Enhancing Unified Communication Services with Communication Context

Krishna Kishore Dhara, Venkatesh Krishnaswamy, Eunsoo Shim, Xiaotao Wu*
IP Communication Research Department
Avaya Labs Research, Basking Ridge, NJ, USA
Email: {dhara, venky, eshim, xwu}@avaya.com

***Abstract*— Unified communication services aggregate user's communication sessions and data across various modalities. While this aggregation provides users easy access to their data, the complexity of organizing, analyzing, and searching the data in real-time is left to the user. Our work focuses on an approach that analyzes and correlates users' communication data to their communication context. We use this communication context to realize rich novel unified communication services. Our approach identifies common threading and topic correlations across user data, and proposes a framework that supports rapid creation of such new services. We present several novel services and discuss services specific algorithms and offer our recommendations. The framework and the services described in this paper have been deployed for trial in an enterprise environment.**

I. Introduction

Unified communication services focus on enhancing end-user communication experience by bridging multi-modal communication sessions with their context and their multiple devices and also by adding new capabilities such as multi-party audio, messaging, and video conferencing and integrating their online social networking (OSN) relationships. Those features may be enriched with context that primarily focuses on user presence, location, OSN activity, and preferences.

In this paper we take a complementary approach to enhance unified communication experience. We focus on a class of services that go beyond the aggregation that is commonly found in the current set of unified communication services. The new class of services are based on the context information constructed from analysis of the user's communication data. The analysis involves learning the user's behavior and service usage patterns and predicting user's interest and need with respect to communication services. We present a set of end-user communication experiences as examples and motivations for the new class of context-based communication services. We also present details of a framework that enables such context-based communication application. Though we use our framework only for illustrative purposes, it highlights and addresses several crucial and interesting challenges for developing context-based communication services.

*Authors in alphabetical order with equal contribution

Manuscript received February 15, 2011; revised May 15, 2011; accepted June 15, 2011

Analyzing communication data is non-trivial. Communication data has several fundamental characteristics that distinguish it from traditional document analysis [1] [2] and from the analysis of content and content flows in social networks [3] [4] [5]. One distinguishing feature is that communication content is spread out and highly contextual. That is, communication data, unlike a document, may not have the entire context within one unit of communication message or session. The context information could be spread over several emails, calls, or events. Further, the corrletion among communication data could be subtle and indirect. An email referring to an earlier email stating that "Yes, here are my thoughts on that" should be associated with a thread containing the referred email. Threading such discrete parts of communication data is crucial in context-based communication services. These threads are not just email threads but across different modes such as emails, calls, and conferencing events.

A second feature that sets analyzing communication data apart is the dynamic nature of user and participant relationships and its dependence on it. Unlike online social networking, user relationship with participants in communication sessions is not pre-determined as a "friend" relationship and varies over time. Context-based communication services have to extract relationships among people from communication data and augment that with a notion of 'thread' and a 'topic' to group communication data. We explore these further in the rest of the paper. Another feature that is crucial to context-based communication services is data sharing and its association to users. While document repositories have policies for sharing and privacy controls that could be extracted while analyzing data, communication data that flows from users to users have a different notion of information sharing. Further, this information sharing is different from the information diffusion in online social networks [4]. The differences are both in the content and in the content flow. In online social networks, a subset of the content that a user intended to publish is available to a node in their friend graph. In communication networks, analysis at a user can combine information from multiple sources and piece together a context from the communication data available at its node. Also, the result of this analysis is private to each individual user.

While some of these individual services along with

doi:10.4304/jcm.7.2.115-131

the learning and prediction algorithms are presented in our earlier papers [6] [7] [8] [9] [10], this paper provides a comprehensive view of how these communication context-based services could be applied to enhance unified communication experience. We discuss the challenges that need to be addressed and present a framework that will enable rapid development of such context-based applications. We present data from a deployed trial server on an enterprise communication system with real users.

In the next section, we define what we mean by communication context and present a unified communication experience with examples. In Section III we discuss related work. Section IV we discuss our framework to realize communication context based services. Section V discusses common analayzers that capture structural and lexical content of communication data and Section VI discusses services along with their algorithms and results. The last section presents our conclusions.

II. CONTEXT-BASED COMMUNICATION EXPERIENCE

In this section, we define the term context in communications. Using this notion, we present a series of novel communication services that enhance users' unified communication experience. We then discuss challenges for such context-based communication services.

