Recommending Recipients in the Enron Email Corpus

Vitor R. Carvalho
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

William W. Cohen
Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

Email is the most popular communication tool of the internet. In this paper we investigate how email systems can be enhanced to work as *recipient recommendation systems*, i.e., suggesting who recipients of a message might be, while the message is being composed, given its current contents and given its previously-specified recipients. This can be a valuable addition to email clients, particularly in large corporations. It can be used to identify people in an organization that are working in a similar topic or project, or to find people with appropriate expertise or skills. Recipient recommendation can also prevent a user from forgetting to add an important collaborator or manager as recipient, preventing costly misunderstandings and communication delays.

In this paper we present the first study of recipient recommendation in a real large-scale corporate email collection, the Enron Email corpus. We begin by defining the problem as a large multi-class multi-label classification task, where each email can be addressed to multiple recipients in the user's address book (i.e., each class is equivalent to an email address in the address book). We propose various baselines to the problem, along with a classification-based reranking scheme to combine two types of features: textual contents and network information from the email headers. Experiments indicate that the reranking scheme significantly outperforms the baselines, and that the best scheme is accurate enough to be useful in email clients. Results are encouraging also because the proposed solution can be easily implemented in any email client – with no changes in the email server side.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Electronic mail

General Terms

Measurement, Experimentation

Keywords

Email, Text Classification, Recommending Systems

1. INTRODUCTION

One important use of work-related email is negotiating and delegating shared tasks and subtasks, and more generally, communication between task-oriented working groups. Previously we have argued [9] that providing intelligent assistance for this use of email is important because the cost of errors in task management is high: for instance, deadlines can be missed or opportunities wasted because of such errors.

Here we consider an automated technique, called *recipient recommendation*, that is designed to avoid a specific type of high-cost email error: errors that result when a message is not sent to all intended recipients. An example of such an error would be forgetting to CC an important collaborator, or manager, on a message to a working group: such an omission could cause costly misunderstandings and communication delays.

Specifically, *recipient recommendation* involves automatically suggesting email recipients of messages being composed. In this paper we experimentally evaluate techniques that provide a ranked list of email addresses that are likely to be intended as recipients of a given message. We formalize this task as a large-scale multi-class classification problem, and evaluate two baseline classification methods developed in the information retrieval community. The most effective baseline turns an extremely efficient method, which can be applied even to very large message collections with many potential recipients.

We then develop and evaluate extensions to this method which substantially increase performance, while still being efficient enough to be incorporated in most email clients. Evaluating on 36 different users from the Enron Email Corpus [14], these extensions improve performance (as measured by mean average precision) by nearly 25% on average over the baseline. While the effectiveness of the method varies from user to user, our results suggest that it is accurate enough to be useful in current email clients. For instance, on the task of predicting secondary recipients of a message (CC'd and BCC'd addresses) given the primary (TO) recipient, the average recall at rank 10 is 59.82%, i.e., nearly 60% of the intended recipients are listed in the top 10 guesses.

This paper is organized in the following way. In Section 2 we introduce the Enron data and the preprocessing steps utilized in the experiments. In Section 3 we explain the two main ideas of this report. Section 3.1 introduces some baselines techniques and the problem definition; while Section 3.2 presents the classification-based reranking scheme used to accommodate network features to the problem. We comment on the experiments in the Analysis (Section 4). Related and future work is presented in Section 5. We finish with some concluding statements in Section 6.

2. PREPROCESSING OF ENRON CORPUS

Although email is ubiquitous, large, public and realistic email corpora are not easy to find. The limited availability is largely due to privacy issues. For instance, in most US academic institutions, an email collection can only be distributed to researchers if all senders of the collection also provided explicit written consent.

In all experiments of this paper we used the Enron Email Corpus, a large collection of real email messages from managers and employees of the Enron Corporation. This collection was originally made public by the Federal Energy Regulatory Commission during the investigation of the Enron accounting fraud. We used the Enron collection to create a number of simulated user email accounts and address books, as described below, on which we conducted our experiments.

As expected, real email data have several inconsistencies. To help mitigate some of these problems, we used the Enron dataset version compiled by Jitesh and Adibi [19], in which a large number of repeated messages were removed. This version contains 252,759 messages from 151 employees distributed in approximately 3000 folders.

Another particularly important type of inconsistency in the corpus is the fact that a single user may have multiple email addresses. We addressed part of these inconsistencies by mapping between 32 "raw" email address and the normalized email address for some email users. This mapping (author-normalized-author.txt) was produced by Andres Corrada-Emmanuel, and is currently available from the Enron Email webpage [8].

In this paper, we describe two possible settings for the recipient prediction task[1]. The first setting is called the *TO+CC+BCC prediction*, where we attempt to predict all recipients of an email given its message contents. It relates to a scenario where the message is composed, but no recipients have been added to the recipient list. The second setting is called *CC+BCC prediction*, in which message contents as well as the TO-addresses were previously specified, and the problem is recommending additional addresses in the CC and BCC fields of the message.

For each Enron user, we considered two distinct sets of messages: messages sent by the user (*sent collection*) and messages received by the user (*received collection*). The received collection contains all messages in which the user's email address was included in the *TO*, *CC* or *BCC* fields. The sent collection was sorted chronologically and then split into two parts: the oldest messages were placed into *sent_train* and most recent ones into *sent_test*. The final message counts for the 36 target Enron users are shown in Table 1. This Table also shows their Address Book counts ($|AB|$), i.e., the number of different recipients that were addressed in the messages of the *sent_train* collection.

More specifically, *sent_test* collection was selected to contain at least 20 "valid-CC" messages, i.e., at least 20 messages with valid email addresses in both TO and CC (or both TO and BCC) fields. This particular subset of *sent_test*, with approximately 20 "valid-CC" messages, is called *sent_test**. The TO+CC+BCC prediction task will be tested on the entire *sent_test* collection, whereas the CC+BCC prediction task will be tested only in a subset of it, the *sent_test** collection. This split was necessary to guarantee a minimum number of test messages for CC+BCC prediction task.

