Ranking Users for Intelligent Message Addressing

Vitor R. Carvalho[1] and William W. Cohen[1,2]

Language Technologies Institute[1] and Machine Learning Department[2]
Carnegie Mellon University, Pittsburgh, PA 15218 USA

Abstract. Finding persons who are knowledgeable on a given topic (i.e. Expert Search) has become an active area of recent research [1–3]. In this paper we investigate the related task of *Intelligent Message Addressing*, i.e., finding persons who are potential recipients of a message under composition given its current contents, its previously-specified recipients or a few initial letters of the intended recipient contact (intelligent auto-completion). We begin by providing quantitative evidence, from a very large corpus, of how frequently email users are subject to message addressing problems. We then propose several techniques for this task, including adaptations of well-known formal models of Expert Search. Surprisingly, a simple model based on the K-Nearest-Neighbors algorithm consistently outperformed all other methods. We also investigated combinations of the proposed methods using fusion techniques, which leaded to significant performance improvements over the baselines models. In auto-completion experiments, the proposed models also outperformed all standard baselines. Overall, the proposed techniques showed ranking performance of more than 0.5 in MRR over 5202 queries from 36 different email users, suggesting intelligent message addressing can be a welcome addition to email.

1 Introduction

Expert search, the task of finding persons who are knowledgeable on a given topic, has been an active area of research recently [1–3]. Here we explore the related task of *Intelligent message addressing*, i.e., finding persons who are potential recipients for a message under composition given its current contents, its previously-specified recipients or a few initial letters of the intended recipient contact. This task can be a valuable addition to email clients, particularly in large corporations, where negotiations are frequently handled via email and the cost of errors in task management is very high. Intelligent Message Addressing can prevent a user from forgetting to add an important collaborator or manager as recipient, preventing costly misunderstandings, communication delays and missed opportunities. Below we present empirical evidence that such errors are very common in the corporate environment. The same technique can also potentially provide assistance in identifying people who have previously worked on specific topics or have relevant skills.

In this paper we formalize two variants of this intelligent email addressing as user-ranking, i.e., finding a ranked list of email addresses that are likely to be intended recipients of a given message. We propose several methods for this task, including classification-based models and adaptations of successful Expert Search formal models [1]. Extensive experiments over 36 different users indicate that a simple model

based on the K-Nearest Neighbors algorithm generally outperforms all other methods, including more refined Expert Search models. In a second set of experiments, we explore how to combine the rankings of different baseline models using rank-based data fusion techniques. Experiments clearly indicate that combined models can significantly outperform all base models on all prediction tasks.

Intelligent message addressing techniques can also be naturally adapted to improve email address auto-completion, i.e., suggesting the most likely addresses based on a few initial letters of the intended contact. Email auto-completion is an extremely useful and popular feature, but in spite of it, little is publicly known on how addresses are ranked in the most popular email clients, and we are not aware of any study comparing different techniques on this particular message addressing problem. In this paper we evaluate several ranking baselines for this problem — including alphabetical, frequency and recency ordering — in a large collection of users. Results clearly indicate that the proposed intelligent addressing models outperform all baselines, significantly improving suggestions in email auto-completion.

Overall we show that intelligent message addressing techniques are able to visibly improve email auto-completion, as well as to provide valuable assistance for users when composing messages. Results suggest it can be a desired addition to most email clients — for instance, on the task of predicting all email recipients, our methods reached 0.47 in MAP and more than 0.5 in MRR. Another advantage is that the best performing methods are computationally efficient and can be easily adapted to large-scale email systems, with no changes in the email server side.

2 Frequency of Message Addressing Problems

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. One of the few datasets available is 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 US Federal Energy Regulatory Commission during the investigation of the Enron accounting fraud. The collection has approximately half a million messages from 150 users' inboxes.