A. Context in Communication

1) Defining communication context: In communication systems the term context represents several things that are related to users, such as user's presence and availability information, location information, data obtained from sensors or end-user devices that track their behavior, and information obtained from user's online activity. While the above listed information are important sources of contextual data, they don't include users' communication activities, in particular, the most important aspects of human communications – users' identities, contacted persons' identities, and the content of communication sessions. These information may provide significant and direct insight on the need and interest of the user. Hence, along with location, sensor and presence data we extend the notion of context to include a user's communication activities such as the participants in communication sessions, the content of communication, and the modalities of communications. In addition, users' communication history or scheduled future communication plan can also become their context for current and future communications, either in individual or group form. We call this kind of context information *communication context*.

In our work, we extract users' communication context and process the context information by mining, analyzing, and finding correlations of users' data across different communication sessions and across different modalities and data sources. We use users' communication context to predict users' present or future behavior with respect

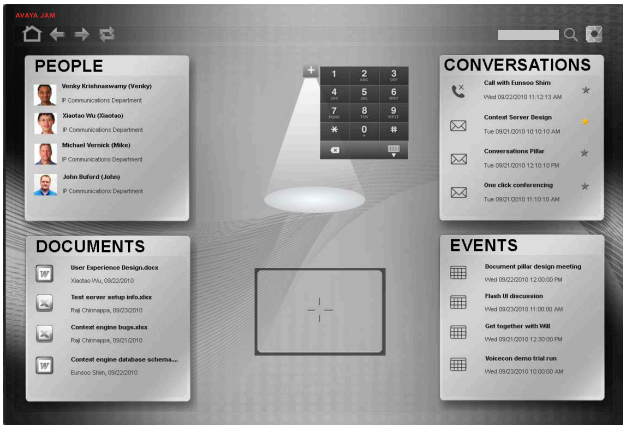


Figure 1: A unified user experience with communication context based services

to communication services and enable enhanced unified communication services. This communication context is complementary to location, sensor, and other contextual data. Though we discuss services and new user experiences that are based on communication context as defined above, we can further enrich these services by adding location information, presence, and other information. However, to highlight the challenges and to discuss various algorithms, their impact, and their performance, we omit any discussion of integrating additional contextual information and focus only on communication context.

2) Enhancing User Experiences using Communication Context: Currently, unified communication services for users are limited to integrating contacts from several sources such as Outlook, GMail, and Facebook, or providing an integrated mailbox for various media from different service providers [11], [12]. While these unified communication services integrate multiple tools and services, it is left to the user to sort out relevant data for their current activity. Our goal is to develop communication services that go beyond such integration and look at the current activity of the user, and present relevant data required for that activity. We use communication context to predict relevant data that is required for users' current activity to enhance their experience. Figure 1 shows our context-based communication client that can present the predicted relevant data. It contains a list of relevant contacts in the top left as `People` service, relevant communication sessions in the top right as `Conversations` service, relevant documents in the bottom left as `Documents` service, and relevant events, including appointments, miss calls, and notifications, in the bottom right as `Events` service. At any given moment, based on communications history, the `People` service predicts the contacts that a user might want to reach. Similarly, the `Conversations` service identifies most relevant conversations the user might want to review, so as for the `Documents` and `Events` services. The content shown in Figure 1 is the default screen with these services. In default screen, there

Figure 2: User experience of a couple of communication context based services on an incoming call

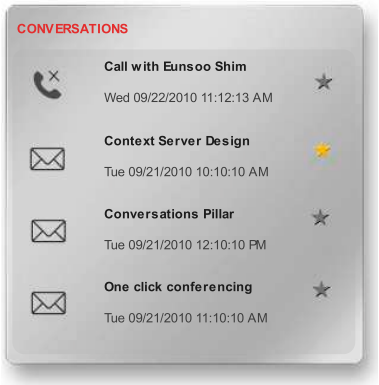


Figure 3: Rich Communication Log

is no explicit user communication activity, such as an active call or scheduled appointment associated with the service results.

While upon receiving an incoming call, the above mentioned service results will be updated by re-computing relevance values based on the communication context of the call, such as the caller's identity, the topic of the call, the time of the call, the priority value of the call, as well as the user's communication history. Figure 2 shows the changes in two of those services, namely `People` and `Conversations`.

In this paper, we use the user experience and the context based services in Figure 1 and Figure 2 for illustrative purpose only. That is we use the user experience described in the figures to understand the nature of a class of services that rely on communication context. In the next subsection, we describe some of these services to highlight the enhanced user experiences and the challenges of providing such services. We also use these examples to understand the commonalities across various services; for example, communication threading, topic modeling, etc., and a service specific notion of 'relevance'.

