Email Recipient Prediction Using Reverse Chronologically Arranged Implicit Groups

Akash Desai
Department of Computer Science and Engineering
The LNM Institute of Information Technology
Jaipur, India 302031
akash.desai89@gmail.com

Subrat Kumar Dash
Department of Computer Science and Engineering
The LNM Institute of Information Technology
Jaipur, India 302031
subrat.dash@gmail.com

***Abstract*—Although social networking has significantly influenced online communication, email still has managed to retain its importance. There are number of techniques proposed in past by researchers for recipient prediction/suggestion. Most of them are complex to implement and takes good amount of computation time. The major factor behind higher time complexity and space complexity is the prediction models these methods use. These days mobile device applications are being widely used for emailing and thus appropriate techniques should be found considering constraints of mobile devices. Keeping this in view our research focuses on proposing prediction model, which takes very less computational efforts to be maintained. Apart from this, existing methods focus on maximizing number of intended recipients in one prediction cycle. In this paper, we also propose a different way of looking at the problem, by targeting 1 intended recipient in each iteration. For this, we introduce hit rate as a good measurement technique to measure the effectiveness of recipient prediction algorithm. We also present a flaw in the compiled version of Enron data set, and show some novel analysis on Enron data set which will help immensely in creating efficient recipient prediction algorithm.**

Keywords*—mail, recipient prediction, Enron dataset, iterative prediction methodmail, recipient prediction, Enron dataset, iterative prediction methode

I. Introduction

Email is considered to be a formal mode of communication, as compared to other online communication modes. A recent report [10] gives some insights into the extent of usage of email as a communication mode. According to the report, currently the average number of business emails are about 100.5 billion, sent/received per day, by 929 million users. The consumer emails are about 82.4 billion, sent/received per day by 2,970 million users. These numbers describe the popularity of email, as a communication mode. Moreover email, like any other online communication method, is used for group communication. So, it is important that whoever sends message through email includes all the intended recipients. Carvalho and Cohen [5] have mentioned that, they found 9.27% of the users, available in Enron corpus, had atleast 1 message where they forgot to add desired recipients. And out of all recipients who are present in the corpus 20.52% of recipients were not included at least once, when they were intended. Such cases cause the delay in information delivery and if it happens in a corporate world, it can cost a fortune to companies. This supports the requirement of email recipient prediction algorithm. Nowadays people use applications on mobile devices for emailing. If we look at the numbers on google play store, Yahoomail and Gmail applications have been rated by around 0.85 million people. These applications does not provide the recipient prediction feature. If we look at email clients for computers, like *Thunderbird* and Microsoft Office *Outlook*, they also need some recipient prediction technique which should run efficiently on limited resources.

There are already several techniques available in the literature for the recipient prediction. All the methods that are available so far can be divided into three major categories, namely, meta-data based methods, content-based methods and mixed approach. Methods based on first category look at the frequency, recency, direction, etc. features (meta-data) of an email. The second category looks at the content of the message subject and message body for prediction purpose. Third category either tries to combine the results, retrieved from different methods belonging to both of the previous categories, and gives the final ranking, or tries to create an ensemble method. In this paper, we propose a novel meta-data based method.

Existing prediction methods try to get as many number of correct intended recipients as possible in one prediction cycle and their aim is to get very high precision-recall. By a prediction cycle we mean all processes that take place for giving a single suggestion list. It is yet far away from being possible that all the intended recipients will be suggested correctly by any of the algorithms in one prediction cycle, everytime. Let's assume, the user will get all correct predictions, even in that case, only one of them will be picked by the user at a time. Keeping this in mind we propose a different approach for the problem of email recipient prediction.

In our approach, we try to get at least one relevant suggestion at the end of a prediction cycle. Suppose we have a seed based suggestion approach like [11], and we start suggesting recipients from seed of size 2. There must be at least one intended recipient among all the suggestions, which the user can select and that will act as a third member of the seed, using which prediction can be done again. Here, seed means a set of recipients given by user in '*To*' field, while composing an email. Because we are no more trying to get all the intended recipients in one cycle we should not measure the effectiveness of the method using precision-recall but by some other goodness measure. We show in this study a relevant goodness measure for our approach.