By searching for messages containing the terms *sorry*, *forgot* or *accident* in the entire corpus, and then manually filtering the results[1], we found that at least 9.27% of the users have forgotten to add a desired email recipient in at least one sent message, while at least 20.52% of the users were not included as recipients (even though they were intended recipients) in at least one received message[2]. These surprisingly high numbers clearly suggest that these problems are very common and that email users can benefit from an intelligent message addressing assistant.

[1] Finding messages containing sentences such as "Oops, I forgot to send it to Vince." or "Sorry....missed your name on the cc: list!".

[2] This is a lower bound since not all errors will be noticed by users and not all error-notification emails would be found by our search. A detailed analysis of these results can be found in an extended version of this paper [4].

Table 1. Number of Email Messages in the Different Collections of the 36 selected Enron users. $|AB|$ is the Address Book size, i.e., the number of different recipients that were addressed in the messages of the *sent_train* collection. **Sent_test*** contains only messages having valid addresses in both TO and CC fields. User specific numbers can be found in [4]

	$\|AB\|$	**sent_train**	**sent_test**	**sent_test***
Mean	377.67	1266.69	144.50	20.50
StDev	263.24	1099.05	116.79	0.69
Median	325	1025	109	20
Max	1262	4730	519	23
Min	36	99	26	19

3 Data Preprocessing and Task Definition

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 [5], in which a large number of repeated messages were removed. In addition, some users in the corpus used multiple email addresses. We partially addressed this issue by mapping "raw" email addresses into normalized email addresses for some users[3].

In this paper, we describe two possible settings for the recipient prediction task. The first setting is called the *TO+CC+BCC or primary 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 or secondary prediction*, in which message contents as well as the TO-addresses were previously specified, and the task is to rank additional addresses for the CC and BCC fields of the message.

We randomly selected 36 Enron users, and for each user we chronologically sorted their *sent collection* (i.e., all messages sent by this particular user) and then split the collection in two parts: the oldest messages were placed into *sent_train* and most recent ones into *sent_test*. Message counts statistics for the 36 randomly chosen Enron users are shown in Table 1. 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 main idea is that TO+CC+BCC prediction will be tested on *sent_test*, and the CC+BCC prediction will be tested on the *sent_test** collection (a subset of *sent_test* in which all messages have a valid CC or BCC address).

This chronological split was necessary to guarantee a minimum number of test messages for secondary prediction task and to simulate a typical scenario in a user's desktop — where the user already has several sent messages, and the goal is to predict the recipients of the next sent messages. We also constructed, for each user, an address book set AB which is the set of all recipient addresses in the user's *sent_train* collection. A complete analysis of this data preparation over the different users can be found in an extended version of this paper [4].

[3] This mapping (author-normalized-author.txt) was produced by Andres Corrada-Emmanuel, and is currently available from the Enron Email webpage [6].

4 Models

In this section we describe models and baselines to be used for recipient prediction. In all cases, we followed this terminology. The symbol *ca* refers to *candidate email address* and t refers to *terms* in documents or queries. A document *doc* refers to *documents* in the training set, i.e., email messages previously sent by the same Enron user. A *query* q refers to a message in the test set. Both documents and queries are modeled as distributions over (lowercased) terms found in the "body" of the respective email messages.

We also define other useful functions. The number of times a term t occurs in a query q or a document *doc* is, respectively, $n(t, q)$ or $n(t, doc)$. The *recipient function* $Recip(doc)$ returns the set of all recipients of message *doc*. The *association function* $a(doc, ca)$ returns 1 if and only if *ca* is one of the recipients (TO, CC or BCC) of message *doc*, otherwise it returns zero. $D(ca)$ is defined as the set of training documents in which *ca* is a recipient, i.e, $D(ca) = \{doc|a(doc, ca) = 1\}$.

4.1 Models

Expert Search Model 1 Predicting recipients (candidates) of a message under composition (query) is a very similar task to Expert Search, the task of predicting experts (candidates) associated with a particular topic (query). The analogy works so well that we can easily adapt many recently proposed Expert Search formal models to the task of recipient prediction.