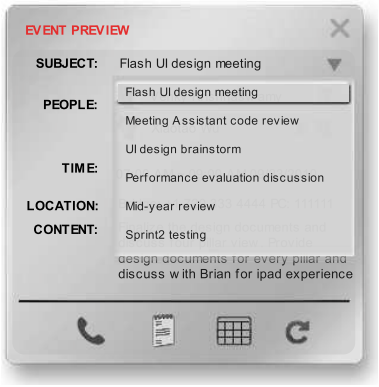
B. Context-Based Communication Services: Examples

In this section, we focus on individual communication services to understand the common structural and semantic building blocks and service specific communication context elements. We discuss some of these examples in detail in later sections.

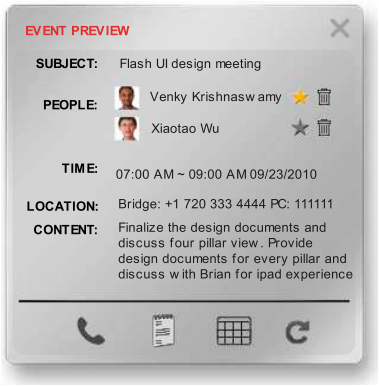
1) Rich Communication Log: Rich Communication Log is a service that collects users' communication session information and presents the information in user customizable views under different communication contexts. An instance of Rich Communication Log is shown as `Conversations` in Figure 1 on the top right and in Figure 3. As shown in the figure, the communication sessions can be voice or video calls, instant messages, emails, voicemails, and short voice and video messages (SVVMS). These customizable views can be any combination of different sorting and grouping mechanisms, such as sorting by time, importance, relevance, or people, and grouping by topic, thread, or session status. The content of Rich Communication Log varies in different communication contexts. These communication contexts can be pre/post-communications, incoming/outgoing sessions, ad-hoc/meet-me meetings, or on-demand queries. At first glance, Rich Communication Log looks similar to other unified communication inbox [11] [12], but its underlying logic is intrinsically different from existing unified communication applications. First, it updates its content based on context-based computing. Context related information can be time, session participants' identifications, ongoing events, or on-demand queries that are based on user's current activity. Secondly, it inter-connects communication sessions with different modalities by learning their semantic meanings, temporal relationships, and associated user activities.

2) Related Documents Service: The goal of the `Related Documents` service, shown as `Documents` in Figure 1 on the lower left, is to present the documents a user is likely to view in the near future. When the user is in an active communication session like a phone call or a meeting, the `Related Documents` service aims to present the documents the user needs to review with the other participants of the session. Realization of the service requires tracking the documents exchanged between the user and her contacts such as email attachments, documents or their URLs embedded in the email bodies and meeting invitation messages. The documents should be ranked based on the likelihood by which the document will be viewed by the user. The likelihood depends on many factors such as whether the documents are references for the meeting that is going on or will start soon, whether the documents were exchanged with the remote parties of the current call, whether the documents were included in the recent emails, whether the documents are related to the topic (subject) of the upcoming meeting, and whether the documents are related to the topic on which the user is actively sending or receiving emails.

3) Predictive Meeting Assistant: The central idea of the Predictive Meeting Assistant is to offer a service that predicts a user's meetings with other users and to provide a mechanism to set up that interaction easily. For example, the current practice in setting up a meeting is to first open a calendar system that has access to calendar



(a) A predicted meeting choices



(b) Predicted participants in a meeting

Figure 4: Sample scenario of predictive meeting assistant.

information of all the participants in the meeting. Then users have to choose each participant manually and select a suitable time for everyone. Further, users need to enter the text, subject or reason for the meeting, and attach any URLs or documents for information sharing during meeting. Additionally, the user has to add conference bridge information if the meeting is a teleconference.

We approach meeting prediction service as a communication activity that has three stages, a pre-session activity where meetings or communication sessions are set up, an in-session activity, and a post-session activity. In the pre-session stage, when a user clicks on the 'schedule a meeting' option on their collaboration or communication interface, meeting prediction service pops up a window that has a several options such as "Subject", "People", "Time", "Location", and "Content". Figure 4 shows a couple of these options for a sample use case scenario. Figure 4 (a) shows the predicted meeting subjects for user as a drop down list. Based on those choices predictive meeting assistant would pre-populate other fields automatically. For instance, if the choice is "Flash UI design meeting" as shown in Figure 4 (b), then the Predictive Meeting Assistant will populate the participant data, location, audio/video bridge information, and possible content. Such a service enhances productivity among enterprise or business users.

