the use of message threads to identify activity membership. For each message marked as being part of an activity, all subsequent replies or follow-ups to that message are also labeled as belonging to that activity. This is the method

used, for example, in Bellotti’s Taskmaster system [3]. We also compare against an updatable naïve Bayes classifier, used by many prior systems for email classification.

In order to verify that our results are not specific to the particular ordering of messages in our corpus, in several of our experiments we report average results over a number of runs with randomized message ordering. There may be content within the sequence of messages in an activity that is time-dependent. Therefore, we randomized the order in which activities occurred in the dataset, while preserving the original ordering of the messages within an activity.

## 5. SIMOVLAP AND SIMSUBSET

Our first algorithm for email activity classification takes advantage of the observation that activities tend to involve specific people with specific roles, which is manifested in the fact that messages are exchanged between different people during the activity. For a message  $M$ , let  $ppl(M)$  denote the set of all people (i.e., distinct email addresses) in the To, From, and CC fields of  $M$ . We generalize  $ppl$  from messages to activities by defining  $ppl(A) = \bigcup_{M \in A} ppl(M)$ .

Given this model, we can define the similarity between an activity  $A$  and a message  $M$ . We have experimented with several different similarity metrics. The first is symmetric, based on the overlap between the people in the activity and the people in the message:

$$SimOverlap(A, M) = \frac{2|ppl(A) \cap ppl(M)|}{|ppl(A)| + |ppl(M)|}$$

This metric is 0 when there is no overlap between the people in the activity and the message, and is 1 when the set of people in the activity is identical to the people in the message. Since the user who received the message is (by definition) a recipient of every message and also belongs to every activity under consideration, we remove the user from both  $ppl(A)$  and  $ppl(M)$  before computing this metric.

A second metric uses the observation that some messages related to an activity may be exchanged with only a subset of members in the activity. This metric calculates the fraction of people in the message that belong to the activity:

$$SimSubset(A, M) = \frac{|ppl(M) \cap ppl(A)|}{|ppl(M)|}$$

As before, we remove the current user from the activity and the message when calculating this metric. This metric is 0 when the message is not addressed to anyone in the activity, and 1 when all the people addressed by the message are part of the same activity. Compared to  $SimOverlap$  however,  $SimSubset$  does not decrease when a message is not addressed to all the people in an activity, which matches our informal observations of activity-oriented messaging. We hypothesize that  $SimSubset$  will perform better than  $SimOverlap$  at email activity classification.

We also experiment with different thresholds, so that if the similarity is not above a certain threshold, the system does not make any prediction. No prediction is considered equivalent to predicting the null activity.

### 5.1 Evaluation: recommendation

We begin by evaluating the person-based algorithms on the recommendation task (predicting the top N activities given a message, for N=1 and N=3, and determining whether the correct activity is in the list). In this task, we assume

Algorithm	$Acc_{rec,1}$	$Acc_{rec,3}$
SimOverlap 0.5	0.61	0.61
SimOverlap 0.7	0.39	0.39
SimOverlap 0.9	0.24	0.24
SimSubset 0.5	0.87	0.91
SimSubset 0.7	0.83	0.86
SimSubset 0.9	0.79	0.83
Threading	0.58	N/A
Naïve Bayes	0.24	0.40

**Table 2: Recommendation results for person-based models, averaged over 100 random message orderings and 14 random orderings for naïve Bayes. Each row represents the use of a different person-based similarity metric, and the corresponding threshold. Threading and naïve Bayes are baseline methods for comparison.**

that the user is only interested in the system’s predictions for messages that truly belong to an activity, and that for no-activity messages, the user will simply ignore the system’s recommendations with no ill effect. Thus we designed the experiment as follows.

For each message that belongs to an activity, use the current model to predict up to N activities. If the correct activity label is in this list, then count this example as correct. Train the activity model with this message, and repeat on the next message that belongs to an activity. The accuracy of the system is the fraction of correct messages out of the total number of activity-labelled messages.

We compare the person-based models against two baseline approaches: threading and a naïve Bayes classifier. The threading approach uses the in-reply-to field in message headers to group together messages in a thread. If a message is a reply to a previous message, then predict the activity to which the previous message belonged, otherwise predict the null activity. As Threading cannot predict more than one activity, we only report results for the N=1 experiment.