[1]Assuming at least one of the recipient fields — TO, CC (or Carbon Copy) or BCC (or Blind Carbon Copy) — is valid.

Enron	$\|AB\|$	**received**	**Sent_**		
user			**train**	**test**	**test***
campbell-l	386	901	505	86	21
derrick-j	179	951	539	224	21
dickson-s	36	1053	99	121	20
geaccone-t	147	648	281	159	21
germany-c	520	2435	3585	101	21
giron-d	179	146	591	519	20
grigsby-m	176	1738	758	157	21
hayslett-r	342	1666	759	26	20
horton-s	242	797	341	133	20
hyatt-k	218	1740	520	109	21
hyvl-d	241	1138	615	108	21
kaminski-v	311	1031	1066	153	20
kitchen-l	599	6568	1457	47	20
lavorato-j	106	1230	223	179	20
lokay-m	135	2633	568	76	20
rapp-b	58	295	105	58	21
ward-k	220	1194	803	146	21
bass-e	164	524	1233	406	21
beck-s	1262	3327	1479	112	20
blair-l	330	2004	1062	37	20
cash-m	407	1586	1138	73	20
clair-c	316	2009	1775	52	20
farmer-d	178	1876	587	390	21
fossum-d	320	774	1001	35	20
haedicke-m	496	2282	1049	70	20
jones-t	869	7849	4371	66	21
kean-s	546	6858	2203	75	21
love-p	447	1255	1490	83	21
perlingiere-d	509	1246	2405	144	21
presto-k	344	2577	996	83	21
sager-e	343	2559	1434	90	20
sanders-r	663	2575	1825	173	20
scott-s	720	1336	1413	409	20
shackleton-s	742	7938	4730	67	21
taylor-m	752	9885	2345	176	20
tycholiz-b	93	970	250	259	20
Average	377.6	2377.6	1266.7	144.5	20.5

Table 1: Number of Email Messages in the Different Collections. $|AB|$ is the number of addresses in the Address Book. sent_test* contains only messages having valid addresses in both TO and CC fields.

This particular split was used to simulate a typical scenario in a user's desktop — where the user already has several sent and received messages, and the goal is to predict the recipients of the next sent messages. In order to make the received collection consistent with this, we removed from it all messages that were more recent than the most recent message in *sent_train*. The general time frames of the different email collections are pictured in Figure 1.

We also simulated each user's address book: for each Enron user u, we build an address book set $AB(u)$, which is the list of all recipient addresses in the messages sent by user u. In other words, the address book of a user u contains all email addresses extracted from all recipients fields (TO,CC

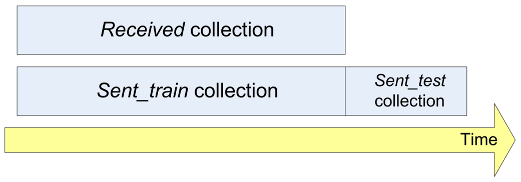


Figure 1: Time frames of different email collections.

and BCC) in the *sent_train* collection of that particular user u.

In all our experiments we represented the content of the messages as a "bag of words", where the counts of all tokens in a message were extracted and taken as feature weights. In this process, a small set of stop words[2] was removed from the email body. In addition, self-addressed messages with no other recipients were also disregarded.

3. METHODS

In this Section we develop different techniques for the CC prediction problem based on the textual contents of the messages. The main idea here is to build a model of "recipient-message" pairs, and then rank the most likely pairs in the test collections according to some measure of confidence. Perfect performance would result in all true recipients being ranked ahead of non-recipients. To evaluate performance, we use well known metrics such as average precision, accuracy and average recall versus rank curves [1]. We assume that the actual recipients of a message are the intended ones.

In order to obtain labeled data, a straightforward procedure for this task is to remove one or more of the "true" recipients from the original message, and then consider these addresses as the classes to be predicted. More precisely, in the CC+BCC prediction task, the classes to be predicted are the "true" email addresses not in the TO field of the message; while the TO+CC+BCC prediction task uses all recipients as "true" classes to be predicted.

It is also important to notice that all methods in this paper are developed "from a single user's perspective". The only information available to construct prediction models are the messages sent to and received by a particular user, and no extra information is assumed. These methods can be naturally extended if information from the email server is taken into consideration, but this also raises several privacy concerns. We will revisit this important issue in Section 4.

3.1 Baselines: Using Textual Content

In this section we develop baseline methods taking advantage of the textual information inside email messages.

3.1.1 TfIdf-Centroid

We start by proposing a technique based on cosine similarity between two TF-IDF (Term Frequency-Inverse Document Frequency) vector-based representations of email messages [18].

During training, for all messages m_u in the *sent_train* collection of user u, we derived the message's TF-IDF vector representation $\vec{m_u}$ from its textual contents and then normalized the vector to length 1.0. We then built, for each recipient r_i in the Address Book $AB(u)$, a *TF-IDF centroid* vector $\overrightarrow{Centroid}(r_i)$ by summing up the normalized TF-IDF vectors of all messages that were sent from u to r_i. In other words, the TF-IDF Centroid is given by:

$$\overrightarrow{Centroid}(r_i) = \sum_{m_u | r_i \in R(m_u)} \vec{m_u}$$

where $R(m_u) = \{r_1, r_2..r_{|R(m_u)|}\}$ is the set of all recipients of message m_u.

When testing, for each test message we computed the cosine similarities between the message's TF-IDF vector representation and the $|AB(u)|$ Centroid vectors. The $|AB(u)|$ recipients are then ranked according to these cosine similarity scores[3]. We refer to this method as *TfIdf-Centroid*. This method is based on the relevance feedback algorithm originally proposed by Rocchio [17], and also known as the TfIdf Classifier [13].

3.1.2 K-Nearest-Neighbors

The second method was based on the K-Nearest Neighbors algorithm described by Yang & Liu [22]. Given a message m_u from user u addressed to a set of recipients $R(m_u) = \{r_1, .., r_{|R(m_u)|}\}$, we found its 30 most similar messages in the training set. The notion of similarity here is also defined as the cosine distance between the text of two normalized TF-IDF vectors. With the top 30 most similar messages selected from the training set, we then computed the weight of each recipient r_i in $|AB(u)|$ according to the sum of similarity scores of the messages (from the top 30 messages) in which r_i was one of the recipients[3]. All recipients are ranked according to this final weight. We refer to this method as *Knn-30*.