In the in-session stage, once a user, the meeting organizer or a participant, joins the meeting, the Predictive Meeting Assistant will find the relevant documents, URLs, and other related people utilizing other communication context based services described herein. The post-session activity varies for different context-aware communication services. The predictive meeting assistant allows notes of the meeting to be taken or attached, notifies all the participants, and uses that to analyze and predict future meetings.

4) Predictive Contacts: Predictive contacts is a service that predicts and ranks the people a user may need to contact at any given moment. The `People` depicted in Figure 1 is an instance of predictive contacts. This service is quite different from the existing contacts and directory applications. Existing contacts applications allow users to maintain their preferred contacts as a user configured list and enable a lookup mechanism that is initiated by the user. Similarly directory services allow users and applications to search in an enterprise database of people. Both are static and the lookup depends on the user's query or search. For instance, on an impending conference call or session, to contact the organizer, the user has to remember who the organizer is and where (in local contacts or in directory services) to find the organizer details. Similarly in enterprise a user's preferred contacts often change based on pending events, group activity such as incoming or outgoing communication sessions and the participation levels in such sessions. Further, the preferred contacts also change based on the current activity. For example, when the user receives a call, the Predictive Contacts service aims to show only the contacts that are relevant to the user and the caller.

To predict and rank the people a user might interact with based on their communication context requires analysis of communication data both at the structural level, such as mean arrival rates, mean response times, event times, etc., and semantic notions such as topics common among users and other people, activity in threads that span across various medium such as emails, calls, etc. Predictive contacts is enabled through the rich communication context mined from users' communication data.

C. Challenges for Communication Context Based Communication Services

To realize communication context based services described in the previous section several challenges need to be addressed. Some of these challenges though generic in nature have some specific communication context related challenges. One such generic challenge is the scalability. The challenges in scalability are in contrast to recommender systems, where the model analyzes millions of users data to apply the learned model on users. The scalability challenge in communication context is the learning model from the data is applied only to that particular user. Consider the number of communication sessions, emails, instant messages, and meeting appointments for

each user, scaling the framework to support real-time applications is non-trivial. Similarly privacy is restricted to using the learned model within the scope of one user. It is a challenging problem to maximize the usage of user's information while keeping the security and privacy boundaries.

We focus on context-based communication specific challenges. We categorize these challenges as follows.

1) Identifying Session Relationships: We need to find a way to detect the implicit relationship among communication sessions with different communication channels. For example, it is very common that a user uses an email or an instant messaging (IM) to initiate a discussion, then escalates the conversation to a voice and video call, and in the meantime, the user may use emails to send documents. The IM session, voice and video call session, and emails are all related to each other. This relationship should be detected and used to prioritize sessions, correlate communication context, and provide enriched information.

2) Inferring User Relationships: Relationships among participants in a communication network, like in online social networks, affect the behavior of users. While the user relationships in online social networks are explicit and the analysis is focused on weak and strong ties[2] and on information diffusion [1], the user relationships in communication networks could be explicit or implicit. The explicit relationships come in the form of a user configured contact or a friend, and initiating a communication sessions or a collaboration session with that contact. Unlike online social networks, communication networks allow session sharing and collaboration with participants that are not in the contact list, hence learning the user behavior with respect to these relationships, which are implicit, is important. Further, the nature and hence the behavior of user-participant relationships in communication networks depends on various aspects such as topics or people that are important to the user or triggering events, such as an approaching event, is important to the user-relationship. Analyzing past behavior and modeling these user-participant relations per user is a key to enabling services such as predictive meeting assistant.

3) Real-Time Temporal Relationships: Communication data is generated continuously. The topics the user works on and the people she interacts with change over time and there are not only long-term topics and relationships but also short-term, even transient ones. The relevance of data fades in a shorter time scale than, let say, ecommerce. The user's situation may change any time and the recommendation for the communication services may have to be completely different. So the recommendations must be refreshed continuously based on the latest data.

4) Extracting Lexical Relationships: While there is extensive research in extracting relationships from documents, communication data presents several interesting challenges.

- Compared to text documents, journal and news articles, communication data, such as email data, call data, event information, etc., contain low lexical content and high noise.
- Communication between a set of users tend to rely on previous context of communication and not to explicitly contain any contextual information. For example, an email or a call that is a follow up of another email or a conference call relies on prior communication context.
- Based on the detected relationship, we need to define an appropriate way to correlate messages with different communication modalities. For example, if a voice call is related to an email, we may infer the call's semantic meanings based on the text of the email. These semantic meanings can help to prioritize the call.

In the rest of the paper we present our approach to analyze and learn communication context from users communication data. We address challenges discussed in this section, present our learning algorithms, and discuss details of some of the communication context based services discussed in this section.