The first recipient prediction model considered here is the *Model 1* proposed for Expert Search by Balog et al. [1]. In this model, the final candidate ranking for each query q is given by the probability of this query being generated by a smoothed candidate language model θ_{ca}[4]:

$$p(q|\theta_{ca}) = \prod_{t \in q} \left\{ (1-\lambda) \left(\sum_{doc} p(t|doc) f(doc, ca) \right) + \lambda p(t) \right\}^{n(t,q)} \quad (1)$$

where λ is the Jelinek-Mercer smoothing parameter, $p(t)$ is the background model probability of term t (maximum likelihood estimates of term in *sent_train* collection), $p(t|doc)$ is the maximum likelihood estimate of the term in the document *doc*, and $f(doc|ca)$ is the document-candidate association function. Similarly to Balog et al. [1], we estimated the document-candidate association functions in two different ways:

$$f(doc, ca) = \begin{cases} \frac{a(doc,ca)}{\sum_{doc'} a(doc',ca)} & \text{, in } \textit{document centric} \text{ (DC) mode;} \\ \frac{a(doc,ca)}{\sum_{ca'} a(doc,ca')} & \text{, in } \textit{user centric} \text{ (UC) mode.} \end{cases} \quad (2)$$

Expert Search Model 2 The second recipient prediction model considered is the *Model 2* proposed by Balog et al. [1]. Basically, the final candidate ranking for each query q is given by the expression:

$$p(q|ca) = \sum_{doc} \left\{ \prod_{t \in q} \left[(1-\lambda) p(t|doc) + \lambda p(t) \right]^{n(t,q)} \right\} f(doc, ca) \quad (3)$$

[4] Please refer to the original reference [1] for further details.

where λ, $p(t|doc)$ and $p(t)$ are defined in the same way as in Section 4.1. Similarly, the two possible views of the document-candidate function $f(doc, ca)$ are defined according to equation 2. Please refer to [1] for further details.

TFIDF Classifier The recipient recommendation problem can naturally be framed as a multi-class classification problem, with each candidate email *ca* representing a class ranked by some notion of classification confidence. Here we propose using the Rocchio algorithm with TFIDF (Term Frequency-Inverse Document Frequency) [7, 8] weights as a classifier for recipient recommendation problems. For each candidate, a centroid vector-based representation is created:

$$\vec{centroid}(ca) = \frac{\alpha}{|D(ca)|} \sum_{doc \in D(ca)} \vec{tfidf}(doc) + \frac{\beta}{|sent\_train| - |D(ca)|} \sum_{doc \notin D(ca)} \vec{tfidf}(doc) \tag{4}$$

where $\vec{tfidf}(doc)$ is the TFIDF vector representation[5]. The final ranking score for each candidate *ca* is produced by computing the cosine similarity between the centroid vector and the TFIDF representation of the query, i.e., $score(ca, q) = \cos\left(\vec{tfidf}(q), \vec{centroid}(ca)\right)$.

K-Nearest Neighbors We also adapted another multi-class classification algorithm, K-Nearest Neighbors as described by Yang & Liu [9], to the recipient prediction problem. Given a query q, the algorithm finds $N(q)$, i.e., the K most similar messages (or neighbors) in the training set. The notion of similarity here is also defined as the cosine distance between the TF-IDF query vector $\vec{tfidf}(q)$ and the TFIDF document vector $\vec{tfidf}(doc)$.

