2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)

DejaVu: A Case for Assisted Email Replies on Smartphones

Uma Parthavi Moravapalle
Georgia Institute of Technology, Atlanta, GA
parthavi@gatech.edu

Raghupathy Sivakumar
Georgia Institute of Technology, Atlanta, GA
siva@ece.gatech.edu

***Abstract*—More email is opened on mobile devices today than on other platforms [1]. At the same time, enterprises are constantly investing in approaches to improve employee productivity. In this paper, we consider the problem of automated information suggestions to assist in $reply$ construction. The basic premise of the work is that a significant portion of the information content of a $reply$ is likely to be present in prior emails. We first show that the premise is valid by analyzing both public and private email datasets. We then present a simple algorithm that relies on inverse document frequency (IDF) and keyword matching to provide relevant suggestions during $reply$ construction. Through prototype evaluations done using the Email datasets, we show that the proposed algorithm has attractive benefits.**

***Index Terms*—Enterprise email, Smartphones, Redundancy**

I. Introduction

Enterprises today are investing heavily in their mobile workforces with an eye toward boosting productivity and customer service. 91% of mobile workers in enterprises use a smartphone for their work. On the other hand, with information now ubiquitously accessible, job functions of enterprise employees increasingly involve handling, using, or analyzing information. The juxtaposition of these two trends: increasing reliance on access to information, and ubiquitous mobile connectivity — forms the context for this paper. We specifically focus on one dominant form of information sharing within enterprises – Email. The average enterprise employee sent/received 126 emails per day in 2015 [2]. This deluge of emails results in an average enterprise worker spending 28% of her work time in reading and responding to emails [3]. A large portion (70%) of these emails are opened on a mobile device [1].

The challenge we explore in this paper is the burdensome experience of typing replies to emails using the smartphone's small on-screen keyboard. A recent study has indicated that over 30% of all email replies are over 100 words long [4]. Assuming the typing speed of an average user on a smartphone to be 20 words a minute [5], it takes more than 5 minutes to type a 100-word email response on a smartphone. This directly translates to productivity related costs for enterprises.

One approach to reduce this burden is to automatically generate suggestions for the content of email replies, which the user can select, modify and send. The content of a typical email response can be classified into two categories: non-informational (e.g., generic words and phrases such as 'okay for a meeting', 'sure', etc.) or informational (specific responses such as an address, a budget proposal, etc.). There are existing solutions that perform email reply assistance by suggesting appropriate non-informational content (Google's Smart Reply [6] and Apple's Quick Type [7]).

In this paper, we explore if such assistance is achievable for the *informational content* of the replies. Specifically, we ask the following question: *For a mobile user, if information required for a $reply$ to an incoming email is available in past emails within the inbox/sent-box/other-folders of that user, could that information be identified, retrieved, and presented to the user in a fashion that eases the burden for the $reply$ construction?* The goal of such an informational email reply suggestion solution is not to replace the existing non-informational suggestion solutions but to compliment them with informational content[1].

In answering the above question, we make the following key contributions in the paper: (1) We use both publicly available email datasets (Enron Corporation email dataset [8] and Hillary Clinton email dataset [9]) and several volunteer user email datasets to analyze the potential for retrieving information from existing emails to help in response construction. In total, we analyze 242853 emails belonging to 26 different users, and show that the results are quite promising with the percentage of responses that have a 60% similarity match with past emails being 61.72% for 3 past emails; (2) We demonstrate the feasibility of suggesting informational replies through a simple algorithm that is based on keyword match between the email being responded to and past emails. Using a prototype, we show that this simple approach is capable of providing effective suggestions nearly 32% of the time. This indicates that there is tremendous scope in reducing the user burden through informational reply suggestions. Given the recent advances in natural language processing and information retrieval, we hope that this paper opens the doors for algorithms that provide more tailored suggestions in the future.

This work was funded in part by the National Science Foundation under grants CNS-1513884, CNS-1319455, and IIP-1701115, and by the Wayne J. Holman Endowed Chair.

[1]It is worth noting that this question is easily extensible to include not just past emails but also other sources of content such as stored files, IM history, online content repositories, the public web, etc., but we restrict the focus of this paper only to past emails.

User	Type	# of emails	User	Type	# of emails
1	Academia	8263	6	Academia	21316
2	Personal	17781	7	Academia	16280
3	Academia	15276	8	Enterprise	23142
4	Personal	10351	9	Academia	29056
5	Enterprise	11595	10	Personal	7841

TABLE I: VOLUNTEER dataset

ID	Employee	# emails	ID	Employee	# emails
1	Hayslett, R	2554	9	Sanders, R	7329
2	Arnold, J	4898	10	Neal, S	3268
3	Kitchen, L	5546	11	Lokey, T	1156
4	Farmer, D	13032	12	Steffes, J	3331
5	Kaminski, V	12363	13	Derrick, J	1766
6	Skilling, J	4139	14	Causholli, M	943
7	Maggi, M	1991	15	Geaccone, T	1592

TABLE II: ENRON dataset

II. Data and Methodology

A. Datasets:

Two of the datasets we analyze are publicly available datasets - (a) ENRON is an email dataset comprising of email accounts of 150 top-level executives made public during the US SEC investigation of Enron Corporation for fraud. In this paper, we restrict our attention to 15 employee email accounts comprising of 74007 emails[2](Table II); (b) HILLARY is an email dataset (7945 emails) made public during the recent investigation into the use of a private email server by former Secretary of State and Senator Hillary Clinton.