III. Related Work

The term 'context-aware' was first introduced by [13], where the context was referred to location, identities of nearby people and objects, and changes to those objects. The definition of context has expanded to location, identifies of the people around the user, the time of day, season, temperature, etc. in [14]. The scope of the context examples further expanded to the user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the user's environment [15]. A general form of the definition of 'context' is provided as any information that can be used to characterize the situation of an entity [16]. The communication context defined in the previous section falls within the scope of this generic notion of context.

Our work on unified communication services with communication context relates to several areas of active research. In the rest of this section, we relate our work with each of these systems. We limit our discussion on complimentary approaches to context such as location, presence, etc and instead focus our discussion on the relation with unified communication applications, recommender systems, data mining and machine learning systems, and OSN applications.

The Conference Assistant [17] is an application that combines 'context-awareness' with wearable computing, aiming to aid the attendees of conferences. The application uses the identity of the user and the location as the context in the conference setting and maps the location and time to the conference program. It provides information of the presentation at the time at the proximate location to the user automatically. It has the feature of recommending presentations that may be interesting to the user. Our work is primarily focussed on communication

data and is orthogonal to location and other such context information.

CyberDesk [18] knows the application a user is working with and the data (both type and content) the user is interested in (via explicit selection with the mouse). These applications primarily look at user's location and the current document the user is looking at and the generated recommendations are largely static mapping of the location and the current document's content to a set of choices. User's communication data is mostly out of scope and their system does not handle the dynamic nature of communication data and the embedded documents.

Recommender or recommendation systems attempt to recommend information items that are likely to be of interest to the user. The examples of the information items are TV programs, movies, music, news, books, web pages, etc. The recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. The commonly accepted formulation of the recommendation problem was first stated in [19]–[21] and this problem has been studied extensively since then. The recommender systems are classified into three categories based on how recommendations are made [22]: demographic-based filtering, collaborative filtering and content-based filtering. The main differences between context-based communication services and recommender services are the scope of the learned model (one user versus all users), lack of any user input, and the dynamic and real-time nature of the content.

Traditional unified communication applications focus on integrating different communication channels, such as voice, video, instant messaging, email, voicemail, and presence, into one inbox. However, as more communication channels integrated and the number of messages rapidly grows in people's inboxes, people require a more intelligent and efficient way to organize their inboxes. Recently, Goolge's email service, Gmail, released a new feature called Priority Inbox [11]. Gmail's Priority Inbox tries to identify users' important emails so users can handle important messages first. This feature is still in its beta stage but shows a useful concept of helping users to organize their inbox.

Another example of providing intelligence in users' inboxes is an application called Xobni [12]. Xobni create rich profiles for every person has communicated with. The profiles include communication statistics, social connections, and communication histories. For an email, Xobni can automatically show people profiles related to that email.

Gmail Priority Inbox and Xobni are two complementary applications. One reduces information to avoid distractions, while the other enriches information to improve productivity. In our Rich Communication Log, we apply the ideas of reducing unnecessary information while enriching relevant information to sessions in a unified inbox. We also present the information in a user customizable way. Gmail Priority Inbox and Xobni only work for text-based emails, applying similar idea to a unified inbox with different communication modalities needs to overcome the many challenges.

In OSNs, the user social graph is explicit and in communication networks the connectivity is dynamic. Further, the focus in OSN applications is to understand information diffusion [4] and the roles of individual nodes [3]. The flow of information in a social network from a node, A, to another node G, is a subset of information published by A. That is the information is propagated from $A \implies G$. However, for communication applications the relevant information is based on the communication data available at each node and is independent of information at A or at other nodes in the graph. The view and hence the mined communication context is based on data at each node and unlike social network it is not based on propagation [9]. This is a subtle but a crucial distinction in learning communication context and for deploying new communication services.

Our work on context-based communication relies heavily on analysis of communication data with both unsupervised learning and supervised learning. We use topic modeling as an unsupervised learning technique to mine correlations across a user's communication data. Topic modeling based on Latent Dirichlet Allocation [23] has been used for searching large volumes of data, for ad-hoc retrieval, for finding relations across a network of documents and for many other applications [2], [24]. Griffiths and Steyvers [2] use topic modeling to find scientific topics from a large corpus of scientific literature (PNAS). They uncover correlations across various documents and different science disciplines. Chang and Blei [25] use a hierarchical topic model capture to relationships across documents based on their content. Their work is aimed at deriving explicit ties between documents and the connections between them. Wei and Croft [26] uses LDA for ad-hoc information retrieval. These works primarily focus on documents that are in repositories. Unlike communication data, these documents are static and are rich in lexical content.