3.1.3 Baseline Results

Results using the baseline methods are shown in the first four columns of Tables 2 and 3. Table 2 tabulates the results of the average precision for the TO+CC+BCC prediction task on 36 Enron users. Table 2 shows average precision results on the CC+BCC prediction task for the same 36 users. Additionally, in both Tables we included a random baseline — it shows the performance when ranking of addresses is chosen randomly from the address book, averaged over 20 trials.

The mean values of average precision over the 36 Enron users are also indicated in Tables 2 and 3, along with their standard errors. In both Tables, we performed Wilcoxon Matched-Pairs Signed-Ranks Tests and the following convention was adopted: the symbols $^{+}$ and $^{++}$ indicate that the value on that particular column is statistically significant with $p < 0.05$ and $p < 0.01$, respectively, over the result in the previous column.

Tables 2 and 3 are divided into two user-groups: the top 17 users and the bottom 19 users. In all experiments in this paper, one of these groups was utilized during the development of the methods and feature set, while the other user-group was used as test after all methods were completely developed.

[2]about, all, am, an, and, are, as, at, be, been, but, by, can, cannot, did, do, does, doing, done, for, from, had, has, have, having, if, in, is, it, its, of, on, that, the, they, these, this, those, to, too, want, wants, was, what, which, will, with, would.

[3]For obvious reasons, in the CC+BCC prediction task, a post-processing step removes all TO-addresses from the final rank.

Results clearly indicate that the random baseline performs poorly — not surprisingly, both *TfIdf-Centroid* and *Knn-30* significantly outperform the random baseline in both Tables. In the TO+CC+BCC prediction task, the difference between *TfIdf-Centroid* and *Knn-30* methods is not statistically significant, whereas in the CC+BCC prediction task *Knn-30* seems to outperform *TfIdf-Centroid* significantly (with $p < 0.05$).

Average precision numbers around 0.32 are encouraging given a problem with so many (typically thousands) of classes. In the next section, we will improve the *Knn-30* results by taking advantage of network-like features such as frequency, recency and co-occurrence of email addresses in the training set.

It is worth noticing that, besides these two baseline methods, we also tried two (log)linear classification algorithms to this problem (Logistic Regression and Voted-Perceptron [11]) in a one-versus-all multiclass scheme. We did not observe any noticeable gain over *Knn-30* results in preliminary tests[3]. Moreover, these two linear methods were considerably more expensive to train than the Centroid and *Knn-30* methods because the number of classes to be learned in the one-vs-all scheme was fairly large, i.e., the number of email addresses in the address book. As a matter of fact, the problem of efficient learning when the numbers of classes or concepts abound is an interesting research problem in itself [15].

3.2 Reranking Using Network Information

3.2.1 Crossvalidation

So far we have considered only the textual contents of emails in both recipient prediction tasks. Yet, it is reasonable to consider other features for these problems, such as the number of received messages, number of sent messages, number of times two recipients were copied in the same message, etc. In this Section we describe how these "network" features can be exploited to improve recipient recommendation performance.

In order to combine textual and network features, we used a classification-based reranking scheme. The idea is to perform recipient prediction in two steps. In the first step we calculate the textual similarity scores using a cross-validation procedure in the training set. In the second step, we extract the network features and then we learn a function that combines those with the previously calculated textual scores.

The textual scores are calculated in the following way. We split the training set (sent_train collection) in 10 parts. Using a 10-fold cross-validation procedure: we compute the score for the *knn-30* on 90% of the training data and use it to make predictions in the remaining 10%. Eventually, each training set examples will have, associated with it, a list of email addresses (from the top 30 messages selected by *Knn-30*) and their predicted scores. Now we have, for each message m_u in user u's training set, a score $KnnScore(m_u, r_i)$ associated with each recipient $r_i \in AB(u)$. These scores will be used as features in the second step of the classification procedure.

3.2.2 Network-Based Features

[3]Actually, *Knn-30* typically outperformed both linear models in preliminary tests.

In addition to the textual scores, we used three different sets of network features. The first set is based on the relative frequency of a recipient's email address in the training set. For each recipient we extracted three features: the *normalized sent frequency* (i.e., the number of messages sent to this recipient divided by the total number of messages sent to all other recipients in this user u's training set), the *normalized received frequency* (i.e., the number of messages received from this recipient divided by the total number of messages received from all other users) and the *normalized sum* of the previous two. We refer to these features as *Frequency* features.

In the CC+BCC prediction task only, information in the TO-addresses of a message can be exploited to improve recipient prediction. This second set of network features is defined in terms of the co-occurrence between TO-addresses and CC(and BCC)-addresses in the training set. The intuition behind this feature is that we expect related CC-recipients to co-occur more frequently with the recipients already addressed in the TO field of the email message. Given a message with three TO-recipients to_1, to_2 and to_3, and a given CC candidate address cc_i, let the frequency of co-occurrence between recipients cc_i and to_j be $F(to_j, cc_i)$ (i.e., the number of messages in the training set that had to_j as well as cc_i as recipients). Then, for a given message, the *Relative CC Frequency* (or RCCF) of cc_i with the TO-recipients will be:

$$RCCF(cc_i) = \frac{\sum_j F(to_j, cc_i)}{F(cc_i)}$$

where $F(cc_i)$ is the number of messages sent to address cc_i in the training set.

The second type of co-occurrence-based feature is called *Relative Joint Recipient Frequency* (or RJRF). For a given message with J TO-recipients, $RJRF(cc_i)$ is defined as the percentage from these TO-recipients that ever co-occurred in the training set with cc_i. In other words, the percentage of these TO-recipients that were at least once addressed in the same message as cc_i. Obviously both RJRF and RCCF features are used only when the number of addresses in the TO field (i.e., J) is two or more. We refer to the RJRF and RCCF features as the *Cooccurrence* features.