Both datasets, while public and unbiased, are limiting in different ways. ENRON is a relatively old dataset dating back to 2001[3]. HILLARY consists of newer emails spanning the last few years, but is limited to the Inbox of one person. Hence, we also rely on VOLUNTEER - a built from scratch dataset comprised of emails to/from ten volunteer users belonging to diverse backgrounds (Table I). We distributed a custom-built python tool to each volunteer who ran it on their respective computers. This tool communicates with the respective email servers using IMAP and dumps emails along with their headers from 1/1/12 to 1/1/16, to the local hard disk. For users using Gmail servers, we adapt the python tool to access the email accounts through XOAUTH2.0 protocol. This dataset has a total of 160901 emails.

B. Processing:

Both the ENRON and VOLUNTEER datasets consist of raw email data and are pre-processed for further analysis. A raw email starts with header data, followed by email body content and any attachments. If the email is a reply, the email clients quote the original message (to which this email is a reply) along with the email body. The format of these quotes differ for different clients and there is no standard way of including quoted text with the email. Sometimes the quoted text is clearly marked with '<'. In other cases, the quoted text follows lines such as '——Original Message——'. Through heuristic rules made from careful observation of the datasets, we scrub the quoted text from the reply text.

At this stage, we also add a 'Is-Reply' field to the header to indicate if the email contained quoted text. As signatures are present in a large number of emails and do not carry any special significance from an information standpoint, we also remove user signatures from the dataset using Talon, a popular library with classifiers to identify signature lines [10].

Using a custom-built python tool, we analyze the $replies$ in all three datasets and compute the amount of information in the $replies$ that is already present in one or many past emails. A large amount of repeated content in the $replies$ indicates the potential for an effective suggestion mechanism in $reply$ construction. For each email account, the tool calculates the *similarity* between every $reply$ and every other email with a timestamp earlier than that of the $reply$. The tool first converts the email text to lower case and removes any punctuations. Then, it deconstructs each email into a vector of words. Stopwords [11], i.e. words that commonly occur in English but do not have any special meaning like 'a', 'an', 'the', etc. are filtered out from this vector of words. Each word in the vector of words is stemmed to its root using the Porter Stemmer [12]. For example: 'presenting' and 'presented are stemmed to 'present'. The tool also maintains the number of emails a particular stemmed word occurs in.

C. Metrics:

A metric that measures the amount of information in one email (say $em1$) that is repeated in another email (say $em2$) should be - (a) high if a large portion of information in $em1$ is present in $em2$ and vice versa; (b) independent of any other information present in $em2$; (c) consider the relative importance of information i.e. the effect of words that are repeated frequently in several emails should be less than that of special words that occur infrequently. Based on the desired properties stated above, we define the similarity between two emails $em1$ and $em2$ as follows[4]:

$$similarity(em1, em2) = \frac{\sum_{w \in WV_1 \cap WV_2} IDF(w)}{\sum_{w \in WV_1} IDF(w)} \quad (1)$$

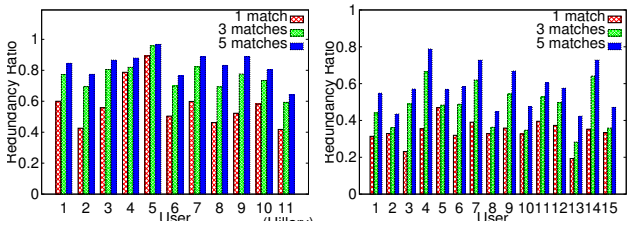
$$IDF(w) = 1 + log(\frac{N}{C(w)}) \quad (2)$$

where WV_1 and WV_2 are lists containing the stemmed words in $em1$ and $em2$, respectively. N is the total number of emails and $C(w)$ is the number of emails containing word w. In other words, similarity between two emails is defined as the weighted ratio of number of words common to both the emails to the number of words present in the first email. Each word's weight is a function of the number of emails it occurs in, called the inverse document frequency function IDF. The value of IDF for frequently occurring words is less than that of words that are relatively less common. This metric is also independent of the size of $em2$.

[2]This subset includes controversial names such as Skilling, former president and COO of Enron

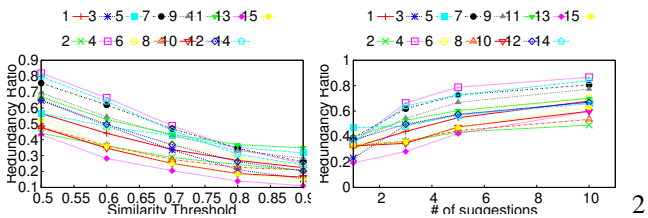
[3]Since the focus of this paper is solely on the email body (and does not extend to attachments), we believe that the relatively dated ENRON dataset does still hold relevance for purposes of our analysis.