In Section II, we highlight a flaw of the compiled version of Enron data set [13] followed by our data pre-processing process on this data set to overcome the said flaw. In Section III, we provide our analysis on Enron data set in detail. Based on this analysis, we propose a novel method for email recipient prediction in Section IV. Section V contains evaluation methods and experimental results of our proposed method. Section VI describes briefly about related work and we conclude the paper in Section VII.

II. Data Pre-processing

For implementation purpose we require an email corpus which should be real-time as well as publicly available. Although emails are very common, we have very limited options when it comes to availability of email corpus. Following are some available email corpora for studies related to email:

The Enron Corpus:
: The Enron data set [8] is the most popular email corpus among researchers. This data set is real-time and still it is publicly available, and is the reason for its popularity. The raw Enron corpus contains 619,446 messages sent by 158 employees of Enron.

The W3C Corpus:
: It was derived for evaluation of content-based email search, in the TREC Entreprise Track[1]. It was crawled from *w3c.org* in 2004; This collection was distributed to the TREC Enterprise participants by the National Institute of Standards and Technology (NIST). Corpus contains 174,311 messages.For QA topic relevance W3C data had been annotated for use in TREC Enterprice 2005 and 2006.

The CSIRO Corpus:
: TREC Enterprise started using similar data as of W3C from CSIRO, which is the Australian organization, subsequent to TREC 2006.

PW Calo Corpus:
: PW Calo Cropus [6] was result of role playing exercise by the participants at SRI during the CALO project. This corpus consists of total 222 emails. Currently this corpus is not freely available.

We used mysql version of compiled Enron email dataset by Shetty and Adibi [13] for experimentation of our method. Before using it for experimentation, first we tried to study the compiled data set itself. During our study we found that, some of the emails are repeating in the *messages* table of the database. The reason was, same messages were available under different folders. We removed such repeating messages using date-time and sender attributes. Because it is real-time data created by humans, there cannot be two messages sent by same user, addressed to same recipients having the same timestamp. We observed that, it was influencing the result significantly, when we tried to compare results, retrieved on both compiled data set and data set after removing repeating messages. The reason is, two same messages with same recipient-list were coming consecutively in compiled data set and this situation was giving 100% precision. It also reduced number of messages sent by individual users significantly. For example, in original version 6272 messages are sent from *Jeff Dasovich*, out of which only 5393 messages were distinct.

For training and testing purpose we ignore the recipient-lists with size less than 3 and greater than 25. Smaller recipient lists will not be useful because minimum size of a seed is 2. Very large recipient-list covers more recipients, who in general might not be related, and can be considered as an outlier. There are two possible ways in which such lists can influence the results. If such list comes in training data set, recipients who are not related to each other come together in such list and might lead to wrong predictions. If it comes in test data set, probability of getting true positive increases, which leads to increased precision.