The last set of network-based features uses the information in the latest messages sent by the user. Similar to the Frequency features above, we extract the normalized sent frequency of all users in the training set. But instead of using the entire training set for the extraction, we only use the last 20, last 50 and last 100 messages. We refer to these features as the *Recency* features.

3.2.3 Reranking

In order to combine the textual scores $KnnScore(m_u, r_i)$ from the 10-fold crossvalidation procedure with the network based features, we used a classification-based reranking scheme. More specifically, we use the confidence of a classifier as ranking score — a scheme also know as Direct Reranking by Classification [12]. This technique is also closely related to the Stacked generalizations proposed by Wolpert [21].

The learning proceeded in the following way. For each training message m_u with J recipients (TO+CC+BCC), we created $|AB(u)|$ new binary examples: J positive examples

Enron user	Random	TfIdf Centroid	Knn-30	Reranked Knn-30 Score			Δ(%) (From +All to Knn-30)
				+Frequency Features only	+Recency Features only	+All Features	
campbell-l	0.037	0.346	0.329	0.348	0.351	0.354	7.599
derrick-j	0.017	0.177	0.234	0.321	0.327	0.318	35.897
dickson-s	0.003	0.313	0.303	0.353	0.339	0.339	11.881
geaccone-t	0.014	0.191	0.220	0.262	0.276	0.263	19.545
germany-c	0.050	0.464	0.537	0.524	0.556	0.518	-3.538
giron-d	0.017	0.142	0.108	0.194	0.195	0.183	69.444
grigsby-m	0.017	0.388	0.467	0.584	0.656	0.606	29.764
hayslett-r	0.033	0.310	0.303	0.345	0.331	0.324	6.931
horton-s	0.023	0.106	0.091	0.118	0.139	0.137	50.549
hyatt-k	0.021	0.269	0.360	0.486	0.475	0.472	31.111
hyvl-d	0.030	0.231	0.218	0.336	0.344	0.336	54.128
kaminski-v	0.057	0.497	0.561	0.674	0.701	0.688	22.638
kitchen-l	0.010	0.429	0.315	0.428	0.440	0.452	43.492
lavorato-j	0.014	0.211	0.260	0.382	0.405	0.378	45.385
lokay-m	0.005	0.854	0.813	0.805	0.810	0.810	-0.369
rapp-b	0.020	0.264	0.297	0.389	0.335	0.328	10.438
ward-k	0.015	0.380	0.444	0.507	0.521	0.514	15.766
bass-e	0.016	0.352	0.407	0.473	0.461	0.478	17.445
beck-s	0.120	0.101	0.237	0.235	0.329	0.314	32.489
blair-l	0.037	0.446	0.485	0.504	0.567	0.568	17.113
cash-m	0.031	0.333	0.237	0.270	0.285	0.287	21.097
clair-c	0.039	0.430	0.338	0.386	0.387	0.385	13.905
farmer-d	0.030	0.427	0.402	0.375	0.406	0.397	-1.244
fossum-d	0.017	0.084	0.058	0.086	0.123	0.121	108.621
haedicke-m	0.033	0.295	0.302	0.448	0.332	0.299	-0.993
jones-t	0.045	0.384	0.398	0.430	0.408	0.403	1.256
kean-s	0.088	0.359	0.366	0.388	0.423	0.421	15.027
love-p	0.054	0.491	0.399	0.388	0.399	0.367	-8.020
perlingiere-d	0.042	0.253	0.339	0.353	0.367	0.371	9.440
presto-k	0.046	0.422	0.295	0.343	0.331	0.295	0.000
sager-e	0.030	0.228	0.184	0.237	0.299	0.295	60.326
sanders-r	0.063	0.268	0.276	0.414	0.425	0.441	59.783
scott-s	0.067	0.455	0.412	0.383	0.531	0.536	30.097
shackleton-s	0.072	0.275	0.373	0.362	0.438	0.331	-11.260
taylor-m	0.073	0.210	0.222	0.290	0.279	0.264	18.919
tycholiz-b	0.009	0.324	0.292	0.313	0.384	0.386	32.192
Mean	0.036	0.325^{++}	0.330	0.381**	0.399**	0.388**	24.079
StdError	0.004	0.024	0.023	0.023	0.023	0.023	4.193

Table 2: TO+CC+BCC Prediction: Average Precision. The symbol $**$ indicates statistical significance significance ($p < 0.01$) over the result in the *Knn-30* column, while $^{++}$ indicates statistical significance ($p < 0.01$) over the values in the immediately previous column.

associated with the true recipients and $|AB(u)| - J$ negative examples with the features associated with all other non-recipients addresses in the Address Book $AB(u)$. Therefore, each message m_u originated $|AB(u)|$ binary examples[4]; and each binary example is related to an email address r_i from the address book. These binary example contains the following features: the score $KnnScore(m_u, r_i)$, Frequency features of r_i, Recency features of r_i and, in case of the CC+BCC prediction task, Co-occurrence features associated with recipient r_i in the message.

By doing this transformation, the recipient prediction tasks became a binary classification problem, where the final ranking score will be determined by the classifier's confidence[5].

[4]In practice, instead of using all $|AB(u)| - J$ negative examples, we used only a random selection of 20% of them. This provided an effective speed up in training time and did not seem to affect performance.

[5]Similar to previous methods, a post-processing step removes all TO-addresses from the final rank in the CC+BCC prediction task.

We used the Voted Perceptron [11] as learning algorithm[6], as an example of a learning method known to be robust and effective in various tasks [6], yet efficient enough to be plausibly embedded in an email client and scale well to very large datasets. In our experiments, it was trained using five passes through the same training data.

3.2.4 Results

Once again, experimental results using the network features are illustrated in Tables 2 and 3. The last four columns of Table 2 display the average precision test results for 36 different Enron users in the TO+CC+BCC prediction task; while the last five columns of Table 3 show the average precision for the CC+BCC preditions.

