2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Predicting Email Recipients

Zvi Sofershtein
The Rachel and Selim Benin School of
Computer Science and Engineering
Hebrew University
Jerusalem, Israel
zvisofer@gmail.com

Sara Cohen
The Rachel and Selim Benin School of
Computer Science and Engineering
Hebrew University
Jerusalem, Israel
sara@cs.huji.ac.il

***Abstract*—The ability to accurately predict recipients of an email, while it is being composed, is of great practical importance for two reasons. First, prediction of recipients allows for effective "auto-complete" of this field, thereby improving user experience and reducing the overhead of manual typing of the recipient. Second, this capability allows the system to alert the user when she has typed unlikely recipients. Such alerts can help avoid human error that might result in forgetting relevant recipients, or, even worse, disclosure of personal or classified information.**

In this demonstration, a system that effectively predicts email recipients, given an email history, will be exhibited. The system takes into consideration a variety of email related features to achieve high accuracy. Extensive experimentation on diverse email corpora has shown that our system adapts well to a variety of domains (such as business, personal and political email). Conference participants will be able to view real emails sent, and to observe how well our system predicted the recipients. In addition, they will be able to "impersonate" users whose email history is already available to the system, to compose a new email, and to view the recipient predictions.

I. Introduction

Email is one of the most popular means of communication, and its volume seems to be consistently increasing. It is estimated that in 2015, around 205 billion email messages will be sent per day [1]. However, email use is fraught with potential serious pitfalls, with one big problem being mistakenly sending an email to the incorrect recipient. Online searches about email blunders yield a large number of such unfortunate occurrences. In fact, one real-life blunder even ended up being the inspiration for a 2011 Superbowl commercial dealing with precisely this mistaken recipient problem.

As email is a common means of communicating, and email mistakes are potentially disastrous, systems that can reduce email error are of importance. This paper considers the problem of accurately predicting recipients for a given email. Once recipients can be predicted, the system can alert the user if he seems to be sending the email to the wrong persons. In addition, email recipient prediction is useful as a GUI feature, as it allows for easy and automatic selection of recipients, reducing the need for manual typing of this information.

Email recipient prediction has been studied in the past and experimentation has focused on the Enron dataset [2]. It was pointed out in [3] that at least 20.5% of Enron users were not included as recipients in some mails, even though they were intended recipients, i.e., mistakes in determining email recipients are commonplace. In [3]–[5] systems were presented that combine textual features with recency and frequency features. However, the textual features used in these works were rather limited, using off-the-shelf text similarity metrics (e.g., they did not use the greeting feature developed here), and experimentation was confined to the Enron dataset. In [6], [7], a recipient prediction method, based on social network features, was proposed. Intuitively, their method was based on a recipient seed in the composed message and past co-occurrence of additional recipients with the seed. However, this approach does not reduce the effort of inserting the seed, nor is it applicable when there is only one recipient.

In this paper we present a system that can accurately predict email recipients for a new email, given the user's email history. The system computes a variety of features, that are either temporal or textual based, and uses these features to learn a ranking function over the user's contacts, given a new email. Our textual features include a more targeted approach (based on email greetings) to improve the effectiveness of the system. Extensive experimentation over personal, business and political emails has shown the effectiveness of the system, which currently works for both English and Hebrew language emails.

II. Predicting Recipients

The problem at hand can be defined as follows. We consider a user u who has, until some point in time t, sent and received emails. Let $\mathcal{C}_{u,t}$ be the *contact list* of user u at time t, i.e., the set of people who have either sent an email to u, or been sent an email by u, before time t. Now, given a new email message m that u composes at time t, we would like to predict the recipient(s) to whom u intends to send this specific email. More precisely, our goal is to define a ranking function $r_m : \mathcal{C}_{u,t} \to \mathbb{R}$, such that for contacts $c_i, c_j \in \mathcal{C}_{u,t}$ it holds that $r_m(c_i) > r_m(c_j)$ implies that it is more likely that m will be sent to contact c_i, than to contact c_j. We say that an email is *predictable* if at least one of its true intended recipients is in $\mathcal{C}_{u,t}$. For obvious reasons, we are only interested in predictable emails (and all experimentation presented in Section III is for such emails).