We used an updateable (online) naïve Bayes classifier from the Weka [24] toolkit. For features we produced a stemmed version of the message body using a Porter stemmer [18]. In addition, we also provided features from the message header, including the subject and people who sent and received the message. This provided the classifier with the information available to our other methods. We ran each test 14 times using a randomized message ordering as explained above.

Table 2 shows the results of this experiment for the person-based activity models. On this recommendation task, the SimSubset metric clearly produces higher accuracy than the SimOverlap metric, as we expected. Surprisingly, the top-1 and top-3 figures are quite similar. We interpret this as meaning that, at least within this data set, the set of people is a fairly good indicator of the activity, and where any match exists, the person-based activity model will make the correct prediction. However, an investigation of the data reveals that in many cases, the set of people in an activity changes over time, which causes a person-based activity model’s accuracy to decrease. For instance, in one of the activities involving ordering a piece of equipment, the activity started out with just the employee and a manager, but then expanded over time to include the site equipment manager, a procurement specialist, the manufacturer, a reseller, and

Algorithm	$Acc_{recg}$	$Prec$	$Rec$	$F$
SimOverlap 0.5	0.82	0.42	0.60	0.58
SimOverlap 0.7	0.87	0.60	0.39	0.40
SimOverlap 0.9	0.85	0.49	0.24	0.26
SimSubset 0.5	0.78	0.39	0.88	0.79
SimSubset 0.7	0.86	0.52	0.84	0.79
SimSubset 0.9	0.87	0.55	0.80	0.77
Threading	0.93	1.00	0.58	0.60
Naïve Bayes( $M = 60$ )	0.28	0.11	0.29	0.21

**Table 3: Recognition results for person-based models, on the original message ordering. Naïve Bayes results based on 14 randomized runs with oversampling.  $Prec$  is the precision,  $Rec$  is the recall,  $F$  is the F-measure with  $\beta = 3$ .**

an automated workflow system. One limitation of person-based models is that they will not perform well on activities whose membership changes dramatically over time.

## 5.2 Evaluation: recognition

The recognition task is more challenging because the system must distinguish between messages that are part of an activity, and messages that are not. In this experiment, we train the system on all messages, not only messages that are part of an activity. No-activity messages are treated as belonging to the null activity. The model is allowed to predict any previously-seen activity or the null activity. The SimOverlap and SimSubset models output the null activity when there is no matching activity whose score is higher than the specified threshold.

To optimize the naïve Bayes classifier for  $F$ , we oversampled the activity messages. Instead of training on each activity message once, the classifier trained on each activity message  $M$  times while training on no-activity messages only once. The result was a more balanced dataset and an improved  $F$ -measure. We varied  $M$  to produce the optimal oversampling amount;  $F$  increased as  $M$  increased. We selected an optimal  $M = 60$  for our baseline comparison.

Table 3 shows the results for the person-based models on this recognition task. Although according to the  $Acc_{recg}$  metric the SimOverlap methods perform best, examination of the raw numbers reveals that the SimOverlap methods actually perform very poorly at correctly categorizing activity messages into activities (cell D). The high accuracy is primarily due to the fact that they correctly categorize no-activity messages as belonging to the null activity (cell A).

In contrast, the  $F$  metric decreases the importance of no-activity messages and provides a more accurate metric of how useful these models would be in an actual activity recognition system. As expected, SimSubset models perform better using this metric than either SimOverlap, Threading or naïve Bayes. The dramatic difference between naïve Bayes and our methods shows the difference between standard foldering and our email activity classification research.

As SimSubset 0.5’s recall is better than SimOverlap or SimSubset with higher thresholds, we chose it as the representative person-based algorithm for the rest of the paper.

## 6. SIMCONTENT

While some activities consist of distinct sets of people, it is often the case that the same set of people may collabo-

rate on multiple activities. For instance, a team of coworkers may work together on an IUI submission, while simultaneously preparing the next release of their software into product. Alternatively, as discussed above, the people involved in an activity may change over time. Person-based activity recognition approaches will break down in both of these situations. However, we hypothesize that there is still a common thread in the content of these messages. Thus, our second activity classification algorithm incorporates information about the textual content of messages using a variant of latent semantic indexing [2, 6].