Yan and et. al [27] use topic modeling to analyze social networks and blogs. Their goals are to identify the topics being discussed in blogs and to identify the social network of authors in the social networks. They extend LDA by introducing an author and link formation model that shares the same Dirichlet parameter for inference. Topic modeling is used to discover roles in social networks by McCallum et. al [5]. They model author-recipient-topic relationship as a Bayesian network that discovers discussion topics in social networks. Our approach to integrating user-people relationships is different. We use LDA as an atomic operation to obtain lexical topic relationships and augment it with user-people topic relationships. Though the user-people topic relationship learns from the lexical topic modeling, it is kept separate. This enables us to model user-people topic relationship based on other factors such as medium (calls, events, etc), user defined

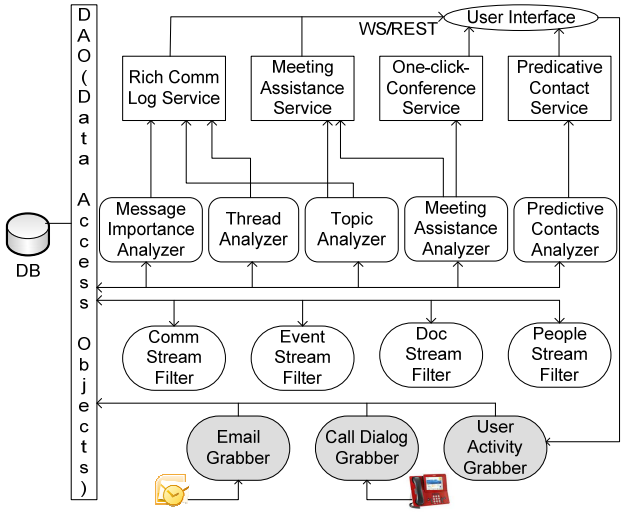


Figure 5: A framework for mining and analyzing communication context

topics, and user preferences.

In the application space, our work extends unified communication by integrating various communication modalities and by bringing relevant information such as people, documents, and communication logs automatically to a unified communication session. Futher, instead of integrating the relevance into a single unified communication service, our framework allows applications to consume the 'relevant information' computed by service analyzers. We argue that the challenges and the scope of class of communication services, such as `meeting prediction assistant`, `rich communication log`, we discuss in this paper are quite different from inferring user activity and context in online social networks. In later sections, we discuss enchancements that allow us to use techniques such as LDA [23] for mining context in communication data. These techniques consider the relationships among people, which are an important factor for setting priorities of communication documents/events and the history of communication events, which are significant for identifying people relationships.

IV. FRAMEWORK FOR ANALYZING COMMUNICATION DATA

In this section we describe a framework for analyzing communication data. Figure 5 shows the overall architecture of our system that supports communication context based services. The lower tier represents grabbers that collect raw information from various communication data sources. These grabbers fetch email data from email servers, such as exchange or gmail servers, call and conference data from communication servers, and user activity from end-points. After fetching the data they pass the data through a series of filters. These filters perform some initial processing on raw messages, such as spam filtering, privacy handling, and keyword extraction. While most filters are quite generic and the generated data are used by many components in the systems, there could be some application specific filters. An example of that would be the meeting prediction filter (to be covered in Section VI) that looks at individual mails to tag them if there is a meeting intention using a classification algorithm. This lower tier addresses general challenges such as scalability, privacy, and modality. While scalability is a continuous effort, we address this by incremental nature of the grabbers and invoking filters only on new communication data. We also ensure that the grabbers and hence the filters run only periodically. As the grabbers and filters fetch and analyze data they view the data for each user and mark all the people that the user communicates as persons related to that user. Privacy is ensured with a clear distinction between user identities from person identities that are related to a user. This distinction results in duplicity of data but guarantees privacy. For instance, an email or a conference call between three participants is grabbed individually for each of the three participants and the meta-data about the email or call is stored separately in each participant's data. Thus three emails (or call) entries will be produced, one for each participant. Each entry will have one user as the owner of the entry and the other users as the participating persons related to that communication. This owner identity sets the boundary for analysis of communication data and the system never crosses the owner boundary for any structural or semantic analysis of data.

Communication data are associated not only to a user but also to each participant of the communications, which requires associating communication identifiers such as email addresses and phone numbers to people. There are external and internal sources for the association. The corporate directory, public whitepages services, and public web pages such as people's homepages are examples of external sources. Email text, in particular, email signatures are an example of internal sources. We incorporated the corporate directory for the implementation presented in this paper.