In the following, we fix a user u. The function r_m will be learned, for users $c \in \mathcal{C}_{u,t}$, from the history of user u by leveraging a variety of features that are either temporal or textual. All features are normalized to a range of $[-1, 1]$ to ensure proper convergence of the learning mechanism applied and to adjust the values so that functions learned from different users will be comparable.

ASONAM '15, August 25-28, 2015, Paris, France
ACM ISBN 978-1-4503-3854-7/15/08
DOI: http://dx.doi.org/10.1145/2808797.2808805

A. Temporal Features

We consider four features that are based on the time in which message m is written. In the following, we use $\mathcal{M}_{c_1 \to c_2,(t_1,t_2)}$ to denote the set of messages sent by c_1 to c_2 in the time interval between time t_1 and t_2 (excluding). Note that we will use the wildcard $*$ instead of c_1 or instead of c_2 to denote, respectively, the set of *all* emails sent to c_2 or sent by c_1 within the given time interval. Also note that we use 0 to denote the beginning of time, and thus, an interval of the form $(0, t_2)$ will allow for all messages until time t_2.

The first two features return values, for a contact c, that are proportionate to the frequency of messages written by u to c, or by c to u.

- **Outgoing Message Percentage:** The *outgoing message percentage* of c for u at time t is the percentage of messages sent by u to c, with respect to all messages written by u, i.e.,
$$\frac{|\mathcal{M}_{u \to c,(0,t)}|}{|\mathcal{M}_{u \to *,(0,t)}|}$$
- **Incoming Message Percentage:** Similarly to the previous feature, the *incoming message percentage* is the percentage of messages sent to u by c, with respect to all messages sent to u, i.e.,
$$\frac{|\mathcal{M}_{c \to u,(0,t)}|}{|\mathcal{M}_{* \to u,(0,t)}|}$$

The other two features are based on the time elapsed since u sent an email to c or received an email from c. Instead of using this time value directly as a feature, we choose to count how many messages have been sent (or received) since the last communication with c. This is an important normalization method as different users can have different habits—some send out emails every few seconds, while for others, hours may pass between consecutive emails.

- **More Recent Outgoing Percentage:**. Let t' be the time at which the last message was sent from u to c, or 0 if u never sent a message to c. The *more recent outgoing percentage* of c at time t is the percentage of messages sent by u since she last emailed c, i.e.,
$$\frac{|\mathcal{M}_{u \to *,(t',t)}|}{\alpha|\mathcal{M}_{u \to *,(0,t)}|}$$
If $|\mathcal{M}_{u \to *,(0,t)}|$ is empty, i.e., u has never written an email, this feature has the constant value 1. Note the use of α in the denominator to further differentiate (and increase the gap) between the cases in which a message was sent to c only very early on, and the case in which no message was ever sent to c. In our experimentation, we used $\alpha = 2$.
- **More Recent Incoming Percentage:** Let t' be the time at which the last message was sent from c to u, or 0 if c never sent a message to u. The *more recent incoming percentage* of c at time t is the percentage of messages sent to u since she last received an email from c, i.e.,
$$\frac{|\mathcal{M}_{* \to u,(t',t)}|}{\alpha|\mathcal{M}_{* \to u,(0,t)}|}$$
As before, if $|\mathcal{M}_{* \to u,(0,t)}|$ is empty, i.e., u has never received an email, this feature has the constant value 1.

B. Textual Features

We consider three features that are based on the textual content of the message m being sent. Note that we include both the subject and the content of the message, when considering the text of m. The first two features are used to indicate whether contact c has either received from u, or sent to u, a message with text similar to that of m.