The final ranking is computed as the weighted sum of the query-document similarities (in which *ca* was a recipient):

$$score(ca, q) = \sum_{doc \in N(q)} a(doc, ca) \cos\left(\vec{tfidf}(q), \vec{tfidf}(doc)\right) \tag{5}$$

Other Baselines: Frequency and Recency For comparison, we also implemented two simple baseline models: one based on the frequency of the candidates in the training set, and another based on recently sent messages in the training set. The first method ranks candidates according to the number of messages in the training set in which they were a recipient: in other words, for any query q the *Frequency* model will present the following ranking of candidates:

$$Frequency(ca) = \sum_{doc} a(doc, ca) \tag{6}$$

Compared to *Frequency*, the *Recency* model ranks candidates in a similar way, but attributes more weight to recent messages according to an exponential decay

[5] For each term t in document *doc*, the value $tfidf(t) = log(n(t, doc) + 1) log(\frac{|sent\_train|}{DF(t)})$, where $DF(t)$ is the document frequency of t.

function. In other words, for any query q the *Recency* model will present the following ranking:

$$Recency(ca) = \sum_{doc} a(ca, doc) e^{\left(\frac{-timeRank(doc)}{\beta}\right)} \quad (7)$$

where $timeRank(doc)$ is the rank of doc in a chronologically sorted list of messages in *sent_train* (the most recent message will have rank 1).

4.2 Effect of Threading

Threading information is expected to be a very important piece of evidence for recipient prediction tasks, but unfortunately it cannot be directly exploited here because the Enron dataset does not provide it explicitly. To approximately reconstruct message threads, we used a simple heuristic based on the approach adopted by Klimt & Yang [10].

For each test message q, we construct a set with all messages on the same thread as q (or $MTS(q)$, *Message Thread Set*) by searching for all messages satisfying two conditions. First, the message is among the last P messages sent previous to q. Second, the message must have the same "subject" information[6] as q. While small values of P may not be enough to find all previous messages on the same thread, larger values are expected to introduce more noise in the thread reconstruction process. In preliminary experiments, however, we observed that on average larger values of P did not degrade prediction performance, so only the second condition was imposed on the construction of $MTS(q)$.

In order to exploit thread information in all previously proposed models, we used the following backoff-driven procedure:

$$threaded\_model_i(q) = \begin{cases} MTS\_model(q) & \text{, if } \|MTS(q)\| \geq 1; \\ model_i(q) & \text{, otherwise.} \end{cases}$$

where

$$MTS\_model(q) = \begin{cases} 1.0 \text{ , if } ca \in \bigcup_{d \in MTS(q)} Recip(d); \\ 0.0 \text{ , otherwise.} \end{cases}$$

That is, if q has no previous messages in its thread, predictions from the threaded version of $model_i$ will be made based on the original model $model_i$ (for instance, Frequency, Knn, TFIDF, Expert Model 1, etc.). Otherwise, if the thread of q contains at least one message ($\|MTS(q)\| \geq 1$), predictions are dictated by $MTS\_model(q)$ — a model that assigns weight 1.0 to all recipients found in the messages in $MTS(q)$ and weight 0.0 to all other candidates[7].

5 Results

5.1 Initial results

In this section we present recipient prediction experiments using the models introduced in Section 4. All those models can be naturally applied to both primary and

[6] Or subjects differing only in terms of reply-to (RE:) or forward (FWD:) markers.

[7] In all models of this paper, candidates with the same scores were ranked randomly.

Table 2. MAP recipient prediction results averaged over 36 users. Statistical significance relative to the best model results (in bold) is indicated with the symbols ** ($p < 0.01$) and * ($p < 0.05$).

	T-only	Freq	Rec	M1-dc	M1-uc	M2-dc	M2-uc	TFIDF	Knn
TOCCBCC	0.221**	0.203**	0.260**	0.279**	0.275**	0.279**	0.313**	**0.365**	0.361
CCBCC	0.261**	0.228**	0.309	0.262**	0.272**	0.236**	0.278**	0.301*	**0.332**
TOCCBCC (thread)	N/A	0.331**	0.363**	0.393**	0.385**	0.384**	0.408**	0.440	**0.441**
CCBCC (thread)	N/A	0.379**	0.424*	0.402**	0.407**	0.391**	0.425**	0.429*	**0.459**

Table 3. Recipient prediction results for the best model (Knn) averaged over 36 users.