The second tier in the framework consists of analyzers and the main intelligence of the system resides in these analyzers. There are two kinds of analyzers. One set of analyzers, such as thread analyzer and topic analyzer, capture the common data analysis. They mine structural, lexical, multi-modal session relationships, and user relationships from the communication data of a user. These analyzers generate meta-data related to a communication session or a communication messages (email, IM, etc), which could be used by applications. These common analyzers address concerns listed in Section II such as prioritization, determining appropriate classification, user relationships, etc. The other set of analyzers are specific to an application. An example of this is the predictive contacts analyzer, which is often associated as part of a unified communication service visible to users. New

communication context based services can be developed by using components of this framework at various levels. The following list presents the communication services described in section II and section VI in terms of their usage of various components in the framework.

1. Rich Communication Log: Rich communication log relies on common analyzers such as thread and topic for meta-data on emails, calls, and other communication data and uses its own application specific analyzers.
2. Predictive Contacts Service: Predictive contacts service uses data from the filters and uses its own analyzer to compute communication parameters such as mean arrival times, mean response times, their variance, etc., for its prediction.
3. Predictive Meeting Assistant: In addition to standard filters, meeting prediction has its own filter to determine if communication document indicates the possibility of a meeting. It then uses thread (and topic) analyzer(s) to correlate topics and participants.
4. One-Click Conference: One-click conference relies on the Event Grabber and a filter to determine the participant information and learn the meeting logistics such as access numbers for the conference bridge.

Note that while services (1-3) rely on common meta-data, service 4 presents an example that does not need to rely on any meta-data but derive its own meta-data or communication context based on grabbers and filters. The above list shows that new applications with different communication context criteria could be developed using such a framework.

In the rest of this paper we focus on these important components that analyze and extract common and application specific communication context. First we present in detail the common analyzers, thread analyzer and topic analyzer. Later, we present detailed algorithms and framework components of some of these novel communication context based services. The results presented in the discussion are from real users on a deployed enterprise server.

V. MULTI-MODAL STRUCTURAL AND LEXICAL ANALYSIS FOR COMMUNICATION CONTEXT

A. Analyzing Threads Across Communication Data

We use the structure of various communication sessions and correlate them as a thread to find relation among a user's various forms of communication. In particular, thread analysis captures relationship among sessions based on message identities, user activities, session participants, and session temporal relationships. The relationship can be invoking, containing, triggering, continuing, replying and forwarding. The result of thread-based analysis can be used by other components, such as topic analysis, and services, such as rich communication log and predictive contacts. First, we describe several causal relationships among communication session to identify threads and then describe the thread analyzer in detail.

1) Communication session relationship:

a) *Invoking relationship* Invoking relationship means that a user starts a conversation from the context of another conversation. For example, a user click-to-call a phone number in an email, we consider that the email and the call are in the same thread and have the invoking relationship. To detect invoking relationship, the system needs to track user activities. As shown in Figure 2, in our system, we use a module called "User Activity Grabber" to catch user inputs. The grabber detects user communication actions, such as 'click-to-call', and save the relationship into database.

b) *Triggering, Containing, and Continuing relationship* Both triggering and containing relationship try to find out the temporal and participant relationship among sessions with different communication channels. Triggering means that a user starts one session because he/she just accessed another session. For example, a user A receives an email from another user B. After reading the email, A immediately makes a call to B. Another example, A just finishes a call with B. Immediately, A sends an email to B with some documents. In both examples, we consider the call and the email are in one thread and with triggering relationship. Containing means that during a conversation (usually a voice/video call), the participants use other channels (usually IM or email) to communicate. The new session is contained in the original session and should be considered in one thread. We use the following code to detect these two relationships:

```
if (overlap_prtcipants_noself(s1, s2))
{
  //Ts-starting time, Te-ending time
  tr = threshold;
  if (Ts(s1)>Te(s2)
       && Ts(s1)-Te(s2)< tr) {
    s2 triggers s1;
  }
  if (Ts(s1) < Ts(s2) < Te(s1)) {
    s1 contains s2;
  }
}
```

Continuing means that one session will immediately follow another session with same participants and same media type. This is usually used for voice/video calls, for example, when a call is dropped and immediately reconnected.

c) *Replying and forwarding relationship:* Replying and forwarding simply inherit the thread definition from email systems. This can be easily handled through a filter that handles emails. For call history, if we can identify that one call is the callback of another call, these two calls also have "replying" relationship. Certainly that only relying on these relationships to

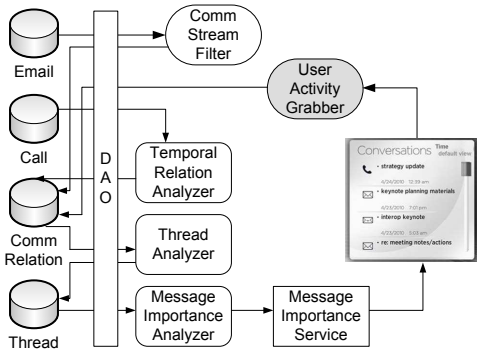


Figure 6: Data flow in Thread Analyzer

automatically identify threads may still have errors. But this can serve as a start point and people may further manually manipulate their threads based on this automated analysis.