[4]Note that the traditional TF-IDF metric does not satisfy (b)



(a) VOLUNTEER, HILLARY (b) ENRON

Fig. 1: Reply redundancy in mailboxes



(a) τ (b) Suggestions

Fig. 2: Sensitivity analysis for ENRON

III. Analysis

Using the tool described in the previous section, we first compute the similarities between every $reply$ and every other email with a timestamp earlier than that of the $reply$ and extract 1/3/5 matching emails having the highest similarity with the $reply$. The amount of $redundancy$ in the $reply$ is defined as $similarity(em, \sum_{i=1}^{M} m_i)$, where $m_i, \quad \forall i = 1 : M$ are the top M matches for em. We then compute $RedundancyRatio(\rho)$ as the ratio of replies with $redundancy > \alpha$. We choose a threshold (α) of 0.6 as the goal of these experiments is not to find identical matches for the $replies$, but to find emails that match a considerable portion of the $reply$ (second example in Table III). We evaluate the effect of threshold α later in this section.

Figures 1a and 1b show the ρ for the ENRON and VOLUNTEER datasets for 1/3/5 matches. The ρ for HILLARY is included in Figure 1a as user 11.[5] For the ENRON dataset, the average percentage of replies with high redundancy (>0.6) was 33.84%, 47.37% and 57.43%, respectively, for the top 1, 3 and 5 prior email matches. For the VOLUNTEER dataset, the average percentage of replies with high redundancy (> 0.6) was 57.75%, 76.07% and 83.20%, respectively, for 1, 3 and 5 matches. These results indicate that there is considerable amount of repeated content in all the email accounts.

Table III shows two examples of $replies$ and their corresponding top match from the mailbox of one user in the ENRON dataset. In the first example, the $reply$ and the matched email contained the approval responses to different expense reports for an employee. The $reply$ from the second example has a meeting announcement that matched with a similar meeting announcement sent out in the past. These examples illustrate ways in which content is repeated.

A. Sensitivity Analysis

Figures 2a and 2b show the effect of changing the similarity threshold α and number of suggestions, respectively, on the ρ for 15 users in the ENRON dataset. The results for the VOLUNTEER dataset are similar. As similarity threshold α is increased, the ρ falls for all the users. This is because as α is increased, the threshold at which we decide whether suggestions are useful or not increases. On an average, increasing α from 0.6 to 0.7 decreases the ρ by 15.48%. On the other hand, decreasing α from 0.6 to 0.5 increases the ρ by 18.01%. Also, as expected, as the number of suggestions increase, the ρ increases. Initially ρ increases rapidly, then it saturates. This indicates, the text in the reply is only concentrated in a few emails in the mailbox and is not spread out across a large number of emails. Specifically, as the number of suggestions is increased from 3 to 5, the ρ increases by 31.26% for the ENRON dataset. On the other hand, decreasing the number of suggestions from 3 to 1 decreases the ρ by 26.72%. Increasing the suggestions beyond 10 has little effect on the ρ. From this figure, it can be observed that 3 would be an ideal number of suggestions as it is around the midpoint of the knee of the curve. We also measured the sensitivity of ρ to the inbox size, sent box size and the number of lines in the reply and found that as the inbox/sent box sizes or the number of lines increase, the ρ increases. The results are omitted in the interest of brevity.

On a Quadcore 3.4GHz linux computer, finding top 3 matches in a database of approximately 15K emails took 43.7 seconds on average. This indicates that for larger email accounts, finding matches solely on a resource constrained smartphone might be prohibitive. This motivates an architecture where in heavy computation related to matching and text processing is done on a cloud and the results pushed back to the email client.

B. Insights

The analysis on the three datasets has led us to the following key insights - (a) The high degree of redundancies in the replies show that the information in the reply is most likely present in the email account in some form and this can be leveraged to reduce user effort on email and hence increase productivity; (b) As the number of suggestions increase, the chances of finding the email with similar content increases. If the content of a reply exists in previous emails, it is concentrated in just a few emails. (c) The ideal number of suggestions and the ideal similarity threshold are 0.6 and 3, respectively; (d) Text processing involved with mining the database is computationally intensive and cannot be done only on the mobile device.

IV. The DejaVu Solution

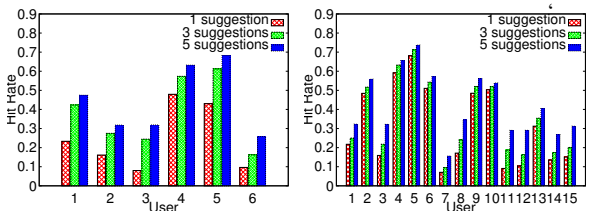
A. Problem Definition and Scope

We define the informational reply suggestion problem as follows - *For an email user, given that a reply to an Inbox email may consist of content that is present in a prior*

[5]Since the HILLARY dataset is preprocessed without headers or quoted text, there was no way of determining which email was a reply. Here, we computed the redundancy values for all sent emails. The presented results will thus be a lower bound.

Similarity	Matches
0.9	**reply:** Ken Lay has approved the attached expense report for Rosalee Fleming **match:** I have approved the attached expense report for Rosalee Fleming
0.65	**reply:** As was earlier announced, we will be bringing all Managing Directors together on a quarterly basis. Please note on your calendars the first Monday of every quarter from 8:30 a.m. to 12:00 noon for this purpose. The first meeting will take place on Monday October 1st. If you have any questions, please call Joannie Williamson. **match:** As announced earlier, we will be bringing all Managing Directors together, on a quarterly basis. Please hold open the first Monday of every quarter (from 8:30 a.m. to 12:00 p.m.) for this purpose. However, our first meeting will be on Tuesday, October 2nd.

