Chapter 4
Recommending Email Recipients

4.1 Introduction

The widespread adoption of email in the workplace is responsible for new issues affecting work management and productivity. One of these problems is that email senders often forget to address one or more intended recipients in their messages. This problem is usually more noticeable in large corporations, where workers interact with peers from various divisions and departments.

To address this problem, in this chapter we proposed several methods of *recipient recommendation*, i.e., the task of recommending persons who are potential recipients for a message under composition given its current contents, its previously-specified recipients, or a few initial letters of the intended recipient contact.

This task can be a valuable addition to email clients, particularly in large corporations, where negotiations are frequently handled via email and the cost of errors in task management can be high. These intelligent message addressing techniques can prevent a user from forgetting to add an important collaborator or manager as recipient, thus preventing costly misunderstandings, communication delays and missed opportunities.

4.2 Evidence of Message Addressing Problems

In order to provide quantitative evidence, using a very large corpus, of how frequently email users are subject to this type of message addressing problem, we focused on the Enron Email collection [Cohen, 2004a] — a large, public and realistic email corpora with approximately half a million messages from 150 Enron employees' inboxes, as previously explained in Section 3.2.

By sampling the Enron collection, one can easily find messages containing sentences such as "Oops, I forgot to send it to Vince. I cc:ed him on this now, though", "Sorry....missed your name on the cc: list!!" or "Sorry, I should have copied you

V.R. Carvalho: Modeling Intention in Email, SCI 349, pp. 53–68, 2011.
springerlink.com

on this". These messages provided strong evidence that, in a previous message, the sender intended to address someone but forgot to include this person as a recipient.

We conducted a thorough search over the entire Enron corpus, looking for messages containing the terms *sorry*, *forgot* or *accident*, and then manually filtered the results in which apologetic messages revealed users forgetting to address intended recipients. We found that at least 9.27% of the 150 Enron users have forgotten to add a desired email recipient in at least one sent message, while at least 20.52% of these users were not included as recipients (even though they were intended recipients) in at least one received message.

These numbers are certainly a lower bound on the real number of messages not going to the intended recipients, since not all errors would be noticed by users and not all apologetic emails would be found by our search. These surprisingly high numbers clearly suggest that such problems can be very common in large organizations, and that email users can benefit from an intelligent message addressing assistant that provides meaningful recipient recommendation.

4.3 Data Preprocessing and Task Definition

We used the Enron Dataset corpus [Cohen, 2004a] to test and validate our methods, and applied the same preprocessing steps described in Section 3.2.

We then utilized 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. This setting relates to the scenario where the message was composed and one or more recipients were already specified, but other recipients can still be added to the recipient list.

We selected the 36 Enron users with the largest number of sent messages, 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 Enron users are shown in Table 4.1. In addition, *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 the 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, as described above.

Table 4.1 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.

	$\|AB\|$	**sent_train**	**sent_test**	**sent_test***
campbell-l	386	505	86	21
derrick-j	179	539	224	21
dickson-s	36	99	121	20
geaccone-t	147	281	159	21
germany-c	520	3585	101	21
giron-d	179	591	519	20
grigsby-m	176	758	157	21
hayslett-r	342	759	26	20
horton-s	242	341	133	20
hyatt-k	218	520	109	21
hyvl-d	241	615	108	21
kaminski-v	311	1066	153	20
kitchen-l	599	1457	47	20
lavorato-j	106	223	179	20
lokay-m	135	568	76	20
rapp-b	58	105	58	21
ward-k	220	803	146	21
bass-e	164	1233	406	21
beck-s	1262	1479	112	20
blair-l	330	1062	37	20
cash-m	407	1138	73	20
clair-c	316	1775	52	20
farmer-d	178	587	390	21
fossum-d	320	1001	35	20
haedicke-m	496	1049	70	20
jones-t	869	4371	66	21
kean-s	546	2203	75	21
love-p	447	1490	83	21
perlingiere-d	509	2405	144	21
presto-k	344	996	83	21
sager-e	343	1434	90	20
sanders-r	663	1825	173	20
scott-s	720	1413	409	20
shackleton-s	742	4730	67	21
taylor-m	752	2345	176	20
tycholiz-b	93	250	259	20
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

4.4 Models

In this section we described models and baselines for recipient prediction. For all models, we used the following terminology. The symbol *ca* refers to *candidate email*

address and *t* refers to *terms* in documents or queries. The symbol *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, i.e., the message under composition. Both documents and queries are modeled as distributions over (lowercased) terms found in the "body" and subject of the respective email messages.

We also defined 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, $D(ca) = \{doc|a(doc,ca) = 1\}$.