In both Tables 2 and 3, we performed Wilcoxon Matched-Pairs Signed-Ranks Tests and the following convention was adopted: the symbols $*$ and $**$ indicate values that are statistically significant with $p < 0.05$ and $p < 0.01$, respectively, over the result in the *Knn-30* column.

[6]To be precise, the "average (unnormalized)" version of the algorithm.

Enron user	**Random**	**TfIdf Centroid**	**Knn-30**	**Reranked Knn-30 Score**				Δ(%)
				+Frequency Features only	**+Cooccur Features only**	**+Recency Features only**	**+All Features**	(From +All to Knn-30)
campbell-l	0.036	0.282	0.304	0.305	0.327	0.308	0.310	1.974
derrick-j	0.017	0.262	0.368	0.384	0.408	0.401	0.378	2.717
dickson-s	0.001	0.329	0.319	0.332	0.331	0.324	0.316	-0.940
geaccone-t	0.014	0.149	0.218	0.242	0.247	0.223	0.235	7.798
germany-c	0.032	0.438	0.533	0.519	0.537	0.545	0.524	-1.689
giron-d	0.015	0.111	0.095	0.115	0.109	0.123	0.133	40.000
grigsby-m	0.017	0.331	0.373	0.515	0.521	0.545	0.550	47.453
hayslett-r	0.033	0.147	0.132	0.189	0.183	0.171	0.164	24.242
horton-s	0.020	0.083	0.078	0.087	0.081	0.129	0.090	15.385
hyatt-k	0.026	0.279	0.498	0.689	0.646	0.661	0.595	19.478
hyvl-d	0.020	0.230	0.302	0.340	0.326	0.353	0.370	22.517
kaminski-v	0.238	0.540	0.730	0.705	0.727	0.735	0.751	2.877
kitchen-l	0.057	0.257	0.137	0.201	0.190	0.184	0.206	50.365
lavorato-j	0.010	0.207	0.235	0.264	0.245	0.276	0.242	2.979
lokay-m	0.013	0.770	0.765	0.764	0.770	0.767	0.769	0.523
rapp-b	0.006	0.192	0.168	0.143	0.194	0.173	0.200	19.048
ward-k	0.019	0.393	0.543	0.592	0.575	0.650	0.642	18.232
bass-e	0.015	0.366	0.452	0.446	0.454	0.448	0.432	-4.425
beck-s	0.102	0.118	0.226	0.221	0.167	0.327	0.257	13.717
blair-l	0.031	0.300	0.450	0.454	0.322	0.504	0.391	-13.111
cash-m	0.039	0.269	0.176	0.205	0.225	0.236	0.255	44.886
clair-c	0.030	0.432	0.333	0.330	0.356	0.320	0.345	3.604
farmer-d	0.017	0.367	0.327	0.270	0.276	0.319	0.278	-14.985
fossum-d	0.030	0.083	0.069	0.069	0.101	0.103	0.098	42.029
haedicke-m	0.047	0.261	0.196	0.256	0.183	0.172	0.113	-42.347
jones-t	0.059	0.140	0.282	0.359	0.278	0.333	0.319	13.121
kean-s	0.052	0.268	0.410	0.412	0.483	0.447	0.478	16.585
love-p	0.043	0.481	0.406	0.400	0.431	0.401	0.394	-2.956
perlingiere-d	0.020	0.138	0.334	0.342	0.340	0.355	0.356	6.587
presto-k	0.033	0.321	0.244	0.284	0.271	0.261	0.244	0.000
sager-e	0.033	0.264	0.245	0.306	0.212	0.346	0.351	43.265
sanders-r	0.050	0.209	0.249	0.405	0.294	0.400	0.388	55.823
scott-s	0.069	0.442	0.442	0.331	0.408	0.529	0.390	-11.765
shackleton-s	0.080	0.198	0.334	0.368	0.393	0.426	0.408	22.156
taylor-m	0.034	0.323	0.350	0.342	0.367	0.328	0.325	-7.143
tycholiz-b	0.009	0.355	0.312	0.311	0.349	0.453	0.453	45.192
Mean	0.038	0.287^{++}	0.323^{+}	0.347**	0.342**	0.369**	0.354**	13.422
StdError	0.007	0.024	0.027	0.027	0.028	0.028	0.028	3.652

Table 3: CC+BCC Prediction: Average Precision of 36 Enron Users. The symbol $**$ indicate statistical significance significance ($p < 0.01$) over the result in the *Knn-30* column. The symbols $^{+}$ and $^{++}$ indicate statistical significance with $p < 0.05$ and $p < 0.01$, respectively, over the result in the immediately previous column.

In the TO+CC+BCC prediction task, there are significant improvements when the proposed reranking scheme is combined with the Frequency features. The Recency features significantly improve average precision values over the *Knn-30* baseline. Likewise, a combination of Frequency and Recency features (the *"+All"* column) in the reranking scheme also shows significantly better results over the baseline. Even though the reranking scheme deteriorated the performance for some Enron users, on average results in Table 2 clearly indicate that the proposed method is very effective to predict recipient addresses. The best setting of features obtained almost 0.4 in mean average precision over the 36 Enron users.

A similar observation is valid for the CC+BCC prediction task in Table 3. Statistically significant gains in average precision can be observed when the Frequency features are used in the reranking scheme. Likewise, both Co-occurrence features as well as the Recency features independently provide significant gains in performance over the *Knn-30* baseline. When all three types of features are combined in the reranking scheme, again, significant gains in performance are observed.

In both Tables 2 and 3, the last column shows the relative gain in average precision over the *Knn-30* baseline when all network-based features are used. On average, gains of more than 24% were observed in TO+CC+BCC predictions, and gains of about 13% for the CC+BCC prediction task. A more comprehensive analysis of Tables 2 and 3 results will be presented in Section 4.

The overall performance for the 36 Enron users on both recipient prediction tasks is illustrated in the top-half of Table 4. This table displays the overall results of both TO+CC+BCC and CC+BCC prediction tasks averaged over the 36 Enron users in terms of three metrics: Average Precision, Accuracy[7] and an alternative metric that we will call "Pr(hit top K)".