	MAP	MRR	R-Prec	P@5	P@10
TOCCBCC	0.361	0.440	0.294	0.182	0.135
CCBCC	0.332	0.405	0.266	0.177	0.126
TOCCBCC (threaded)	0.441	0.516	0.398	0.225	0.157
CCBCC (threaded)	0.459	0.540	0.425	0.239	0.156

secondary recipient prediction tasks: the only difference is that, for obvious reasons, in the secondary prediction task, a post-processing step removes all TO-addresses from the final rank, and the test set contains only messages having at least one CC or BCC address.

Similarly to Balog et al. [1], in our experiments both Expert Model 1 and 2 used a smoothing parameter $\lambda = 0.5$. The TFIDF Classifier model had $\alpha = 1$ and $\beta = 0$, creating a centroid of positive examples for each candidate *ca*. We set $K = 30$ in the Knn Model and $\beta = 100$ in the Recency model, values that delivered the best results in preliminary tests.

Table 2 shows Mean Average Precision (MAP) results for all models presented in Section 4. *T-only* refers to *Thread Only* — the prediction based only on detecting threads, i.e., if no thread is detected, candidates are chosen randomly. *Freq* refers to the Frequency model, while *Rec* refers to the Recency model. The symbol *TFIDF* refers to the TFIDF Classifier model. Expert models one and two are referred as *M1* and *M2*, with the candidate-document association indicated by *-uc* (user centric) or *-dc* (document centric). *Thread* refers to models with thread processing (Section 4.2). Two-tailed paired t-test were used for statistical significance tests. Results in Table 2 clearly indicate that the best recipient prediction performance is typically reached by the Knn model, followed by TFIDF. It also reveals that Recency is typically a stronger baseline for this task than the Frequency model. Overall, the expert models M1 and M2 presented statistically significant inferior results when compared to Knn. It is also interesting that the best Expert Search-based model was consistently M2-uc, the same behavior observed by Balog et al. [1] on the TREC-2005 Expert Search task. The use of thread information clearly provided considerable performance gains for all models and tasks. These gains are somewhat expected because, in many cases, email users are simply using the "reply-to" or "reply-all" buttons to select recipients. These improvements are consequently a strong indication that the thread reconstruction algorithm is working reasonably well in this dataset and also the fact that a large proportion of the test messages was found to have a non-empty Message

Thread Set $MTS(q)$. In fact, 29% of the test messages in the primary prediction task had non-empty $MTS(q)$, while the same number for secondary predictions was 35%.

To give a complete picture of the best results, Table 3 shows the Knn performance metrics in terms of other metrics, such as Mean Reciprocal Rank (MRR), R-Precision (R-Prec), and Precision at Rank 5 and 10 (P@5 and P@10) [11]. Overall, the average performance over the 36 Enron users had MRR of more than 0.5, a very good result for such a large prediction task (5202 queries from 36 different users). A closer look in the numbers revealed a much larger variation in performance over different users than over different models. For the primary prediction (threaded), over the 36 users sample, the maximum MAP was 0.76, the minimum was 0.186, with a standard deviation of 0.101.

Based on this variability, we measured the Pearson's correlation coefficient R (quotient of the covariance of the two variables by the product of their standard deviations) between variables that might influence performance. First, the correlation between training set size ($|sent\_train|$) and the number of classes or ranked entities (address book size) is 0.636 — a clear indication that users who send more messages tend to have larger address books. More surprising, perhaps, was the fact that the Pearson's correlation between performance and training set size, as well as the one between performance and Address Book size, was smaller than 0.2 in absolute values — suggesting there is no apparent strong correlation between these variables[8]. One possible explanation is that these two variables contribute inversely to the performance (while recipient prediction is certainly easier with smaller Address Book sizes, it is certainly harder with less training data) and the overall effect is hence weak.

5.2 Combining Evidence with Data Fusion Methods

Ranking results can be potentially improved by combining the results of two or more rankings to produce a better one. One set of the techniques commonly applied to rank combination is *Data Fusion* [12], whose methods have been successfully applied to many areas, including Expert Search [3] and Known Item Search [13].