TABLE III: Example matching email snippets for a user in ENRON dataset



(a) VOLUNTEER **(b) ENRON**

Fig. 3: Hit rates for different users

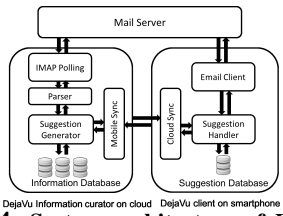


Fig. 4: System architecture of *DejaVu*

email, can appropriate information be retrieved from earlier emails and provided as suggestions to the user while the reply is being constructed?[6]. Specifically, the reply suggestion solution should - (1) suggest relevant content and should have high similarity to the intended reply; (2) be presented in an unobtrusive fashion on a smartphone; (3) should not place a severe burden on the smartphone's constrained resources; (4) be user friendly and easy to learn;

B. DejaVu: A preliminary Solution

While the previous section shows that there is considerable redundancy in the sent email content, the question of whether this redundancy can actually be leveraged to make useful reply suggestions still remains at large. In this section we present details of *DejaVu*, a very simple automated approach to generation of suggestions based on keyword matching to assist in $reply$ construction. While the approach is simple, the algorithm shows tremendous promise and presents a case that answering the question posed in Section 3.4. At a high level (Figure 4), *DejaVu* consists of a *Information Curator* (on a cloud) that constructs an *Information Database* with the user's mailbox and indexes it. When the user wants suggestions for constructing a $reply$ to an email, the *Information Curator* extracts context from this email, computes suggestions from the *information database* through a Suggestion Generator using the context. These suggestions are synced to the mobile device's Suggestion Database. When the user selects $reply$ on the *DejaVu* client, she is presented with suggestions - best matching previous emails split at a sentence level granularity (retrieved through the *Suggestion Handler*. The user can then modify the selected suggestion and send the reply. In the rest of this section, we describe the key design elements of *DejaVu*.

1) What is the granularity of suggestions?: DejaVu considers a full email to contain the lowest granularity of stand-alone information, independent of other emails. *Information Curator* parses emails in their entirety and stores them in the *Information Database*. Therefore, the granularity of suggestions is also full emails. We make this design choice as opposed to other granularities such as sentences because the amount of information (keywords) present in a sentence is low and a sentence is usually not independent but depends on other sentences around it.

2) What information is stored in the database?: Information Curator of the *DejaVu* system parses each email from the user's mailbox irrespective of the folder it is in. It separates out the email header from the MIME message and filters out any content that is not plain text, such as attachments, pictures, HTML, etc. Any quoted text (original email attached to a reply) and signature lines are then removed from the from the email body using the pre-processing step described in Section 2. Apart from the email body, the $date$ the email was sent/ received, the ID and the $subject$ are also extracted from the email. Modern email headers have an 'In-Reply-To' field for $replies$ that contains the ID of the (parent) email to which the current email is the $reply$. *Information Curator* collects this parent email message ID from the header. Any remaining lines that do not have any quoted text or signatures are added to the *Information Database* along with the ID, the parent email ID (if any), the $date$ and the $subject$.

3) How is the content indexed?: Each entry in the *Information Database* i is indexed by a set of keywords extracted from it. The text of an entry in the *Information Database* is initially converted to lowercase and then split into constituent words $W(i)$. Any punctuations are removed from these words. The most common words in English, also called 'stopwords' are then filtered and removed from $W(i)$, To capture the core context of text in the $index$ and to avoid duplicates of words that are close in meaning to each other, we also trim every word in $W(i)$ to its root. Each entry in the *Information Database* is then indexed on the set of roots of words in $W(i)$.

4) How are suggestions extracted?: The email whose suggestions are to be extracted (em) is parsed and the core context in the form of a list of keywords $KW(em)$ is extracted from it. The *Information Curator* then matches $KW(em)$ with the $index$ of entries in the *Information Database*. An obvious solution for finding suggestions would be to match $KW(em)$ with just the keywords in the $index$ of an entry in the *Information Database*. However, this simplistic solution

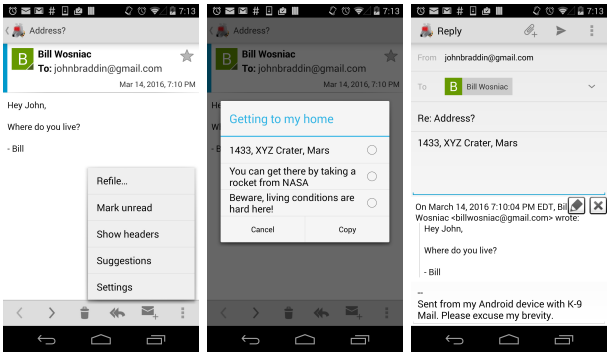
[6]We do not consider email attachments, something that would be of obvious use to consider. We defer such consideration to future work

will most likely not work well in the context of emails. This is because email is primarily designed as a medium of asynchronous communication between two parties. Most of the emails are conversations between individuals and the context of one conversation might not be contained entirely within one email and can span multiple emails. In many cases, the information content in the $reply$ doesn't hold a lot of significance on its own and the email provides context for the $reply$. Therefore, an email and it's $reply$ when considered together carry significance, and not separately. Also, given the rising trends in usage of email on mobile devices, users often resort to shorter replies and informal sentence construction In this case, without the parent email's $index$, it would be hard to retrieve any information relating to the conversation, just from the child email's context. Therefore, *DejaVu* combines the keywords in the indices of an entry and its parent (if any) to find matches i.e. $KW(em)$ is matched with $index(i) \cup index(parent(i))$.