Our SimContent algorithm is based on the observation that messages in an activity tend to concern similar topics. Hence, if message  $M_1$  addresses the same topics as another message  $M_2$  and  $M_2$  is associated with activity  $A$ , then it is likely that  $M_1$  is also associated with activity  $A$ .

To determine message similarity, we use an algorithm based on latent semantic indexing (LSI). Unlike traditional bag-of-words models and similarity metrics that are based on how many words two documents share in common, LSI projects documents onto a reduced-dimensionality subspace, and computes similarity as the distance between two document vectors in the reduced subspace. The intent of subspace projection is to capture concepts inherent in similar documents, such that each dimension in the subspace corresponds to a different concept in the document corpus.

We use LSI to classify emails into activities as follows. Assume that the user has labelled all activity-related email received thus far (and that unlabelled messages are implicitly labelled with the null activity). Build an LSI index containing all message documents seen thus far. On receipt of a new incoming message, compute its similarity to all documents in the index, and output a ranked list of documents and their corresponding scores. Apply activity labels to the documents in order to produce a ranked list of activities. Given this ranked list, we formulated a number of metrics for producing a ranked list of activities most similar to a given message:

- **Top N:** output the first N unique activity labels.
- **Weighted score (P, N):** for the P most similar documents, group them by activity label. Add up the scores for all documents in each activity, and output the N activities with the highest score.

In our experiments, the weighted-score metric with  $P = 5$  resulted in the highest performance on the recognition task, so we have used that metric for all further experiments.

One improvement to LSI, iterative residual rescaling (IRR) [2], purports to improve LSI’s performance when the document corpus contains a non-uniform distribution of document topics. This distribution is probably an accurate characterization of activity-oriented email; based on our evaluation dataset, for example, 88% of email does not belong to any activity, whereas the remaining 12% is unevenly distributed over 27 different activities. Thus we hypothesize that IRR will do better than LSI at the recognition task.

## 6.1 Evaluation: recommendation

Table 4 shows the results for the recommendation task for SimContent. Because the LSI and IRR classifiers take significant amounts of time to train, we report results only on the original message ordering of the dataset. Both LSI and IRR perform better than our baselines.

Algorithm	$Acc_{rec,1}$	$Acc_{rec,3}$
LSI, weighted-score(5,3)	0.84	0.92
IRR, weighted-score(5,3)	0.77	0.90
LSI, top 3	0.86	0.92
IRR, top 3	0.86	0.94
Threading	0.58	N/A
Naïve Bayes	0.24	0.40

**Table 4: Recommendation results for SimContent models, using several different metrics to calculate the top scoring activities, evaluated only on the original message ordering. Threading and naïve Bayes are baseline methods for comparison.**

Algorithm	$Acc_{rcg}$	$Prec$	$Rec$	$F$
LSI	0.86	0.52	0.76	0.72
IRR	0.87	0.54	0.76	0.73
Threading	0.93	1.00	0.58	0.60
Naïve Bayes( $M = 60$ )	0.28	0.11	0.29	0.21

**Table 5: Recognition results for SimContent experiments. The weighted-score ( $P=5$ ) metric was used to calculate the most likely activity.**

Surprisingly, IRR does not perform better than LSI. We hypothesize that this is because for the recommendation task, the size of topics is relatively uniform (between 1-38 messages), and thus IRR’s advantages in handling unevenly-sized topics are not relevant here.

## 6.2 Evaluation: recognition

Table 5 shows the results for SimContent on the recognition task, using the weighted-score metric ( $P = 5$ ) for calculating the most likely activity. Again, the results are reported only for the original message ordering.

LSI and IRR produce nearly the same results on this problem. Compared to Threading, LSI and IRR are less precise but have higher recall, indicating that they can more often correctly identify messages as being part of an activity than Threading. This is reflected in the higher F-measure score for LSI and IRR compared to Threading. For an activity management system where it is more important that a message is identified as part of an activity than to have it be correctly classified into the right activity, we believe that the SimContent algorithms are an improvement over the baseline Threading method.