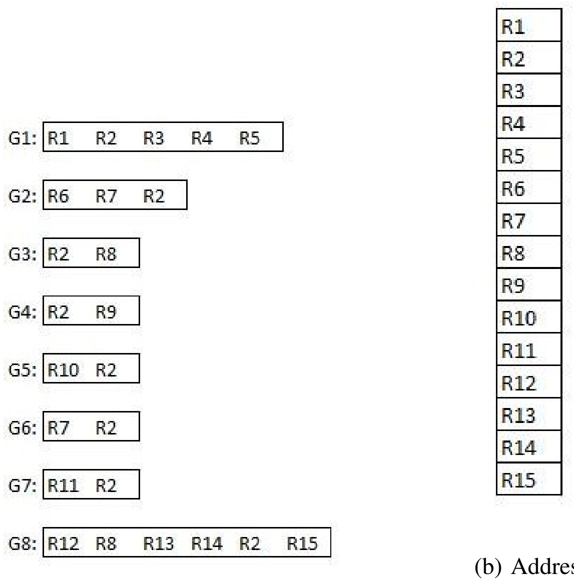
III. Our Analysis of Enron Dataset

Our work is based on following idea. If from the user's address-book we randomly select two or more recipients, how many distinct groups may have them as a subset. For example, if one has two hypothetical friends or colleagues A and B, how many distinct groups one may have with $\{A,B\}$ as a subset. Intuitively, the distinct groups which contains same two or more recipients as a subset must be very less. Later we show how we used this idea.

To find a strong evidence for our idea, we conducted a test on Enron data set to find average number of groups which intersect at given seed. A group is a collection of recipients in a message. It is same as the hyper-edge mentioned in [11]. A seed is a set of any recipients given by a user based on which further recipients need to be suggested. A seed of size n refers to a seed having n number of recipients. For all 151 users of Enron data set, we calculated all the distinct groups. Then we created addressbook for each of the 151 users.

For an example, Figure 1a shows distinct groups observed for a hypothetical user. Using this, as shown in Figure 1b, addressbook is generated for that user. For seed of size 2 and seed of size 3, we used all combinations of size 2 and size 3 of all the users from addressbook respectively. After that we tried to find number of intersecting groups for a given seed. To find average intersecting groups we ignored all those cases which returned 0, because it indicates that such combination is not present in any group. For the example given in Figure 1, if we take $\{R2,R7\}$ as a seed, there are two groups G2 and G6 which contain this seed. If we consider the seed as $\{R1,R7\}$, it is not present in any of the groups, so we ignore it. We observed that for seed of size 2 average intersecting groups are 1.37 for 145 users, and for the remaining users it is below 1. For seed of size 3 average intersecting groups are 1.32 for 134 users out of 151, and for remaining 17 users it is below 1. We further discovered that in case of seeds with size 2, only 65 users have at least one seed which was intersection of more than 5 groups. The number of such seeds on an average are 8%, which means, given a seed of size 2, the chances of it appearing in more than 5 groups is 8%. For seeds of size 3, there are only 53 users who have atleast 1 seed which is common in more than 5 groups and average number of seeds which are common to 6 or more groups are only 7%. It indicates that there are very rare cases where 2 or more recipients simultaneously repeat themselves in larger number

[1] *http://www.ins.cwi.nl/projects/trec-ent/*



(a) Distinct groups (b) Address Book

Fig. 1: Example

of groups. And if a user is giving 2 recipients or more as a seed there is more chance that the user will use one of the available groups involving all the recipients in the seed rather than creating a new group. If user enters a seed {R2,R7}, then it indicates that, the probability of the user using G2 or G6 is more than the user creating a new group with this seed.

In addition to this, there is one more observation. If we look at any of the generated seeds with size 2 or more, at a time only the recent most group with that seed is used for all following emails until a new group is created. Obviously there are few exceptions but they also tend to repeat maximum two recent most groups. If we assume that groups shown in Figure 1a are arranged in chronological order, for a seed {R2,R7}, G6 will be used in most cases instead of G2. This observation can help us reducing search domain by discarding older groups.

After these observations, we can conclude that user is not inclined towards creating more and more groups involving same set of recipients. And once a new group is created with some set of recipients, user normally does not use older group, with the same set of recipients. For our hypothetical case, if one already has a group with {A,B} as a subset, most likely one will not create new group with same subset. In case, if new group is created with {A,B}, old group will become less important.

IV. Email recipient prediction using reverse chronologically arranged implicit groups

In this section, we propose a novel email recipient prediction method. The unique thing about our proposed approach is that it only requires recipient-lists to be maintained in reverse chronological order, as the prediction model. This helps in reducing computation overhead at the time of prediction.