[7]Accuracy is defined as the percentage of test messages in which the first guess of the learning algorithm was correct, i.e., the percentage of test messages having a "true" recipient on the top of the prediction ranking.

		TfIdf Centroid	**Knn-30**	**Reranked Knn-30 Scores**			
				+Frequency Features only	**+Cooccur Features only**	**+Recency Features only**	**+All Features**
TO+CC+BCC	Avg. Precision	0.325	0.330	0.381	N/A	0.399	0.388
	Accuracy	0.354	0.392	0.378	N/A	0.415	0.383
	Pr(hit top 3)	0.558	0.608	0.591	N/A	0.637	0.611
	Pr(hit top 5)	0.642	0.684	0.680	N/A	0.692	0.692
	Pr(hit top 10)	0.770	0.771	0.789	N/A	0.787	0.785
CC+BCC	Avg. Precision	0.287	0.323	0.347	0.342	0.369	0.354
	Accuracy	0.214	0.272	0.272	0.266	0.298	0.270
	Pr(hit top 3)	0.398	0.451	0.459	0.430	0.481	0.461
	Pr(hit top 5)	0.484	0.539	0.534	0.507	0.556	0.511
	Pr(hit top 10)	0.612	0.626	0.633	0.622	0.659	0.641
TO+CC+BCC (cleaned)	Avg. Precision	0.335	0.340	0.392	N/A	0.411	0.400
	Accuracy	0.364	0.403	0.389	N/A	0.426	0.394
	Pr(hit top 3)	0.574	0.625	0.607	N/A	0.655	0.628
	Pr(hit top 5)	0.660	0.703	0.699	N/A	0.712	0.712
	Pr(hit top 10)	0.791	0.792	0.811	N/A	0.809	0.806
CC+BCC (cleaned)	Avg. Precision	0.313	0.353	0.380	0.371	0.401	0.385
	Accuracy	0.233	0.296	0.296	0.289	0.324	0.294
	Pr(hit top 3)	0.434	0.491	0.500	0.468	0.524	0.501
	Pr(hit top 5)	0.527	0.587	0.581	0.552	0.605	0.556
	Pr(hit top 10)	0.667	0.681	0.689	0.677	0.717	0.698

Table 4: Overall Recipient Prediction Performance.

Pr(hit top K) is the Probability of a 'hit' in the top K entries of the rank. In other words, it is the probability of having "recall at rank K" larger than zero; or the probability of having at least one correct guess in the top K entries of the rank. For instance, Pr(hit top 5) quantifies the percentages of test messages that had at least one correct prediction up to rank 5. We use this metric in this paper to provide some ground for comparison with this previous work, as detailed in Section 5.

3.2.5 *Unseen Recipients*

A closer look in the results revealed that some of test messages consistently extremely low performance values. A careful analysis revealed that these messages contained recipients that were never addressed in the training set. Because these email addresses would not be found in the user's Address Book, they could never be predicted by any algorithm.

In order to circumvent this problem, we redid the experiments with a small modification in the evaluation: we disregarded all test messages in which all recipients were unseen. Therefore, if the test message had at least one recipient in the Address Book, it would not be disregarded. This modification is indicated with the keyword *(cleaned)* in Table 4.

The overall results for both tasks are shown in Table 4. As expected, performance was improved for all methods and feature settings after the test set evaluation is "cleaned". Generally speaking, the same qualitative observations on the top half results of Table 4 can be made about the top bottom.

One particular difference is that the improvements in performance were more pronounced in the CC+BCC prediction task than in the TO+CC+BCC one. This can be explained by the smaller number of test messages in CC+BCC predictions. Disregarding the messages with unseen recipients, the best configuration was able to correctly guess recipients with accuracies of 0.426 and 0.324, respectively, for the TO+CC+BCC and CC+BCC tasks.

Figure 2 shows the Average Recall versus Rank curves for all 36 Enron users. Those curves display values of average recall from rank 1 to rank 1262 (the largest Address Book size). A closer look in the top ten rank positions of the same curves is depicted in Figure 3.

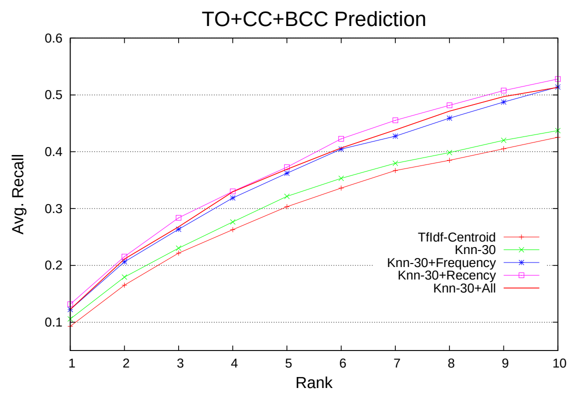
Figures 2 and 3 displays the avgerage recall performance for the TfIdf-Centroid baseline, the *Knn-30* baseline and all possible variants of the the reranking scheme. All plots in those Figures clearly show that the reranking scheme typically outperforms the textual-only baselines. In particular, the difference between the reranking-based methods and the textual baselines is more pronounced in the TO+CC+BCC curves than in the CC+BCC curves.

Another interesting observation from Figures 2 and 3 is the behavior of the *Knn-30* baseline. It consistently outperform the TfIdf baseline in the top of the rank, but then between rank 20 and 30 the *Knn-30* performance seriously deteriorates and its average recall values are considerably worse than the TfIdf baseline. This behavior is explained by the fact that *Knn-30* utilizes only the top 30 most similar training messages for predictions — completely disregarding the other messages.

It is also interesting that the CC+BCC prediction presents higher average recall values than the TO+CC+BCC task from rank 1 to 10. Results were very encouraging, average recalls at rank 10 were nearly 60% for the CC+BCC task and approximately 53% on the TO+CC+BCC one.