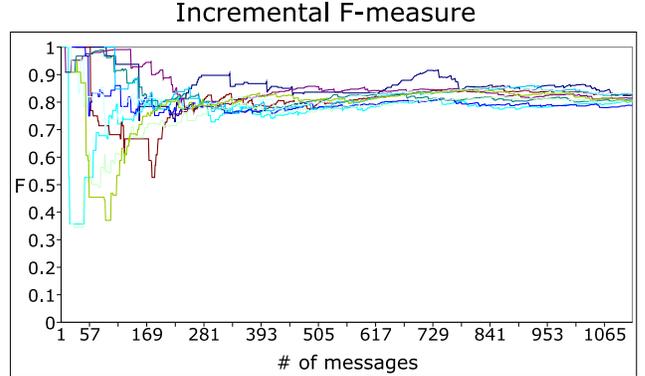
## 7. COMBINED MODEL

While our SimSubset and SimContent methods perform better than the baseline Threading and naïve Bayes methods, we wanted to improve the system’s performance on the recognition task even further. We observed that it was often the case that when two base methods agreed on an activity label, it was likely to be correct. This observation led us to try a simple voting approach: for each message, generate the results of all three base methods. If the method fails to make a prediction, predict the null activity. Give each resulting prediction one vote; the final prediction is the label with the most votes, choosing randomly in case of ties. We selected as our three base learners Threading, SimSubset 0.5 and IRR weighted-score ( $N=5$ ).

Table 6 shows the results of applying this voting method

Algorithm	$Acc_{rcg}$	$Prec$	$Rec$	$F$
Threading	0.93	0.99	0.58	0.60
SimSubset	0.70	0.33	0.86	0.74
SimContent	0.83	0.46	0.74	0.70
Voting	0.93	0.74	0.81	0.81

**Table 6: Results from voting, over 14 randomized message orderings.**



**Figure 4: Results showing incremental F-measure on 14 randomized orderings.**

to the three base learners, over 14 different randomized message orderings. The combined Voting algorithm is able to match the Threading method’s accuracy, increasing the recall to nearly the level of SimSubset, and having a precision somewhat in between SimContent and Threading. Thus overall, Voting receives the highest F-measure score, indicating that its performance is superior to that of any of the individual base methods on the email activity classification problem. Since our methods achieved an accuracy of close to 95% for the recommendation task, we did not attempt to create a combined approach for recommendation as well.

One concern is that the performance of a system would be very high in the early stages, when there is a small number of activities to predict, and then degrade over time as the number of activities increases. However, we have found that despite the difficulty of this problem, our combined approach manages to achieve a fairly consistent performance over the course of the dataset. Figure 4 shows the dynamic behavior of the F-measure for several randomized orderings as a function of the number of training messages.

## 8. CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated two different approaches to the problem of email activity classification, based on the people involved in an activity and the contents of messages in the activity. We have shown empirically that the SimSubset and SimContent algorithms perform better on our dataset than the baseline approach of message threading and a naïve Bayes classifier, and that a combined model that votes together the predictions of all three base learners performs better than any individual learner alone.

Furthermore, our results demonstrate that email activity classification is a very difficult problem. For the recommendation task, a naïve Bayes classifier, a standard approach to classification tasks, is unable to correctly identify the activity with 3 guesses half the time. Threading, the most

common approach, correctly recommends the right activity less than 60% of the time. Such poor performance makes it difficult to implement the interfaces that we envision. In contrast, our methods produce an accurate suggestion 94% of the time. The recognition task proved to be even more challenging since the activity related messages are only a small fraction of the total messages. Where the best baseline methods performed with an F-measure of 0.60, our combined method achieved 0.81. In practice, our system can automatically populate activities with email such that more than 80% of the relevant messages are automatically identified. The accuracy of our methods allows for the development of a useful and accurate activity management system.

There are many directions for future work. Our first priority is to evaluate our algorithms on a wider variety of activity-labelled email to verify that our results generalize to a variety of users and their activities. We also imagine many ways of improving the algorithms discussed here. For example, we plan to investigate the use of more sophisticated metalearning techniques beyond mere voting in order to improve the performance of the overall system. McCallum *et al* [14] associates links in a social network graph with topics, perhaps allowing for richer interaction between our content and person based methods. We also plan to investigate new base learners, based on features such as temporal locality of activities, more sophisticated social network analysis, and the use of text analytics to extract better features from message content.

In the more general sense, this work is just a first step towards a broader vision of activity-based computing. Our ultimate goal is to infer the structure and characteristics of those activities from the messages (and other artifacts) that make up the activity, in order to provide people tools to manage their activities more effectively.