The degree of match ($similarity$) between a set of keywords KW and the combined index $cindex(i) = index(i) \cup index(parent(i))$ is computed as $\frac{\sum_{w \in KW \cap cindex(i)} IDF(w)}{\sum_{v \in KW} IDF(v)}$, where IDF is the inverse document frequency function defined in Equation 2. In other words, $similarity$ is the ratio of the sum of IDF for words that are present in both the set of keywords KW and the combined index $cindex(i)$ to the sum of IDF for all the keywords in KW. Using IDF as weights in the ratio for keyword matching ensures that the presence/absence of keywords that occur less frequently in the user's mailbox carries a higher weight in computing the $similarity$. This is based on the intuition that keywords that occur with less frequency carry more importance. After computing the $similarity$ between KW and every other entry i in the *Information Database* , *Information Curator* then returns a set of information entries with the highest $similarities$ to i as suggestions to the email [7].

5) When are the suggestions retrieved?: *DejaVu* client uses a hybrid push/pull model for retrieving suggestions. Upon receiving a new email em in any folder of the mailbox, the *Information Curator* computes suggestions $S(em)$ and stores them in a *Suggestion Database*. $S(em)$ is pushed to the *DejaVu* client on the smartphone, who stores it in a local database. This database on the smartphone only stores the suggestions for a small fixed number of latest emails (say 100) due to resource constraints on the smartphone. When the smartphone user hits 'reply' to an email, the suggestions are retrieved from the local database by the *DejaVu* client and presented to the user. If the suggestions for the email are not already present on the local database, the *DejaVu* client pulls them from the *Information Curator*. Storing a copy of suggestions on the *DejaVu* client enables the smartphone user to retrieve suggestions even when she is offline and not connected to the *Information Curator*.

[7]Note that while the combined index of an entry and its parent is used in matching, only the entry is included in the suggestions.



(a) Reply Menu (b) Suggestions list (c) Email reply

Fig. 5: Prototype screenshots

6) How are the suggestions presented?: When the user selects 'reply', a list of constituent sentences in a suggestion grouped by their $subject$ lines are shown to the user. The user can select any number of these sentences, upon which they are automatically copied onto the clipboard and pasted during reply construction.

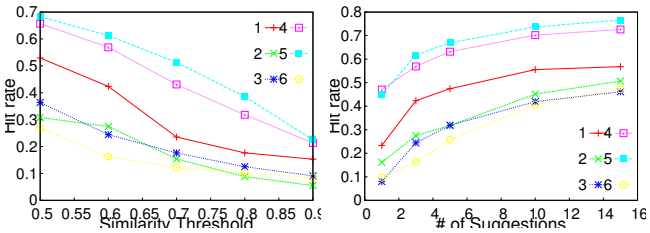
V. Prototype and Evaluation

A. Prototype

We developed a prototype for *DejaVu* client on Android OS and *Information Curator* on a Linux machine (in Python). We modified K-9 mail client [13], a popular open source email client application for Android to act as a *DejaVu* client. We added a 'Suggestions' options to the menu UI of an email (Figure 5a). When the user selects this option, a list of suggestions for that email, retrieved from a database in the external storage, are displayed. The subject line of the suggestion is displayed on the list. When the user selects one of these suggestions, another dialog box with a list of constituent sentences is brought up on the screen with options to select any number of these sentences (Figure 5b). When the user hits 'copy', a $reply$ is constructed with the selected sentences. The user can edit the $reply$ before sending it out (Figure 5c).

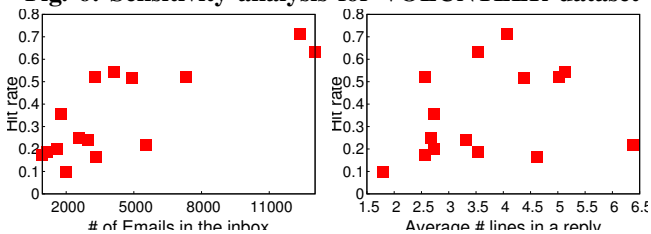
B. Evaluation:

We evaluated *DejaVu* prototype on 15 users from ENRON dataset and the first 6 users from VOLUNTEER datasets. For the VOLUNTEER dataset, the emails from the sorted list that are dated between January 2012 to December 2014 are used to populate the information database in *Information Curator*. All the other emails are processed in order. The reply text for the emails from the VOLUNTEER dataset was extracted by looking up $reply\_id$ in the database. The similarity (equation 1) between the reply text and the union of the suggestions is calculated. We process the ENRON dataset differently from the VOLUNTEER dataset due to the absence of 'In-Reply-To' field in the header, which prohibits easy access to the parent email. We further process the quoted text for these emails using same rules as in Section 2 to obtain the text of the parent email. We process the emails in sorted of date. If the email is not a reply, it is added to the database. If the email is a reply, the text of the parent email extracted from the quotes is used to lookup the database for suggestions. The similarity



(a) Similarity Threshold **(b) Suggestions**

Fig. 6: Sensitivity analysis for VOLUNTEER dataset



(a) Inbox size **(b) Reply size**

Fig. 7: Sensitivity to various parameters for ENRON

between the email text and the union of suggestions (from the quoted text) is calculated.