4. ANALYSIS

Generally speaking, the reranking scheme was shown to be significantly more effective than any of the textual base-



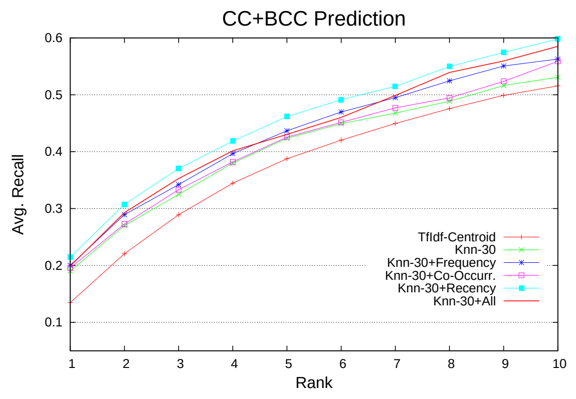


Figure 3: Average Recall vs Rank Curves: From Rank 1 to 10.

lines. Also, the best possible results were not reached by using all possible features. Instead, using Recency features (only) seem to be the most effective setting for this problem, outperforming all other configurations in most of the tests. This indicates how important these features are.

In reality, we used the Recency features as an approximation for the notion of email threads. Threading information is expected to be a very important piece of evidence for this task[8], but unfortunately we could not exploit it directly because the available Enron dataset does not contain standard message threading information.

Our training/test split is not ideal because the later test messages have less accurate estimates of Recency information (as well of other features). Possibly, the best test conditions for this problem would be a setting where, on each test message prediction, the classifier uses information from all messages seen so far, including previous messages in the same email thread. In this paper we approximated this desirable setting using the train/test splits described in Section 2. In future works we plan to implement this desirable setting possibly by taking advantage of fast, and still effective online learning algorithms [6].

[8]The importance of email thread information to CC prediction has also been suggested by Pal & McCallum [16].

With the exception of the Frequency features, no other feature or method in this study exploited information from the received emails (the *received* collection). We found it largely unnecessary because of two major reasons. First, preliminary tests indicated that there was no significant gain in using information from the received set. Secondly, by using all messages and recipients from the received collection, training times and memory requirements of the learning algorithms were substantially increased.

Indeed, one of the main advantages of the proposed methods are their efficiency and applicability in large-scale systems. Both textual baselines, as well as in the Voted Perceptron-based reranking method, are very efficient to train and easy to implement.

5. RELATED WORK

The CC Prediction problem was initially described by Pal & McCallum [16], where Naive Bayes and Factor Graphs models were used to predict recipients. In their short paper, performance was evaluated in terms of what we called "Pr(hit top 5)" performance measure. They report performance values around 44% for the Pr(hit top 5) measure on a collection of one researcher's personal email — however their version of this computation is substantively different from ours, as they assume that all recipients but one are given and the task is to predict the missing recipient, whereas we assume only the primary recipients are known. For comparison, our best system achieves a value of "Pr(hit top 5)" of more than 60% when averaged over the 36 Enron users.

The email recipient prediction problem is closely related to the *Expert Finding* task in Email. The goal of this task is explicitly finding expertise using only the email messages exchanged inside an organization, as described by Dom et al.[10], Balog et al.[2], Campbell et al.[5] and Sihn an Heeren[20]. This area of research has received considerable attention from the TREC community and an expert-finding task has been run under the TREC Enterprise track since 2005.

Though apparently similar, the Expert Finding task and the email recipient prediction task have some fundamental differences. First, the latter is focused on a single email user, while the former is typically focused in an organization or group. The former is explicitly trying to find expertise in (typically) very narrow areas of knowledge, while the latter is not necessarily trying to find expertise — instead, it is trying to recommend users related to an indiscriminate topic.

The recipient prediction task is also related to the *Email Leak Prediction* task [7]. The goal of this task is preventing information leaks by detecting when a message is accidentally addressed to non-desired recipients. In some sense, the recipient prediction task can be seen as the negative counterpart of the email leak prediction task: in the former, we want to find the intended recipients of email messages, whereas in the latter we want to find the unintended recipients or email-leaks.

Another important area of work obviously related to this problem is collaborative filtering or recommending systems [4, 3]. The CC Prediction problem can be seen as a special type of recommendation system where email recipients are recommended based on the current message being composed and the previous messages exchanged among users.

6. CONCLUSION

With the widespread adoption of email, users are increasingly required to handle large numbers of messages in short periods of time. To help users cope with this "email storm", machine learning techniques have been applied to different email-related tasks. In this work we addressed the email recipient prediction problem, that is, recommending recipients of email messages being composed. This task can be a valuable addition to email clients, particularly in large corporations — it can prevent a user from forgetting to add an important collaborator or manager as recipient, preventing costly misunderstandings and communication delays. In addition, it can potentially be used to identify people in an organization that are working (or have worked) in a similar topic or project, or that have an expertise in a specific subject.

We addressed the problem as a multi-class multi-label classification task, in which every email address in the user's address book is a possible class to be assigned to the current message. We started by proposing two different effective techniques as baselines. Then we developed extensions (based on the Voted Perceptron algorithm) over these baselines to combine the two types of features in this problem: textual contents of messages and network information from the email headers.

All evaluations were carried out using 36 different users of the Enron Email Corpus. Experiments clearly showed that the proposed extensions significantly outperform the baselines and that it is accurate enough to become a valuable feature in email clients. For instance, on the task of predicting all email recipients, our methods reached almost 40% of average precision and more than 41% of accuracy.

Another advantage of this approach is that it can be easily implemented in any email client, not requiring changes in the email server side. Indeed, both the baselines, as well as in the Voted Perceptron-based reranking method, are very efficient to train and easy to be implemented in a large-scale systems, especially over an email client that already includes traditional IR search over messages.

7. ACKNOWLEDGMENTS

Thanks to Yiming Yang and Andrew Carlson for helpful comments.

8. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.

[2] K. Balog and M. de Rijke. Finding experts and their details in e-mail corpora. In *Proceedings of the WWW-2006*, pages 1035–1036, 2006.

[3] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.

[4] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[5] C. S. Campbell, P. P. Maglio, A. Cozzi, and B. Dom. Expertise identification using email communications. In *CIKM*, 2003.

[6] V. R. Carvalho and W. W. Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of KDD-2006*, Philadelphia, PA, 2006.