Because not all ranking scores of the proposed methods in Section 4 are normalized, it is not reasonable to use score-based fusion techniques such as *CombSUM* and *CombMNZ* [3]. Instead, we utilized *Reciprocal Rank* [3] (or RR), a rank-based fusion techniques in which the aggregated score of a document is the sum of inverse ranks of this document in the rankings, i.e., the sum of one over the rank of the document across all rankings.

Table 4 shows experimental results on aggregating recipient recommendation techniques with rank-based Fusion methods. The symbol $\odot$ represents the aggregation operation over different models (all threaded). On each line, the best performing model (in bold face) is selected to be part of the base aggregation in the following line. For instance, the second line displays aggregation results when Knn is combined with the best model in the previous line (TFIDF) and all other three remaining methods. The initial baseline model is threaded Knn. Results clearly show

[8] Similar results were observed for different models on both for primary and secondary predictions.

Table 4. MAP values for model aggregations with Reciprocal Rank. The $*$ and $**$ symbols indicate statistically significant results over the Knn baseline.

Task		Freq	Recency	TFIDF	M2-uc
TOCCBCC	Knn ⊙	0.417**	0.432	**0.457****	0.444
	Knn ⊙ TFIDF ⊙	0.455**	**0.464****	—	0.461**
Baseline: Knn	Knn ⊙ TFIDF ⊙ Rec ⊙	0.451**	—	—	**0.470****
MAP = 0.441	Knn ⊙ TFIDF ⊙ Rec ⊙ M2-uc ⊙	**0.464****	—	—	—
CCBCC	Knn ⊙	0.455	0.470	0.462	**0.474***
	Knn ⊙ M2-uc ⊙	0.476**	**0.491****	0.482**	—
Baseline: Knn	Knn ⊙ M2-uc ⊙ Rec ⊙	0.491**	—	**0.494****	—
MAP = 0.458	Knn ⊙ M2-uc ⊙ Rec⊙ TFIDF ⊙	**0.501****	—	—	—

noticeable performance improvements over the baseline. MAP gains up to 0.042 in the secondary prediction task, and close to 0.03 on primary predictions. In most cases, the gains over the Knn baseline are statistically significant[9]. In a second set of experiments, we used a weighted version of RR, where the weights for each base ranking were determined by the performance obtained by the respective model in a development set. More specifically, this development set was constructed using the 20% most recent messages in *sent_train*, and used as test after training the models in the remaining 80%. Overall, results were statistically significantly better than the Knn baseline, but not statistically significantly better than the unweighted results in Table 4.

5.3 Auto-completion Experiments

Email address auto-completion is the feature in email clients that provide a list of email addresses after the user typed a few initial letters of the intended contact address. Typically email clients allow users the option to turn on or off the auto-completion feature, but rarely are users allowed pick how the suggested addresses should be ranked. In this section we analyze different strategies for email auto-completion ranking.

In order to test different strategies and models for email auto-completion, we used the following experimental procedure. For each query message q, we extracted all its recipient $Recip(q)$, and for each recipient in $Recip(q)$, we extract its V initial letters[10]. Then these V initial letters are used to filter out candidates ranked by the recommendation model. Table 5 presents performance values in terms of MRR* for different values of V and different recommendation models. Notice that for each query q, $|Recip(q)|$ different auto-completion rankings are created, one for

[9] We also experimented with the Borda Fuse [3] aggregation method, but it presented consistently worse results when compared to RR. A similar observation can be drawn from other rank aggregation tasks [3, 13]

[10] In a general case, initial letters from the contact's email address, last name, first name and nickname can be used. We used only email addresses because those were the only contact information consistently available in the Enron corpus; but results can be extended for the general case.

Table 5. Auto-completion Experiments. Performance values for different models and V values. Statistical significance relative to the previous column value is indicated with the symbols ** ($p < 0.01$) and * ($p < 0.05$).