To evaluate the suggestion retrieval algorithm, we define a metric $HitRate$ for a similarity threshold to be the ratio of number of emails whose reply has a similarity greater than a threshold τ with the suggestions to the total number of emails with replies. A high value of $HitRate$ indicates that the suggestions were useful in writing replies. We evaluated $HitRate$ for at threshold $\tau = 0.6$ for both the 6 users in VOLUNTEER dataset (Figure 4a) and 15 users in the ENRON dataset (Figure 4b), for 1, 3 and 5 suggestions. For the VOLUNTEER dataset, the average $HitRate$ for 1, 3 and 5 suggestions was 0.33, 0.38 and 0.47 respectively. For the ENRON dataset, the average $HitRate$ for 1, 3 and 5 suggestions is 0.31, 0.42 and 0.51, respectively.

In other words for the case of 3 suggestions, on an average *DejaVu* was able to retrieve useful suggestions for one in 3 replies for both VOLUNTEER dataset and the ENRON dataset. For the personal email accounts in the VOLUNTEER dataset (users 2 and 4), the average $HitRate$ for 3 suggestions is 0.42. It is 0.28 for academic email accounts and 0.57 for the enterprise email account. The $HitRate$ for enterprise user is twice that for academic users. This is because in enterprises, email is the primary medium of communication and a large amount of information is formally shared between several parties through email. On the other hand in an academic/personal environment, face to face communication also plays an important role. These numbers clearly indicate the efficiency of the *DejaVu*'s suggestion algorithm in retrieving useful suggestions for a user.

C. Sensitivity Analysis

In this section, we evaluate the sensitivity of $HitRate$ to various parameters. Figure 6a shows the effect of changing the similarity threshold τ on the $HitRate$ for 6 users of VOLUNTEER dataset. The results for ENRON dataset show similar trends. As the similarity threshold τ is increased, the $HitRate$ falls for all users. This is because as τ increases, the threshold at which we decide whether suggestions are useful or not increases. On an average, increasing the similarity threshold from $\tau = 0.6$ to $\tau = 0.7$ decreases the $HitRate$ by 15.48% for users in ENRON dataset and by 29.54% for users in VOLUNTEER dataset. On the other hand, decreasing the similarity threshold from $\tau = 0.6$ to $\tau = 0.5$ increases the $HitRate$ by 18.01% for ENRON users and by 30.33% for VOLUNTEER users.

Figure 6b shows the variation of $HitRate$ to changes in the number of suggestions for 6 users in VOLUNTEER dataset. The results for the ENRON dataset are similar. In general, as the number of suggestions is increased, the $HitRate$ increases. Initially the $HitRate$ increases rapidly and then it saturates. This indicates that the text in the reply is only concentrated in a few emails in the mailbox and is not spread out across a large number of emails. Specifically, as the number of suggestions is increased from 3 to 5, the $HitRate$ increases by 31.26% for ENRON users and by 40.07% for the VOLUNTEER users. On the other hand, as the number of suggestions is decreased from 3 to 1, the $HitRate$ decreases by 26.72% for ENRON and 23.06% for VOLUNTEER datasets. For both these datasets, increasing the number of suggestions beyond 10 has little effect on the $HitRate$. From these figures, it can be observed that 3 would be an ideal number of suggestions as it is around the midpoint of the knee of the curve.

We also evaluate the sensitivity of $HitRate$ to the size of the Inbox and the average number of lines in the reply for both ENRON (shown in Figure 7) and VOLUNTEER datasets. However, we only present the results for the ENRON datasets in this paper. As the inbox size increases, the $HitRate$ increases for the ENRON dataset. With larger inbox sizes, there is more information available in the database for lookup, and hence a higher chance for finding the right suggestions for the replies. As the replies become longer (number of sentences in the replies increases), $HitRate$ generally increases (if a few outlier points are ignored). This is probably because for a larger reply there is more scope for a suggestion to be useful. To conclude, in general, larger inbox size and larger reply size tends to correlate with a larger $HitRate$. As more and more content is encountered in the mailboxes, the $HitRate$ is expected to improve for any user.

D. Examples

Table IV shows an example (from the ENRON dataset) of an inbox email, its corresponding reply and a snippet from the suggestion to that email. The first example is a meeting scheduling email sent to an executive, to which the reply is a confirmation email. One of the suggestion snippets for this was a sentence from a confirmation email from another meeting scheduling email from the past, where in the same executive asks the meeting be put on the calendar. From this example, we can see that *DejaVu* is able to suggest relevant snippets to the user, thereby reducing the burden on the user in typing these replies[8].

[8]We are unable to show any snippets from the VOLUNTEER dataset to preserve anonymity of the volunteers.

0.69	**Email:** Carol St. Clair asked me to schedule a meeting regarding the review of pulp and paper's confidentiality agreements. I have tentatively set it for Friday, September 10 at 10 AM. Let me know if this day and time works for you? **Reply:** works fine for me. **Suggestion:** Please put on my calendar

TABLE IV: Examples of email snippets

VI. Issues and discussion

In this section we present a few issues concerning *DejaVu* which we plan to explore as part of future work.