- **Outgoing Textual Similarity:** We index all previous emails sent to u, or received by u, using the Apache Lucene search engine.[1] Then, given a new message m, we search the indexed data to find similar messages sent by u. Let $\mathcal{M}^m_{u \to *,(0,t)}$ be messages similar to m, sent by u, found using the Okapi BM25 textual similarity measure with Lucene. Then, for a given contact c, the *outgoing textual similarity* feature is 1 if $\mathcal{M}^m_{u \to *,(0,t)}$ includes a message sent to c, and -1, otherwise.
- **Incoming Textual Similarity:** This feature is similar to the previous. Let $\mathcal{M}^m_{* \to u,(0,t)}$ be the most similar messages to m, sent to u, according to the Okapi BM25 textual similarity measure. Then, for a given contact c, the *incoming textual similarity* feature is 1 if $\mathcal{M}^m_{* \to u,(0,t)}$ includes a message sent by c, and -1, otherwise.

We note the underlying assumption here is that the interaction between two users may be on diverse topics, and it is sufficient to find a single match for a contact c, to derive a strong indication that c should be the message recipient. Previous works measured the average similarity of the messages between the user and the candidate, but this feature degrades for contacts that are very popular, and hence, appear in many contexts.

Greeting. Our final textual feature takes into consideration the fact that emails often include, in their content, the name, or nickname, of the recipient, by way of a *greeting* at the beginning. For example, an email may begin with text such as `Dear Sam and Jim`, `Hi Mom`, or `Sweetheart!`. In each of these cases, the user has included some indication as to the recipient (i.e., `Sam`, `Jim`, `Mom` and `Sweetheart`). To the best of our knowledge, this feature has not been considered in the past for email recipient prediction.

To leverage email greetings, we associate with a message m the set $\mathcal{G}_m$ of all names or nicknames appearing in the greeting of m. We note that this set can be empty, if there is no greeting, or if the greeting does not contain any names (such as `Hi All!`). We explain below how to automatically create the set $\mathcal{G}_m$ using a rule-based system.

We define the value of the *greeting* feature given a new message m and a contact $c \in \mathcal{C}_{u,t}$ as:
$$\begin{cases} 0 & \mathcal{G}_m = \emptyset \\ 1 & \mathcal{G}m \cap (\cup_{m' \in \mathcal{M}_{u \to c,(0,t)}} \mathcal{G}_{m'}) \neq \emptyset \\ -1 & \text{otherwise} \end{cases}$$

[1]http://lucene.apache.org/

Rule	Text
<Greeting name>	`David,`
	`honey`
<Greeting word><Greeting name>	`Dear David.`
	`Hi dear!`
<Greeting name><Connector ><Greeting name>	`David or Mike:`
<Greeting word><Greeting name><Connector ><Greeting name>	`Hello David & Mike`

TABLE I. SEVERAL OF OUR PARSING RULES, AND EXAMPLES OF TEXT THAT IS SUCCESSFULLY PARSED USING THESE RULES.

In other words, if m contains no greeting names, then the feature has the constant value of 0 for all contacts. Otherwise, m has some greeting names. In this case, the feature has the value of 1 for contacts who have been sent emails from u using some greeting from $\mathcal{G}_m$, and -1 for the remaining contacts.[2]

We now discuss our rule-based system for deriving $\mathcal{G}_m$ from m. This system is not foolproof, but has been experimentally evaluated over large corpora (both in English and in Hebrew) and has been shown to be very effective. We present the main features of our system, but some details and special cases are omitted due to space restrictions.

The basic assumption behind our rules is that greetings tend to be short and mostly comply with one of several popular structures. Typically, a greeting will appear at the beginning of the message, before any separating symbol (such as a period, comma or colon). An exception to this rule is for messages with multiple recipients, in which case separator symbols may appear between names (e.g., `Bob, Jim, Sally:`). We do not discuss this case any further, but special rules have been developed that are applicable for messages with multiple recipients.