Table I shows the algorithm of our proposed method. The algorithm takes two input parameters, the seed (S) and the set of distinct groups arranged in reverse chronological order (G). Let $g_i \in G$ be the distinct groups. It also uses one parameter n, the number of suggestions, which can also be taken as an input parameter. Given S and G, the algorithm searches for most recently used recipient-list which contains the seed. As the list are arranged in reverse chronological order, the algorithm starts its search from g_1. Once it obtains such a list, it adds n recipients from that list into the suggestion list, sug, except those which are already present in the seed. It considers recipients in order of their appearance. Then, it checks for the size of suggestion list and if it is less than n it goes for second most recent list and repeat step 3, else it returns the suggestions. The reason behind taking recipients in order of their appearance is to reduce search time in subsequent iterations for finding edges that contain seed with increased size.

There is one variable we used, that is number of suggestions we are going to show to user. Let it be n. The algorithm accepts the recipients entered by the user as the seed, S. Then it searches for most recently used recipient-list which contains the seed. Because the lists are arranged in reverse chronological order it will use the first list, g_1. $g_i \in G$, where G is the set of all distinct groups arranged in reverse chronological order. Once it obtains such a list, it will add n recipients from that list into our suggestion list, sug, except those which are already present in the seed. It considers recipients in order of their appearance. Then, it checks for the size of suggestion list and if the suggestions are less than n it goes for second most recent list and repeat step 3, else it returns the suggestions. The reason behind taking recipients in order of their appearance is to reduce search time in subsequent iterations for finding edges that contain seed with increased size.

TABLE I: Algorithm for suggesting recipients

```
Algorithm
$S$ : Seed (Recipients given by user in to field)
$G$ : Set of all distinct groups ordered in reverse chronological order
$m$ : Total number of distinct groups
$n$ : Number of required suggestion

Input : $S, G$
Output : $sug$
 1)  $sug = \phi$
 2)  for $i = 1$ to $m$, do
 3)    if $S \subset g_i$ then
 4)      for each $r_j \in g_i$ do
 5)        if $r_j \notin sug$ and $r_j \notin S$
 6)          $sug \leftarrow sug \cup r_j$
 7)          if $size(sug) := n$
 8)            return $sug$
 9)          endif
10)        endif
11)      endfor
12)    endif
13)  endfor
```

Figure 1a, shows a sample recipient list set from a user's *sent mailbox* arranged in chronological order. In order to prepare our suggestion model, we reduce multiple occurrences of edges into single occurrence by keeping only the most recent occurrence. We arrange all such single occurrences in reverse chronological order. This gives us our suggestion model as shown in Figure 2.

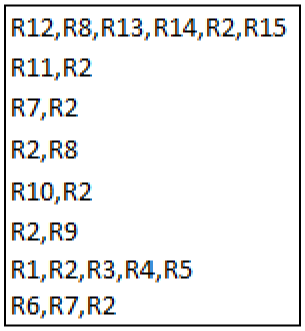

R12,R8,R13,R14,R2,R15
R11,R2
R7,R2
R2,R8
R10,R2
R2,R9
R1,R2,R3,R4,R5
R6,R7,R2

Fig. 2: Prediction Model

Suppose, a user is composing a new email and the intended recipient-list for this email is $\{R12, R8, R13, R14, R2, R15\}$. If the user enters $R2$ as a recipient, our algorithm considers it as a seed and begins the first iteration as shown in Figure 3. During iteration 1, it starts from the first recipient list in the dataset and checks whether the seed is present or not. Here, the recipient-list contains $R2$, so the algorithm will put all the members of the list, except $R2$, into the suggestion set. There is no restriction on number of suggestions as n is undefined, so it takes all recipients. Then it moves towards second list and does the same operation and so on. At the end, it provides the suggestions shown in Figure 3. Out of those suggestions user will find a match which is $R12$.