Matching algorithm: *DejaVu* uses an index constructed from the text of an email and its parent for finding relevant suggestions. A full email contains vast amount of other information (from the header) apart from text that could be leveraged to find better matches. A matching algorithm that considers not just the email text but also this other information and learns relative importance of matches of these various fields would be beneficial in retrieving better suggestions beyond simply matching keywords. For example, based on the intuition that emails having similar subject lines are very likely to have related content, and email conversations between the same set of senders/recipients probably are on the same topic, a classifier such as a Support Vector Machine or a Decision Tree Classifier can be trained with a set of inputs - text, subject line and sender/recipient similarity between an $email$ and an entry in the database, and a set of outputs - 1 if the entry is among the top few matches of $email$'s $reply$, 0 otherwise. The matches between different fields of a new email and an entry in the database can be fed into this classifier to find out if the entry is a suggestion or not. With recent advances in natural language processing techniques, specifically with better machine comprehension and question answering systems, better suggestions can be generated.

Knowledge Channels: *DejaVu* looks for suggestions to a $reply$ from information that is present only in an email inbox. However, a knowledge worker encounters several different knowledge channels each day. These channels could be categorized as read/write (Email), read only (Dropbox) or write only (Slack). By adding more information channels to the Information database of *DejaVu*, the suggestions for replies could be improved. For example, the documents from the user's Dropbox can be fetched using Dropbox APIs [14] by the *Information Curator*. Different topics in these documents can be added as entries into the *Information Database*. We plan to extend *DejaVu* to other knowledge channels in the future.

Evaluation: We only evaluate *DejaVu* offline by computing $HitRate$ for various parameters. However, the usefulness of these suggestions can be truly judged by real users using it on a daily basis. The usefulness can be captured through an opinion score metrics, where in the users rate every suggestion using a score of 1(not helpful) to 5(very helpful). In the future, we plan to evaluate *DejaVu* by distributing a production version of the prototype to a large set of volunteers and capturing subjective metrics. However, $HitRate$, an objective metric is a valuable metric to evaluate *DejaVu* on a higher level.

Search expansion: In English, a word can have several synonyms and can be present in different forms. *DejaVu* only deals with the latter by stemming the word and extracting its root. The former problem could be solved by expanding the index used for matching to include all possible synonyms for the words encountered in that index (obtained from a resource such as WordNet [15]). This way, suggestions containing words that have different roots but similar meaning can be retrieved as matches.

VII. Related Work

Email Optimizations: The problem of information overload in email was first recognized in [16] in 1996 and was validated a few years later by [17], [18]. Several solutions have been proposed to optimize email to combat email overload. [19]–[21] suggest intelligent categorizing techniques to manage information efficiently. Few works used content summarization techniques to extract summaries from email [22]–[25], to be used for better presentation of email lists. [26]–[28] identify certain speech acts in email such as - statement, request, propose (meeting), amend, commit, deliver .etc, to better help the user track the status of an ongoing task. Few other solutions prioritize each email as being important or not to help user quickly deal with and respond to a large inbox [29], [30] Just like the semantic web, semantic email has been proposed by [31], [32] where in each email is tagged with certain semantic information that can be leveraged at a later stage for context specific applications. Apart from these solutions, all email clients provide a search feature to retrieve relevant information easily in an overloaded inbox. There are about a 900+ startups working on optimizing various aspects of email and providing new features for increasing productivity [33].

Knowledge Management: Enterprise worker is typically exposed to several knowledge sources during his work day. Due to the diverse nature of types of information (documents, emails, IM etc.) and the applications (Dropbox, Gmail, local storage, Slack etc.), it is hard for the typical enterprise worker to quickly retrieve relevant information. Several solutions exist today [34]–[37] that provide an unified index on these various sources of information and thereby helping the user retrieve relevant information in a timely fashion. However, these solutions do not specifically focus on automatically helping users construct email responses.

Reply Prediction: [25], [38] have explored the idea of predicting whether the email needs a reply or needs an attachment. However, they do not predict the content of the reply or which attachment to include. Some works [39]–[41] have explored identifying experts through email conversations. This information is very useful and can be used to direct the conversation on a topic towards the expert and elicit responses. However, these works do not provide a way to lookup information that is potentially available in the user's own inbox and construct responses from there. The closest

related work to *DejaVu* is Smart Reply on Google Inbox. Smart Reply [6] suggests responses to emails based on their content. It encodes the Inbox email through a recurrent neural network and extracts context out of it. This context is then used to predict coherent responses from another recurrent neural network. These recurrent neural networks are trained on the user's inbox. However, smart reply only constructs generic email responses (and not content specific responses) and focuses on grammatical correctness and cohesion of these responses. *DejaVu* on the other hand focuses on suggesting content specific information bullets.

VIII. Conclusions

In this paper, we explore the problem of automated information suggestions to assist in $reply$ construction for Email on mobile devices. The basic premise of the work is that a significant portion of the information content of a $reply$ is likely to be present in prior emails. Through analysis of multiple Email datasets, both public and private, we first establish that there indeed is considerable redundancy between $replies$ and prior emails. We then present a simple solution called DejaVu that provides automated email suggestions during $reply$ construction. When applied to the same datasets, we show that DejaVu shows a lot of promise . We discuss several issues with DejaVu that we intend to explore in future

References

[1] "The ultimate mobile email statistics overview," http://www.emailmonday.com/mobile-email-usage-statistics.