	Primary Prediction (TOCCBCC)							
V	Alpha	Freq	Rec	Knn	Fus	Δ(Knn-Rec)	Δ(Fus-Rec)	Δ(Fus-Knn)
0	0.022	0.274**	0.300**	0.377**	0.394**	25.542%	31.124%	4.447%
1	0.250	0.620**	0.653**	0.690**	0.731**	5.753%	11.893%	5.806%
2	0.557	0.846**	0.857	0.858	0.895**	0.078%	4.412%	4.331%
3	0.737	0.911**	0.923*	0.917	0.942**	-0.683%	2.001%	2.702%
	Secondary Prediction (CCBCC)							
0	0.025	0.329**	0.364**	0.398*	0.436**	9.526%	19.927%	9.496%
1	0.265	0.668**	0.718**	0.717	0.777**	-0.125%	8.289%	8.424%
2	0.549	0.858**	0.875	0.865	0.910**	-1.189%	3.928%	5.178%
3	0.729	0.915**	0.929	0.915	0.946**	-1.558%	1.811%	3.423%

each member of *Recip*(q) (each ranking contains a single relevant recipient and all other recipients in the Address Book who share the same initial letters). MRR* is the mean value of MRR over these rankings.

When $V = 0$, no initial letter of the email contact is known, just like in previous Sections 5.1 and 5.2. As V increases, more is known about the intended recipient and consequently prediction performance becomes better. In addition to the threaded versions of *Knn*, *Recency*(Rec) and *Frequency*(Freq), Table 5 shows results for when recipients are presented in alphabetical order (Alpha). It also contains a model called *All-Fusion* (Fus), displaying results with the aggregated rankings from all models in Table 4 (i.e., using rankings produced by the combinations indicated in the 4^{th} and 8^{th} lines of that Table).

In general, Table 5 indicates that Knn performs slightly better than Recency, which in turn performs better than Frequency. This difference is more noticeable for small values of V — exactly where most email users will benefit the most from auto-completion. When $V = 2$ or $V = 3$ the different between Knn and Recency is not statistically significant. The *All-Fusion* model shows the best auto-completion results overall, significantly outperforming all other models for all values of V. Table 5 also displays the relative performance gains between Knn and Recency, All-Fusion and Recency as well as All-Fusion and Knn.

Compared to any of the other models, auto-completion based only on the alphabetical order presents a rather low performance on both primary and secondary prediction tasks. All other methods can provide significant gains in performance when compared to it. It is surprising that some email clients still provide auto-completion based on this method, given that simple baselines such as Frequency or Recency can provide visible gains in recommendation ranking.

6 Related Work

The email recipient prediction problem is closely related to the *expert search* task. In the former, the task is to retrieve the most likely recipients of a message under

composition, while in the latter the task is to retrieve the most likely experts in a topic specified by a textual query. In fact, it is easy to find similarities between recipient prediction and early expert search work using enterprise email data [14–16]. Recently, interesting models for Expert Search have been motivated by the TREC Enterprise Search, where different types of documents are taken as evidence in the process of finding experts. Because of the similarity between these tasks, many ideas in this paper were motivated by expert search models recently proposed by Balog et al. [1], Fang & Zhai [2] and Macdonald & Ounis [3].

Though relatively similar, expert search and email recipient prediction 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 narrow areas of knowledge (queries with a small number of words), while the latter is not necessarily trying to find expertise — instead, it is trying to recommend users related to one or more indiscriminate "topic(s)" (i.e., a message query that may have up to a few hundred words).

In a related work, Pal & McCallum [17] described what they called the CC Prediction problem. In their short paper, two machine learning models were used to predict email recipients in the personal collection of a single user. However their modeling assumptions is substantively different from ours: they assume that all recipients but one are given and the task is to predict the final missing recipient. Performance was evaluated in terms of the probability of having "recall at rank 5" larger than zero, i.e., the probability of having at least one correct guess in the top 5 entries of the rank. They report performance values around 44% for this metric on their single private email collection. For comparison, our best system achieves 64.8% and 70.6% on the same metric for primary and secondary predictions, respectively, averaged over the 36 different Enron users.