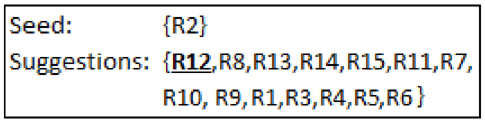

Seed: {R2}
Suggestions: {**R12**,R8,R13,R14,R15,R11,R7, R10, R9,R1,R3,R4,R5,R6}

Fig. 3: Iteration 1

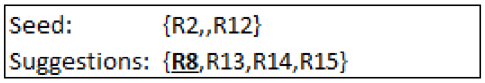

Seed: {R2,,R12}
Suggestions: {**R8**,R13,R14,R15}

Fig. 4: Iteration 2

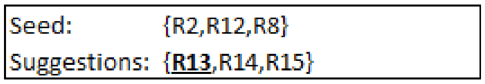

Seed: {R2,R12,R8}
Suggestions: {**R13**,R14,R15}

Fig. 5: Iteration 3

Now, $R12$ becomes the second member of the seed and second iteration of the algorithm start. At the end of second iteration, the algorithm provides suggestions shown in Figure 4. Intended recipient from the suggestion set is $R8$, which will become third member of the seed and the algorithm will execute for third iteration. These iterations will continue to occur until the suggestion list becomes empty or the user obtains all the desired recipients.

TABLE II: Summarized results in terms of precision for 96 users

	seed of size 2	seed of size 3
MAP	0.332	0.433
p@5	0.614	0.686
p@10	0.551	0.629

TABLE III: Summarised results in terms of recall for 96 users

	seed of size 2	seed of size 3
n	0.754	0.769
n=5	0.485	0.521
n=10	0.658	0.692

V. Evaluation and Comparison

During initial experimentation, we used mysql database server. But we observed that the application was taking approximately 30 hours to derive results for all 151 users. This was due to the time for communication with the database server, on the contrary we require only recipient lists of recipients of each sent message. So, we created files for each user in the Enron dataset containing recipient lists arranged in reverse chronological order of messages. Hence, we use 151 files containing recipient lists of all 151 users for experimentation. This change in the way of input gave results for all users in approximately 15 minutes.

We used Enron dataset [13] which has 151 users. We did our experiments on all 151 users of Enron dataset and retrieved detailed results. First thing we observed is, out of 151 there are 96 users which were giving valid results. Other 55 users had insufficient number of sent messages, which satisfy our screening criteria. We tried many variations during evaluation process. We defined 6 different settings, using 3 variations on n and for each value of n two variations in size of seed.

For evaluation purpose we used recent most 20% transactions of all the user, where, a transaction refers to a list of recipients for a mail. For each transaction we used all combinations of recipients of size 2 and 3, and we call it seed of size 2 and size 3 respectively. For each combination we received n suggestions and retrieved average precision and recall. Initially, we tested with undefined n, then we took n=5 and then n=10.

For showing the goodness of the algorithm, we used summarized results of these 96 users.

The result shown in Table II represents the mean average precision (MAP) we obtained for 96 users. First row is MAP for undefined n. p@n is value of MAP for n suggestions. So row 2 and row 3 depict the results for $n = 5$ and $n = 10$ respectively. As the results indicate, precision values for seed of size 3 are better than precision values for seed of size 2. Moreover, lesser value of n gives better result. In case of undefined n, the algorithm uses all lists which contain the given seed. So number of suggestions are very large than number of intended recipients. In case of smaller n it uses only most recent transactions, mostly 1 or 2 most recent ones. Although number of suggestions are less, suggestions contain intended recipients. Thus, it implies that most recent 1 or 2 transactions which contain given seed are sufficient for the prediction purpose.

In addition to information provided in Table II, while observing precision for all 96 users, we noticed that, in case of $n = 5$, there are 13 users, for whom we are getting precision greater than 90%, given seed of size 2. There are 22 users, for whom the precision is more than 90%, given seed of size 3.