4.4.1 Expert Search Model 1

Predicting recipients (candidates) of an email message under composition (query) is a very similar task to *Expert Search*, the task of predicting experts (candidates) on a particular topic (query) [Balog et al., 2006, Fang and Zhai, 2007, Macdonald, 2006]. 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. [2006]. 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}. More specifically, each message term from q is assumed to be independently generated, thus:

$$p(q|\theta_{ca}) = \prod_{t \in q} p(t|\theta_{ca})^{n(t,q)} \tag{4.1}$$

where $p(t|\theta_{ca})$, the probability of term t being generated by a smoothed candidate language model θ_{ca}. The distribution θ_{ca} can be estimated from the empirical probability $p(t|ca)$ smoothed by the background term probabilities from the entire collection $p(t)$ (i.e., maximum likelihood estimates of the terms in the *sent_train* collection):

$$p(t|\theta_{ca}) = (1-\lambda)p(t|ca) + \lambda p(t) \tag{4.2}$$

where λ is the Jelinek-Mercer smoothing parameter. The probability of a term given a candidate $p(t|ca)$ can be estimated as:

$$p(t|ca) = \sum_{doc'} p(t|doc')p(doc'|ca) \tag{4.3}$$

where $p(t|doc)$ is the maximum likelihood estimate of the term in the document doc.

Therefore, the following final model for the probability of a query q given a candidate ca can be estimated as:

$$p(q|\theta_{ca}) = \prod_{t \in q} \left\{ (1-\lambda) \left(\sum_{doc'} p(t|doc') f(doc', ca) \right) + \lambda p(t) \right\}^{n(t,q)} \tag{4.4}$$

As a variation of the method above, one can use Bayes' Rule $p(doc|ca) = \frac{p(ca|doc)p(doc)}{p(ca)} \propto p(ca|doc)p(doc)$ to estimate Equation 4.3 as $p(t|ca) \propto \sum_{doc'} p(t|doc')p(ca|doc')$. Denoting $f(doc, ca)$ as either $p(doc|ca)$ or $p(ca|doc)$, we can then express the following final model for the probability of a query q given a candidate ca:

$$p(q|\theta_{ca}) \propto \prod_{t \in q} \left\{ (1-\lambda) \left(\sum_{doc'} p(t|doc') f(doc', ca) \right) + \lambda p(t) \right\}^{n(t,q)} \tag{4.5}$$

where factor $f(doc, ca)$ is the document-candidate association function which can be estimated in two different ways [Balog et al., 2006]:

$$f(doc, ca) = \begin{cases} p(doc|ca) = \frac{a(doc,ca)}{\sum_{doc'} a(doc',ca)} & \text{, in } \textit{document centric} \text{ (DC) mode;} \\ p(ca|doc) = \frac{a(doc,ca)}{\sum_{ca'} a(doc,ca')} & \text{, in } \textit{user centric} \text{ (UC) mode.} \end{cases} \tag{4.6}$$

This model directly creates a candidate model for each candidate in a user's address book. This model is based on the term information contained on all previous messages sent to the recipients. After representing each candidate as smoothed language models, the recipients ca for a message q under composition are recommended based on their $p(q|\theta_{ca})$ probabilities.

4.4.2 Expert Search Model 2

The second recipient prediction model considered is the *Model 2* proposed by Balog et al. [2006]. The basic difference to *Model 1* is that candidates are not directly modeled. Instead, previous email messages (documents) act as hidden variables between candidates and queries.

By summing over all document, one can express the probability of the query given the candidate in two ways:

$$p(q|ca) = \sum_{doc'} p(q|doc') p(doc'|ca) \tag{4.7}$$

in document-centric mode, or in candidate-centric mode as below:

$$p(q|ca) \propto \sum_{doc'} p(q|doc')p(ca|doc') \tag{4.8}$$

The probability $p(q|doc)$ can be estimated via a smoothed document model $p(q|\theta_{doc})$. More specifically,

$$p(q|\theta_d) = \prod_{t \in q} p(t|\theta_d)^{n(t,q)} \tag{4.9}$$

where the probability of the term t given the document model θ_{doc} can be estimated as:

$$p(t|\theta_{doc}) = (1-\lambda)p(t|doc) + \lambda p(t) \tag{4.10}$$

where λ, $p(t|doc)$ and $p(t)$ are defined in the same way as in Section 4.4.1. We can then express the final candidate ranking for each query q is given by the expression:

$$p(q|ca) = \sum_{doc} \left\{ \prod_{t \in q} [(1-\lambda)p(t|doc) + \lambda p(t)]^{n(t,q)} \right\} f(doc, ca) \tag{4.11}$$

Similar to *Model 1*, the two possible views of the document-candidate function $f(doc, ca)$ are defined according to equation 4.6.