We differentiate between five types of tokens (i.e., textual sequences) that can appear at the beginning of a message, until the first separator:

1) *Greeting words* such as `hi`, `dear`, `mr.`, `thanks`, etc.
2) *Nonspecific names* such as `there`, `again`, `all`, etc. All of these are commonly used in the same place where a regular name or nickname could be used.
3) *Standalone words* such as `yes`, `sure`, `oops`, etc. (These are needed, e.g., to differentiate between a message beginning `Jim!` and one beginning `Oops!`.)
4) *Greeting names* such as `David`, `Honey`, etc. These are words not belonging to the previous groups.
5) *Connectors* such as `or`, `&`, etc.

Some examples of extraction rules using these groups, as well as text that is correctly parsed, appear in Table I. We note that our system correctly recognizes that the following phrases contain no names or nicknames: `Hi`, `thanks!`, `hi there`, `hello again`, `Yes.`, `sure...`.

C. Learning a Ranking Function

Given the above features, we learned a ranking function using a ranking SVM [8]. For every predictable email, we generated one positive example per recipient. Correspondingly, every contact that is not a recipient of the email can be considered as a negative example. Mailboxes usually have dozens or hundreds of contacts, but most emails have only a few recipients. Therefore, taking all possible non-recipient contacts will result in a training set that consists almost entirely of negative examples. Since an SVM attempts to minimize the error on the training set, when applying it to such datasets, it will learn to always classify as negative and will output an arbitrary ranking function. To avoid this problem, we limited the number of negative examples of each email by the number of its positive examples. These examples were randomly selected from all possible negative examples of the email.

[2]Some emails have several recipients and more than one name in the greeting. In such cases, we cannot distinguish to which recipient each name refers, and therefore $\mathcal{G}_m$, for an email sent to c, may actually contain a name not referring to c.

Dataset	Users	All Emails	Predictable
Enron	140	517401	42028
Gmail	10	-	12536
Political	3	323180	13309
Total	153	> 840581	67873

TABLE II. DATASETS. DUE TO PRIVACY CONCERNS, WE EXTRACTED FROM PRIVATE GMAIL MAILBOXES ONLY THE BARE MINIMUM NECESSARY FOR OUR EXPERIMENTATION. THEREFORE, THE NUMBER OF ALL MAILS IN HISTORY IS MISSING.

III. EXPERIMENTAL EVALUATION

We used three datasets to evaluate our method, as detailed in Table II. The Gmail dataset was extracted from private mailboxes of Gmail users who participated in our experiments. Note that we did not collect the data itself, but only the required feature values. The political dataset is a collection of freely available mailboxes of 3 well-known public figures: John “Jeb” Bush, Elena Kagan and Sarah Palin.

The predictable emails are obtained by discarding duplicates, incoming emails, first 10 emails of a user, reply emails and emails to a new contact. In all these cases the prediction task is either meaningless or trivial.

In our experimentation we learned a personalized ranking function for each user in each dataset. For each user, we used the first half of the predictable mails as a training set and the remainder as a test set. We tested our prediction in two settings:

- **Full Ranking:** Rank all contacts.
- **Known-Character Ranking:** Rank only contacts with addresses beginning with the same character as some correct recipient. This setting simulates a scenario where the user correctly inserts the first character of a recipient and the system attempts to “auto-complete” the recipient’s address.

	Full Ranking				Known-Character Ranking			
Method	Overall	Enron	Gmail	Political	Overall	Enron	Gmail	Political
Personalized Function	**0.470**	**0.468**	**0.437**	**0.526**	**0.780**	**0.751**	**0.813**	**0.838**
More Recent Outgoing Percentage	0.326	0.303	0.405	0.329	0.701	0.667	0.768	0.743
Outgoing Textual Similarity	0.322	0.327	0.257	0.397	0.588	0.604	0.576	0.553
More Recent Incoming Percentage	0.274	0.202	**0.491**	0.312	0.564	0.521	0.807	0.474
Outgoing Message Percentage	0.272	0.283	0.186	0.350	0.606	0.589	0.579	0.687
Incoming Textual Similarity	0.213	0.206	0.192	0.278	0.415	0.404	0.504	0.366
Incoming Message Percentage	0.199	0.193	0.154	0.299	0.491	0.489	0.560	0.435
Greeting	0.112	0.142	0.056	0.043	0.276	0.327	0.257	0.133
Random Guess	0.038	0.044	0.021	0.026	0.194	0.224	0.190	0.113

Fig. 1. MRR scores of our personalized functions in comparison with ranking functions induced by each feature alone.