TABLE IV: Summarised results in terms of hit rate(%) for 96 users

	seed of size 2	seed of size 3
n	93.20	96.03
n=5	88.99	93.79
n=10	92.10	95.67

Precision alone can not be considered a good measurement, because it gives the idea about how many are correct out of all the retrieved suggestions. So we are also presenting recall values for same set of users, to show how many intended recipients are suggested. Table III shows the recall value that is averaged over the same 96 users. Undefined n gives recall around 0.75 and that indicates that most intended recipients are suggested by the algorithm. Even in case of n=5 we are getting recall value around 0.50 which is good considering many times intended recipients are more than 5, which makes it impossible to get 1 as a recall value. For example, keeping n=5, if the total intended recipients are 7, maximum recall can be $\frac{5}{7} = 0.714$

In all the available literature for email recipient prediction, precision and recall were used for goodness measurement. So we also presented our experimentation results in terms of precision and recall. But, we can not rely on the precision and recall values for testing goodness of prediction algorithm. There are cases which gives 1 in precision and recall and there are cases where precision and recall values are 0. Average precision or recall values do not focus specifically on cases which returns 0 correct predictions. If for some user 50% of times precision was 1 and 50% of times precision was 0, we will get average precision 0.50. We will consider that a good result but it will not throw light on the fact, that the algorithm failed 50% of times. And because we have to avoid cases with 0 true prediction, it is important that measurement focus on such cases. Preceding discussion shows that it is very difficult to measure the goodness of some algorithm just by precision and recall. We worked out that measuring hit rate can be a good way to define goodness. Hit rate shows how many times algorithm returned at least on correct prediction. Table IV shows the results in terms of hit rate(%) for all six settings.

For a moment, if we think intuitively, out of the 6 different settings that we used, in which condition one can get good prediction. There are good chances of being correct if we use 3 recipients in seed and undefined n. And the worst setting for prediction is limiting number of predictions to 5 and using seeds of size 2. The reasons are, limiting the predictions can make intended recipients escape, and taking less recipients in seed can invite unintended recipients into suggestion-list. Now if we look at the hit rate for all six settings it clearly supports the preceding logic. As one can see using seed of size 3 and suggesting all possible recipients our proposed algorithm gives 96% times at list 1 correct suggestion. And as we have mentioned earlier it is not about getting all correct recipients at once, but it is about getting at least one recipient every time.

As Bartel and Dewan [2] observed that content-based methods are more efficient than meta-data based methods or fusion, we compare our proposed method with two content-based methods [11] and [2]. These meta-data based methods use implicit social graph[11]. The implicit social graph is a weighted hyper-graph and it requires weights of hyper-edges to be updated after each message is sent. In our proposed approach, during model formation recipient list of last sent message is added on top of the the reverse chronologically ordered list, and if this new recipient-list is already in the model the older occurrence is deleted. When compared, older methods seem to involve a lot of computing overhead in model formation. Hence, they are not suitable for hand-held devices, which have limited computation power and battery life.

Older methods require records of all the messages' meta-data, like recipient-list, direction of message (whether message was in inbox/sentbox), timestamp of the message, etc. To store and maintain this meta-data of a user database management system is required. That also is not suitable for a hand-held device.

Our observations in Section III clearly indicate that, in case of an implicit social graph, if seeds of size 2 or more are used then, most of the time only two edges need to be considered. Hence, building and maintaining an implicit social graph is not required, and maintaining the recipient lists in reverse chronological order can achieve the desired result. The number of lists will be the same as the number of hyper-edges in implicit social graph, but no other data will be required. When we compare the performance, based on results presented by Bartel and Dewan [2] they had average precision of [2] is around 0.75 and average precision of [11] is around 0.20, given $n = 4$ in their experimentation. Our proposed method gives average precision around 0.60, given $n = 5$. Apparently, our algorithm is trustworthy as well as hand-held device friendly.