Instead of creating user models, Model 2 directly creates a document model for each message previously sent by the user. After representing document as smoothed language models, the recipients *ca* for a message q under composition are recommended based on their $p(q|ca)$ estimates from equation 4.11.

4.4.3 TFIDF Classifier

The recipient recommendation problem can naturally be framed as a multi-class classification problem, with each candidate address *ca* representing a class ranked by classification confidence. Here we propose using the Rocchio algorithm with TFIDF [Joachims, 1997, Salton and Buckley, 1988] weights as a baseline. For each candidate, a centroid vector-based representation is created:

$$\vec{centroid}(ca) = \frac{\alpha}{|D(ca)|} \sum_{doc \in D(ca)} \vec{tfidf}(doc) + \frac{\beta}{|sent\_train| - |D(ca)|} \sum_{doc \notin D(ca)} \vec{tfidf}(doc) \tag{4.12}$$

where $\vec{tfidf}(doc)$ is the TFIDF vector representation of message *doc*. More specifically, for each term t in message *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.

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., $score(ca,q) = cosine\left(\vec{tfidf}(q), \vec{centroid}(ca)\right)$.

4.4.4 K-Nearest Neighbors

We also adapted another multi-class classification algorithm, K-Nearest Neighbors as described by Yang & Liu [Yang and Liu, 1999], to the recipient prediction problem. Given a query q, the algorithm finds the set $N(q)$, 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):

$$score(ca,q) = \sum_{doc \in N(q)} a(doc,ca)\, cosine\left(\vec{tfidf}(q), \vec{tfidf}(doc)\right) \tag{4.13}$$

4.4.5 Other Baselines: Frequency and Recency

For comparison, we also implemented two even simpler 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:

$$frequency(ca) = \sum_{doc} a(doc,ca) \tag{4.14}$$

Compared to *Frequency*, the *Recency* model ranks candidates in a similar way, but attributes more weight to recent messages according to an exponential decay function. In other words, for any query q the *Recency* model will present the following ranking:

$$recency(ca) = \sum_{doc} a(ca,doc)\, e^{\left(\frac{-timeRank(doc)}{\tau}\right)} \tag{4.15}$$

where $timeRank(doc)$ is the rank of doc in a chronologically sorted list of messages in *sent_train*[1]. In the experiments below the parameter τ in Equation 4.15 was set to 100, thus emphasizing the 100 most recent messages.

[1] The most recent message has rank 1, the second most recent message has rank 2, and so on.

4.4.6 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 and Yang [2004].

For each test message q, we construct a set with all messages on the same thread as q (henceforth $MTS(q)$, *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[2] 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[3].

4.5 Results

4.5.1 Initial Results

In this section we present recipient prediction experiments using the models introduced in Section 4.4. All those models can be naturally applied to both primary and 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.

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

[3] In all models, candidates with the same scores were ranked randomly.

Similarly to Balog et al. [2006], in our experiments both Expert Model 1 and 2 used a smoothing parameter $\lambda = 0.5$. The TFIDF Classifier model had $\beta = 0$, creating a centroid of positive examples for each candidate *ca*. We set $K = 30$ in the Knn Model and $\tau = 100$ in the Recency model, the values that delivered the best results in preliminary tests for six users.

Table 4.2 shows Mean Average Precision (MAP) [Baeza-Yates and Ribeiro-Neto, 1999] results for all models presented in Section 4.4. *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 4.4.6). Two-tailed paired t-test were used for statistical significance tests.

Table 4.2 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 * ($p < 0.05$).

	TOCBCC	**CCBCC**	**TOCCBCC (thread)**	**CCBCC (thread)**
T-only	0.221**	0.261**	N/A	N/A
Freq	0.203**	0.228**	0.331**	0.379**
Rec	0.260**	0.309	0.363**	0.424*
M1-dc	0.279**	0.262**	0.393**	0.402**
M1-uc	0.275**	0.272**	0.385**	0.407**
M2-dc	0.279**	0.236**	0.384**	0.391**
M2-uc	0.313**	0.278**	0.408**	0.425**
TFIDF	**0.365**	0.301*	0.44	0.429*
Knn	0.361	**0.332**	**0.441**	**0.459**

Results in Table 4.2 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 inferior results when compared to Knn, and the difference was statistically significant. It is also interesting that the best Expert Search-based model was consistently M2-uc, the same behavior observed by Balog et al. [2006] 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 4.3 shows the Knn performance metrics in terms of other common ranking metrics, such as Mean Reciprocal Rank (MRR), R-Precision (R-Prec), and Precision at Rank 5 and 10 (P@5 and P@10) [Baeza-Yates and Ribeiro-Neto, 1999]. 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, as attested by Table 4.4. 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.