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

[8] W. W. Cohen. *Enron Email Dataset Webpage.* http://www.cs.cmu.edu/ enron/.

[9] W. W. Cohen, V. R. Carvalho, and T. M. Mitchell. Learning to classify email into "speech acts". In *Proceedings of EMNLP 2004*, pages 309–316, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[10] B. Dom, I. Eiron, A. Cozzi, and Y. Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In *Data Mining and Knowledge Discovery Workshop(DMKD2003) in ACM SIGMOD*, 2003.

[11] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[12] H. Ji, C. Rudin, and R. Grishman. Re-ranking algorithms for name tagging. In *HLT/NAACL 06 Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, 2006.

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

[14] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *ECML*, 2004.

[15] O. Madani and W. Greiner. Learning when concepts abound. Technical report, Yahoo!, 2006.

[16] C. Pal and A. McCallum. Cc prediction with graphical models. In *CEAS*, 2006.

[17] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.

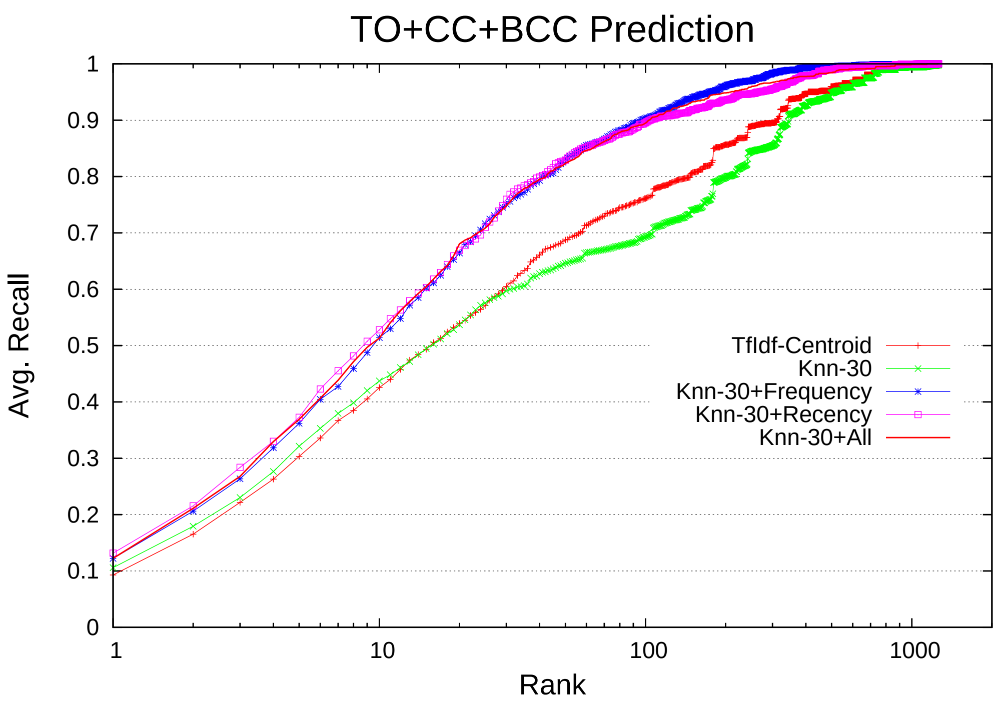
[18] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[19] J. Shetty and J. Adibi. Enron email dataset. Technical report, USC Information Sciences Institute, 2004. Available from http://www.isi.edu/ adibi/Enron/Enron.htm.

[20] W. Sihn and F. Heeren. Expert finding within specified subject areas through analysis of e-mail communication. In *Proceedings of the Euromedia 2001*, 2001.

[21] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[22] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, August 1999.



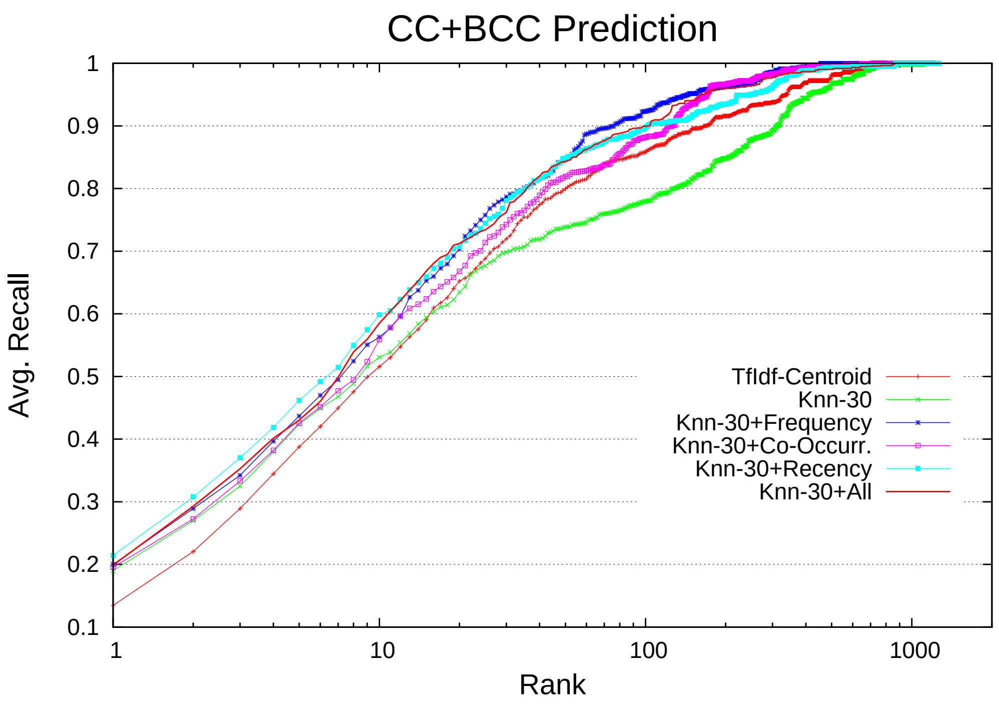


Figure 2: Average Recall vs Rank Curves: Complete Rank.