The recipient prediction task is also related to *email leak prediction* [18]. 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.

7 Conclusions

In this work we addressed the the problem of recommending recipients for messages under composition, a task relatively similar to Expert Search. Evidence from a very large real email corpus (Enron corpus) revealed that at least 9% of the users forgot to address an intended recipient at least once, while more than 20% of the users have been accidentally "forgotten" as intended recipients. We proposed several possible models for this task, and evaluated their predictive performance on 36 different users from the Enron corpus. Experiments showed that a simple model based on the K-Nearest Neighbors algorithm generally outperformed all other methods, including frequency or recency based models, and more refined formal models previously proposed for Expert Search.

We also investigate how to combine the rankings of different models using rank-based data fusion techniques, such as sum of Reciprocal Ranks. Experiments clearly indicated that aggregated models can generally outperform all base models, both on primary and secondary recipient prediction tasks. We then applied the proposed ideas to the email auto-completion problem, where the initial letters of the email contact are typed by the user. Results clearly indicate that the proposed models can provide intelligent email auto-completion, outperforming auto-completion based on alphabetical ordering (currently used by some email clients).

Acknowledgments We would like to thank Jonathan Elsas for helpful comments. This material is based upon work supported by the DARPA under Contract Numbers NBCHD030010. Any opinions, findings and conclusions expressed in this material do not necessarily reflect the views of DARPA.

References

1. Balog, K., Azzopardi, L., de Rijke, M.: Formal models for expert finding in enterprise corpora. In: SIGIR 2006. (2006)
2. Fang, H., Zhai, C.: Probabilistic models for expert finding. In: ECIR. (2007) 418–430
3. Macdonald, M.;Ounis, I.: Voting for candidates: Adapting data fusion techniques for an expert search task. In: CIKM, Arlington, USA; November 6-11, 2006. (2006)
4. Carvalho, V.R., Cohen, W.W.: Predicting recipients in the enron email corpus. Technical Report CMU-LTI-07-005 (2007)
5. Shetty, J., Adibi, J.: Enron email dataset. Technical report, USC Information Sciences Institute (2004) Available from http://www.isi.edu/ adibi/Enron/Enron.htm.
6. Cohen, W.W.: Enron Email Dataset Webpage. http://www.cs.cmu.edu/ enron/.
7. Joachims, T.: A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In: Proceedings of the ICML-97. (1997)
8. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing and Management **24**(5) (1988) 513–523
9. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: 22nd Annual International SIGIR. (August 1999) 42–49
10. Klimt, B., Yang, Y.: The enron corpus: A new dataset for email classification research. In: ECML. (2004)
11. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley (1999)
12. Aslam, J.A., Montague, M.: Models for metasearch. In: Proceedings of ACM SIGIR. (2001) 276–284
13. Ogilvie, P., Callan, J.P.: Combining document representation for known item search. In: ACM SIGIR. (2003)
14. Dom, B., Eiron, I., Cozzi, A., Zhang, Y.: Graph-based ranking algorithms for e-mail expertise analysis. In: Data Mining and Knowledge Discovery Workshop(DMKD2003) in ACM SIGMOD. (2003)
15. Campbell, C.S., Maglio, P.P., Cozzi, A., Dom, B.: Expertise identification using email communications. In: CIKM. (2003)
16. Sihn, W., Heeren, F.: Expert finding within specified subject areas through analysis of e-mail communication. In: Proceedings of the Euromedia 2001. (2001)
17. Pal, C., McCallum, A.: Cc prediction with graphical models. In: CEAS. (2006)
18. Carvalho, V.R., Cohen, W.W.: Preventing information leaks in email. In: Proceedings of SIAM International Conference on Data Mining (SDM-07), Minneapolis, MN (2007)