Table 4.3 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

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[4]. 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.

4.5.2 Rank Aggregation

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* [Aslam and Montague, 2001].These methods have been successfully applied in many areas, including Expert Search [Macdonald, 2006] and Known Item Search [Ogilvie and Callan, 2003].

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

Table 4.4 MAP values using the Knn baseline for all 36 Enron users.

Enron user	TOCCBCC	CCBCC	TOCC(threaded)	CCBCC(threaded)
campbel	0.263	0.319	0.336	0.444
derrick	0.228	0.515	0.379	0.503
dickson	0.390	0.348	0.463	0.476
geaccone	0.549	0.244	0.533	0.351
germany	0.362	0.587	0.349	0.578
giron	0.403	0.112	0.488	0.254
grigsby	0.377	0.372	0.523	0.728
hayslett	0.291	0.138	0.419	0.307
horton	0.293	0.103	0.334	0.151
hyatt	0.462	0.506	0.508	0.639
hyvl	0.444	0.314	0.459	0.329
kaminski	0.765	0.692	0.759	0.703
kitchen	0.366	0.149	0.523	0.581
lavorato	0.356	0.258	0.347	0.246
lokay	0.450	0.770	0.523	0.812
rapp	0.300	0.140	0.425	0.377
ward	0.356	0.561	0.433	0.695
bass	0.468	0.581	0.507	0.616
beck	0.295	0.196	0.357	0.297
blair	0.457	0.437	0.513	0.499
cash	0.301	0.165	0.357	0.226
clair	0.352	0.325	0.404	0.332
farmer	0.442	0.362	0.512	0.417
fossum	0.063	0.067	0.186	0.198
haedicke	0.273	0.237	0.387	0.433
jones	0.370	0.276	0.419	0.314
kean	0.287	0.383	0.397	0.526
love	0.398	0.431	0.511	0.674
perlingiere	0.335	0.235	0.433	0.552
presto	0.419	0.296	0.574	0.530
sager	0.227	0.16	0.286	0.314
sanders	0.286	0.248	0.332	0.416
scott	0.484	0.483	0.558	0.553
shackleton	0.261	0.290	0.445	0.507
taylor	0.287	0.369	0.418	0.424
tycholiz	0.352	0.298	0.490	0.516
Mean	0.361	0.332	0.441	0.459

Because not all ranking scores of the proposed methods in Section 4.4 are normalized, it is not reasonable to use score-based fusion techniques such as *CombSUM* and *CombMNZ* [Macdonald, 2006]. Instead, we utilized *Reciprocal Rank* [Macdonald, 2006] (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 4.5 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 $\odot$	0.417**	0.432	**0.457****	0.444
	Knn $\odot$ TFIDF $\odot$	0.455**	**0.464****	—	0.461**
Baseline: Knn	Knn $\odot$ TFIDF $\odot$ Rec $\odot$	0.451**	—	—	**0.470****
MAP = 0.441	Knn $\odot$ TFIDF $\odot$ Rec $\odot$ M2-uc $\odot$	**0.464****	—	—	—
CCBCC	Knn $\odot$	0.455	0.470	0.462	**0.474***
	Knn $\odot$ M2-uc $\odot$	0.476**	**0.491****	0.482**	—
Baseline: Knn	Knn $\odot$ M2-uc $\odot$ Rec $\odot$	0.491**	—	**0.494****	—
MAP = 0.458	Knn $\odot$ M2-uc $\odot$ Rec$\odot$ TFIDF $\odot$	**0.501****	—	—	—

Table 4.5 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). For instance, in the TOCCBCC task, the aggregation of Knn and Freq (Knn $\odot$ Freq) rankings produced a final ranking with MAP of 0.417. 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 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[5].

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.

4.5.3 Email Auto-completion

Email address auto-completion is the feature in email clients that provides 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

[5] We also experimented with the Borda Fuse [Macdonald, 2006] aggregation method, but it presented consistently worse results when compared to RR. A similar observation can be drawn from other rank aggregation tasks [Macdonald, 2006, Ogilvie and Callan, 2003].

should be ranked. In this section we analyze different strategies for email auto-completion ranking.

Email auto-completion is essentially an email recipient recommendation task in which the user provides the initial characters (or some key characters) of the recipient's name or address. Therefore, the same ranking models and strategies previously utilized in Section 4.4 can naturally be adapted to email auto-completion.