[2] S. Radicati, "E-mail statistics report, 2014–2018," *Paolo Alto: The Radicati Group Inc*, 2014.

[3] "The social economy: unlocking value and productivity through social technologies," http://www.mckinsey.com/industries/high-tech/our-insights/the-social-economy.

[4] F. Kooti, L. M. Aiello, M. Grbovic, K. Lerman, and A. Mantrach, "Evolution of conversations in the age of email overload," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. New York, NY, USA: ACM, 2015, pp. 603–613. [Online]. Available: http://doi.acm.org/10.1145/2736277.2741130

[5] "Smartphone keyboards vs. your productivity," http://www.informationweek.com/mobile/mobile-devices/smartphone-keyboards-vs-your-productivity/d/d-id/1102283?

[6] "Computer, respond to this email," http://googleresearch.blogspot.com/2015/11/computer-respond-to-this-email.html.

[7] "What's new in ios," http://www.apple.com/in/ios/whats-new/.

[8] "The enron email dataset," https://www.cs.cmu.edu/~./enron/.

[9] "The hillary clinton email dataset," https://www.kaggle.com/forums/f/15/kaggle-forum/t/16444/hillary-clinton-email-dataset.

[10] "talon," https://github.com/mailgun/talon.

[11] "Dropping common terms: stop words," http://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html.

[12] K. Sparck Jones and P. Willett, Eds., *Readings in Information Retrieval*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.

[13] "K-9 mail," https://play.google.com/store/apps/details?id=com.fsck.k9&hl=en.

[14] "Dropbox api," https://www.dropbox.com/developers.

[15] "Wordnet. a lexical database for english," https://wordnet.princeton.edu/.

[16] S. Whittaker and C. Sidner, "Email overload: exploring personal information management of email," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1996, pp. 276–283.

[17] D. Fisher, A. J. Brush, E. Gleave, and M. A. Smith, "Revisiting whittaker & sidner's "email overload" ten years later," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, ser. CSCW '06. New York, NY, USA: ACM, 2006, pp. 309–312. [Online]. Available: http://doi.acm.org/10.1145/1180875.1180922

[18] L. A. Dabbish and R. E. Kraut, "Email overload at work: An analysis of factors associated with email strain," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, ser. CSCW '06. New York, NY, USA: ACM, 2006, pp. 431–440. [Online]. Available: http://doi.acm.org/10.1145/1180875.1180941

[19] R. Bergman, M. Griss, and C. Staelin, "A personal email assistant," 2002.

[20] K. Mock, "An experimental framework for email categorization and management," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 392–393.

[21] S. Whittaker, T. Matthews, J. Cerruti, H. Badenes, and J. Tang, "Am i wasting my time organizing email?: a study of email refinding," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 3449–3458.

[22] G. Carenini, R. T. Ng, and X. Zhou, "Summarizing emails with conversational cohesion and subjectivity." in *ACL*, vol. 8, 2008, pp. 353–361.

[23] ——, "Summarizing email conversations with clue words," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 91–100.

[24] S. Joty, G. Carenini, G. Murray, and R. T. Ng, "Exploiting conversation structure in unsupervised topic segmentation for emails," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 388–398.

[25] M. Dredze, H. M. Wallach, D. Puller, and F. Pereira, "Generating summary keywords for emails using topics," in *Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 2008, pp. 199–206.

[26] W. W. Cohen, V. R. Carvalho, and T. M. Mitchell, "Learning to classify email into"speech acts"." in *EMNLP*, 2004, pp. 309–316.

[27] V. R. Carvalho and W. W. Cohen, "On the collective classification of email speech acts," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 345–352.

[28] S. Joty, G. Carenini, and C.-Y. Lin, "Unsupervised modeling of dialog acts in asynchronous conversations," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 3, 2011, p. 1807.

[29] D. Aberdeen, O. Pacovsky, and A. Slater, "The learning behind gmail priority inbox," in *LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[30] "Priority inbox," https://support.google.com/mail/answer/186531?hl=en.

[31] L. McDowell, O. Etzioni, A. Halevy, and H. Levy, "Semantic email," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 244–254.

[32] S. Scerri, B. Davis, S. Handschuh, and M. Hauswirth, "Semanta–semantic email made easy," in *The Semantic Web: Research and Applications*. Springer, 2009, pp. 36–50.

[33] "Email startups," https://angel.co/email.

[34] "Coveo," http://www.coveo.com/.

[35] "Sinequa, from enterprise search to real time big data search and analytics," http://www.sinequa.com/.

[36] "Attivio, the data dexterity company," http://www.attivio.com/.

[37] "Hp autonomy," http://www.autonomy.com/html/promote/search-rescue/html/search_rescue/fast/index.html.

[38] M. Dredze, J. Blitzer, and F. Pereira, "Reply expectation prediction for email management." in *CEAS*, 2005.

[39] C. S. Campbell, P. P. Maglio, A. Cozzi, and B. Dom, "Expertise identification using email communications," in *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 2003, pp. 528–531.

[40] K. Balog, L. Azzopardi, and M. De Rijke, "Formal models for expert finding in enterprise corpora," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 43–50.

[41] J. Zhang, M. S. Ackerman, and L. Adamic, "Expertise networks in online communities: structure and algorithms," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 221–230.