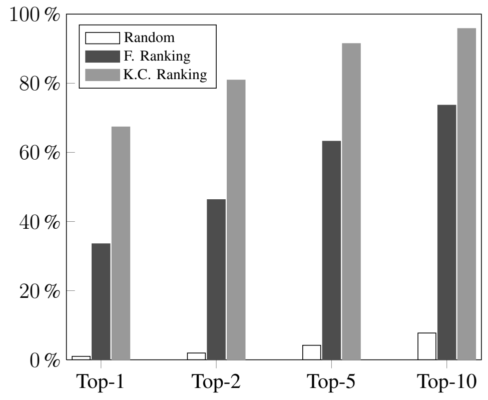

Fig. 2. Probability to have a hit among the top-k ranked elements.

Figure 1 contains the MRR (mean reciprocal rank) value for our personalized functions (Line 1), for functions determined by a single feature (Lines 2–8) and for the baseline random guess (Line 9). As seen in this figure, our personalized functions outperformed all ranking functions induced by each feature alone, in almost every case.[3] Comparison with the baseline "random guess" demonstrates the effectiveness of both the features and the personalized functions.

Figure 2 shows the probability (averaged over all datasets) to have at least one correct recipient in the top-k ranked contacts in three settings: random guess for full ranking, personalized function for full ranking and personalized function for known-character ranking. The effectiveness of our system is immediately apparent.

Our personalized function performed especially well on the political dataset. This is probably due to the fact that the mailboxes for the political users were particularly large, and hence, our function could learn with greater accuracy. Nevertheless, the personalized function performed well on all corpora. The results also show that our feature set was sufficiently small and indicative to be accurately learned even when just a small training set is available (small mailboxes are common in the Enron and Gmail datasets).

[3]In one case "more recent incoming percentage" outperformed the personalized function. This seems to be due to the fact that the Gmail datasets were mostly small accounts, except for two very large accounts of users—including one of the authors of this paper—who have a compulsive habit of immediately answering every email they receive. For larger and more diverse datasets, we expect the personalized function to be the best performing.

IV. Application and Demo Scenario

In our demonstration, conference participants will be able to "impersonate" people from our political dataset (Jeb Bush, Elena Kagan and Sarah Palin) and view recipient predictions for fake emails that they compose on behalf of the true mailbox owner. Participants will also be able to view predictions for the real emails within this corpora, and thus, will be able to see how effective our system is. Finally, we will also exhibit our warning mechanism, which is triggered when conference participants attempt to "send" some composed email to unlikely recipients or to omit intended recipients.

V. Acknowledgements

The authors were partially supported by the Israel Science Foundation (Grant 1467/13) and the Ministry of Science and Technology (Grant 3-9617).

References

[1] S. Radicati and Levenstein. Email statistics report, 2015-2019. Technical report, 2015.

[2] J Shetty and J Adibi. Enron email dataset. *USC Information Sciences Institute, Tech. Rep*, 2004.

[3] Vitor R. Carvalho and William W. Cohen. Ranking users for intelligent message addressing. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR'08, pages 321–333, Berlin, Heidelberg, 2008. Springer-Verlag.

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

[5] Vitor R. Carvalho. *Modeling Intention in Email - Speech Acts, Information Leaks and Recommendation Models*, volume 349 of *Studies in Computational Intelligence*. Springer, 2011.

[6] 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 Conference on Knowledge Discovery and Data Mining*, 2010.

[7] Jacob W Bartel and Prasun Dewan. Towards hierarchical email recipient prediction. In *CollaborateCom*, pages 50–59, 2012.

[8] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.