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[6]. Then these V initial letters are used to filter out candidates ranked by the recommendation model.

Table 4.6 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 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, and the task is the same as the original recipient recommendation from Sections 4.5.1 and 4.5.2. 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 4.6 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 4.5 (i.e., using rankings produced by the combinations indicated in the 4^{th} and 8^{th} lines of that Table).

In general, Table 4.6 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 4.6 also displays the relative performance gains between Knn and Recency, All-Fusion and Recency as well as All-Fusion and Knn. Auto-completion performance numbers for larger values of V are illustrated in Figures 4.1 and 4.2.

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 provided significant gains in performance when compared to it.

[6] 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 4.6 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%

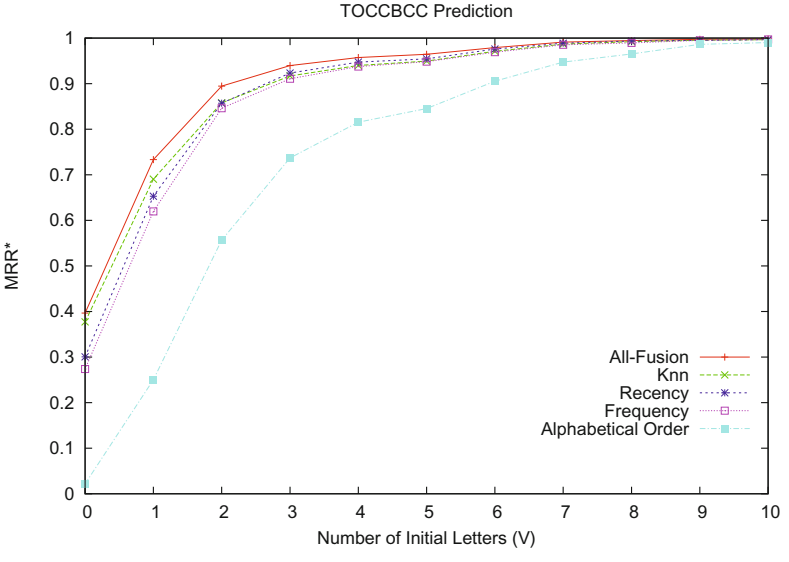

Fig. 4.1 Auto-completion performance on the TOCCBCC task for different number of initial letters.

4.6 Discussion and Related Work

We addressed the problem of recommending recipients for messages under composition. Evidence from a very large work-related real email 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

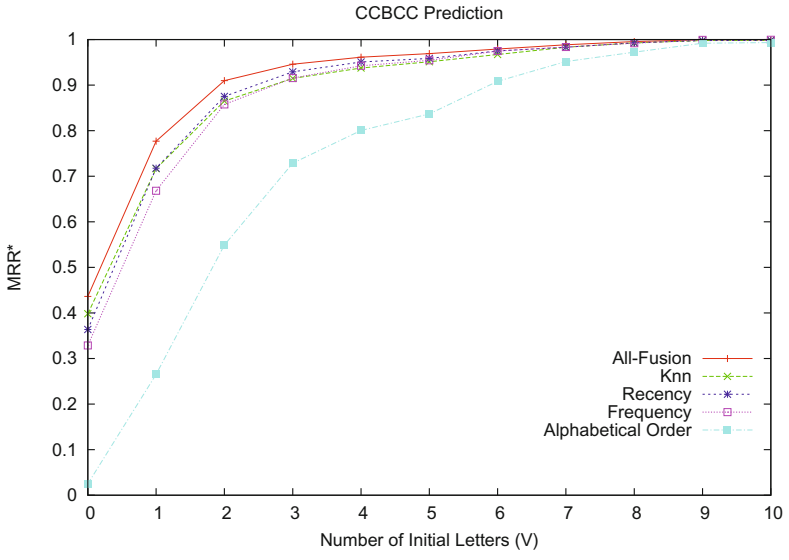

Fig. 4.2 Auto-completion performance on the CCBCC task for different number of initial letters.

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 investigated 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 generally outperform all base models, both on primary and secondary recipient 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. We evaluated 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 for 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.

The email recipient prediction problem is 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 on 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 [Campbell et al., 2003, Dom et al., 2003, Sihn and Heeren, 2001]. 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 of the presented ideas were motivated by recently proposed expert search models [Balog et al., 2006, Fang and Zhai, 2007, Macdonald, 2006].

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 a message "query" that may have up to a few hundred words.

In a related work, Pal and McCallum [2006] 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 a 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. Only two of the Enron users presented values smaller than 44% in this metric for primary predictions, and only one Enron user on secondary predictions.