VI. Related Work

As we have mentioned before, there are three categories of methods available for recipient prediction. In this section, we provide a brief description of past work done in this area. Pal and McCallum [9] very first tried CC prediction using message content, they took recipients present in '*To*' field as message content as well. They used multinomial naive Bayes model and standard naive Bayes model. Then,Carvalho and Cohen [4, 5] presented some prediction methods based on content. They treated message as bag of words and used TF-IDF(Term Frequency Inverse Document Frequency) vector [12] based representation. They tried some purely content based method, and also tried to extend them using meta-data based methods. Implicit social graph was used by Roth et al. [11] to predict users. It was purely meta-data based methods and features like frequency, recency and direction of message were used. It was also a seed based method, in which recipients entered by users were taken as a seed. This was the first attempt to predict users, using only the meta-data. [7] proposed Participant Co-occurrence in Social Network(PCSN) method. It also uses both meta-data and content. Instead of using message as TF-IDF vector, they tried topic similarity based model [3] on message content. It too uses the social graph. Bartel and Dewan [2] proposed a method for recipient prediction and made it clear that for prediction purpose groups are more effective than contents. They also used implicit social graph but instead of considering intersections between seed and hyper-edges they considered only those hyper-edges, having the seed as a subset.

This is not the first attempt to create a recipient prediction algorithm that uses low computational power. Balasubramanyan et al. [1] created an extension for mozilla thunderbird. Hit rate for goodness measurement is also not entirely new concept. Pal and McCallum [9] evaluated their CC prediction using hit in top 5.

VII. Conclusion

Considering the importance of email as a group communication medium, it is desirable that all the intended recipients of an email are included while composing and sending an email. Many prior studies have considered this as a challenge and have proposed many recipient prediction methods. In this study, we highlight the challenges faced by different categories of recipient prediction methods and propose computationally efficient iterative prediction method, which can be implemented in hand-held devices. The key difference between previous methods and our proposed technique is locality of generation of suggestions. This is possible, because our technique needs very less data to be stored and processed. Due to locally generated suggestions on user's device, scalability is no issue for our method. To check the efficiency, we experimented it on Enron dataset, because it is the only real-time, naturally generated and freely available email dataset. Moreover, due to the characteristics of Enron dataset we can imply that if algorithm gives good results for the employees of Enron it can do good for any user in real-time. Our method is able to achieve hit rate of 94.091% for seed of size of 2 and 97.34% for seed of size of 3 keeping n=10. For n=5, these figures are 90.88% and 95.67% respectively.

References

[1] Ramnath Balasubramanyan, Vitor R Carvalho, and William Cohen. Cutonce-recipient recommendation and leak detection in action. In *AAAI-2008, Workshop on Enhanced Messaging*, 2008.

[2] Jacob Bartel and Prasun Dewan. Towards hierarchical email recipient prediction. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 50–59. IEEE, 2012.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[4] Vitor R Carvalho and William Cohen. Recommending recipients in the enron email corpus. *Machine Learning*, 2007.

[5] Vitor R Carvalho and William W Cohen. Ranking users for intelligent message addressing. In *Advances in Information Retrieval*, pages 321–333. Springer, 2008.

[6] William W Cohen, Vitor R Carvalho, and Tom M Mitchell. Learning to classify email into speech acts. In *Proceedings of EMNLP*, volume 4. sn, 2004.

[7] Qi Hu, Shenghua Bao, Jingmin Xu, Wenli Zhou, Min Li, and Heyuan Huang. Towards building effective email recipient recommendation service. In *Service Operations and Logistics, and Informatics (SOLI), 2012 IEEE International Conference on*, pages 398–403. IEEE, 2012.

[8] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, pages 217–226. Springer, 2004.

[9] Chris Pal and Andrew McCallum. Cc prediction with graphical models. In *Conference on Email and Anti-Spam*, 2006.

[10] Sara Radicati and Justin Levenstein. Email statistics report, 2013-2017. Technical report, THE RADICATI GROUP, INC. http://www.radicati.com, 2013.

[11] Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher, Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, and Ron Merom. Suggesting friends using the implicit social graph. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242. ACM, 2010.

[12] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[13] Jitesh Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California*, 4, 2004.