## Acknowledgements

The authors would like to thank Rie Ando, Daniel Avrahami, Catalina Danis, Daniel Egnor, Stephen Farrell, and Wendy Kellogg for insightful discussions about this work. Dredze is supported by a NDSEG fellowship.

## 9. REFERENCES

- [1] Manu Aery and Sharma Chakravarthy. eMailSift: mining-based approaches to email classification. In *SIGIR '04: Proc. of the 27th annual intl. ACM SIGIR conf. on information retrieval*, pages 580–581. ACM Press, 2004.
- [2] Rie Kubota Ando and Lillian Lee. Iterative residual rescaling. In *SIGIR '01: Proc. of the 24th annual intl. ACM SIGIR conf. on information retrieval*, pages 154–162. ACM Press, 2001.
- [3] V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI '03: Proc. of the SIGCHI conf. on Human factors in computing systems*, pages 345–352. ACM Press, 2003.
- [4] W. Cohen, V. Carvalho, and T. Mitchell. Learning to classify email into "speech acts". In *Proc. Conf. Empirical Methods in Natural Language Processing*, 2004.
- [5] Alex Cozzi, Tom Moran, and Clemens Drews. The shared checklist: Reorganizing the user experience around unified activities. In *10th Intl Conf on Human-Computer Interaction (INTERACT 2005)*, Sept. 2005.
- [6] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.
- [7] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. TaskTracer: a desktop environment to support multi-tasking knowledge workers. In *IUI '05: Proc. of 10th intl. conf. on Intelligent User Interfaces*, pages 75–82. ACM Press, 2005.
- [8] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *interactions*, 8(5):30–38, 2001.
- [9] Y. Huang, D. Govindaraju, T. Mitchell, V. Rocha de Carvalho, and W. Cohen. Inferring ongoing activities of workstation users by clustering email. In *Proc. of the 1st Conf. on Email and Anti-Spam*, July 2004.
- [10] R. Khoussainov and N. Kushmerick. Email task management: An iterative relational learning approach. In *Proc. Conf. Email and Anti-Spam*, 2005.
- [11] S. Kiritchenko, S. Matwin, and S. Abu-Hakima. Email classification with temporal features. In *Proceedings of Intelligent Information Systems, New Trends in Intelligent Information Processing and Web Mining (IIPWM) 2004*, pages 523–534. Springer Verlag, 2004.
- [12] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. In *CASCON '01: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, pages 192–201. IBM Press, 2001.
- [13] N. Kushmerick and T. Lau. Automated email activity management: an unsupervised learning approach. In *IUI '05: Proc. of the 10th intl. conf. on Intelligent User Interfaces*, pages 67–74. ACM Press, 2005.
- [14] Andrew McCallum, Andres Corrada-Emmanuel, and Xuerui Wang. Topic and Role Discovery in Social Networks. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, July 2005.
- [15] Luke McDowell, Oren Etzioni, Alon Halevy, and Henry Levy. Semantic email. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 244–254. ACM Press, 2004.
- [16] Thomas P. Moran, Alex Cozzi, and Stephen P. Farrell. Unified Activity Management: Supporting People in eBusiness. *Communications of the ACM*, 2005. To appear.
- [17] M. J. Muller, W. Geyer, B. Brownholtz, E. Wilcox, and D. R. Millen. One-hundred days in an activity-centric collaboration environment based on shared objects. In *CHI '04: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 375–382. ACM Press, 2004.
- [18] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [19] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
- [20] R. Segal, J. Crawford, J. Kephart, and B. Leiba. SpamGuru: An Enterprise Anti-Spam Filtering System. In *Proceedings of the First Conference on Email and Anti-Spam*, July 2004.
- [21] R. Segal and J. Kephart. Incremental Learning in SwiftFile. In *ICML '00: Proc. of the 17th Intl. Conf. on Machine Learning*, pages 863–870, San Francisco, CA, 2000.
- [22] A. Surendran, J. Platt, and E. Renshaw. Automatic discovery of personal topics to organize email. In *Proc. of the 2nd Conf. on Email and Anti-Spam*, July 2005.
- [23] Steve Whittaker and Candace Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 276–283, New York, NY, USA, 1996. ACM Press.
- [24] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd ed. Morgan Kaufmann, 2005.