B. Thread analyzer

We use a component called thread analyzer, which is invoked periodically, to check session relationships. As shown in Figure 6, the Communication Relation Analyzer builds session relationships from the data stored by different grabbers and filters. Based on the session relationship information, the thread analyzer assigns a thread identifier (thread-id) to the sessions.

Note that assigning thread-ids to sessions is straightforward. But because ”User Activity Grabber”, ”Email Filter”, and ”Temporal Relation Analyzer” are independent to each other, there can be race conditions that affect thread assignment. For example, three users `A`, `B`, and `C`. `B` sends an email (the email is represented as `E1`) to `A`, then `A` immediately calls `B` (the call is represented as `C2`). The relationship between `E1` and `C2` is *'triggering'* relationship, represented as `triggering(E1,C2)`. During the call, `A` sends an email to `B` (represented as `E3`), and we have the relationship `containing(C2,E3)`. `E3` contains `C`'s contact information, and `B` clicks-to-call `C` (`C4`) from the email, and we have `invoking(E3,C4)`. Since *'invoking'* is handled by user activity grabber, while *'containing'* and *'triggering'* is handled by temporal relation analyzer, we may have *'invoking'* relationship before others in Comm Relation table. So we will first assign `thread-1` to `E3` and `C4`, then assign `thread-2` to `E1` and `C2`, then we have `C2` and `E3` in the same thread, at this moment, we should merge two threads.

C. Correlating Topics Across Communication Data

While thread analysis groups communication sessions at a structural level, to find relevant communication data, we compute a notion of similarity that can group items together based on their lexical content. By communication data, in the context of topic analysis, we mean content of emails, event notifications, IMs, and other communication

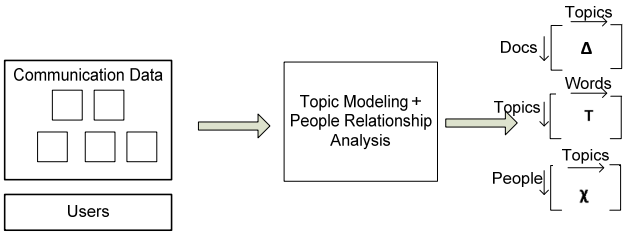


Figure 7: Topic Modeling with User Relations

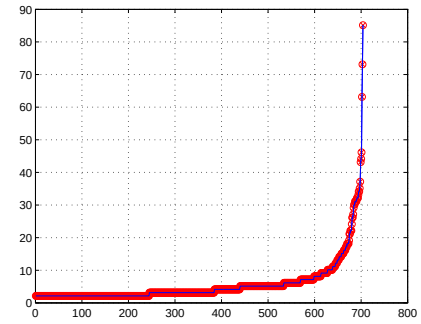
TABLE I.: Notation

Δ	Final Document-Topic Matrix
ΔL	Lexical Document-Topic Matrix
ΔR	User Relationship Document-Topic Matrix
χ	User-Topic Matrix
$\gamma(\vec{t_1}, \vec{t_2})$	cosine similarity function
τ	Topic-Word matrix based on lexical topic modeling (LDA)

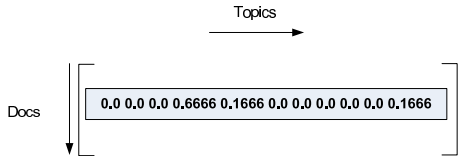
session data. To lexically group communication data we use an unsupervised learning technique based on probabilistic topic modeling [1]. However, the applicability of standard topic modeling to communication data falls short in several aspects. Topic modeling works well if the corpus is lexically rich. Communication data, unlike documents, does not always have rich data and has a lot of noise. Another aspect of communication data is prior context in the content of the data. For example, an email with “I updated the document” is lexically poor but when considered with the users it is shared with it could be grouped appropriately. This grouping is used by communication context based services such as Related Documents and Rich Communication Log. To overcome these challenges and to successfully apply topic modeling to communication data, we augment topic modeling with user communication relations [6].

Our solution is based on the idea of learning user relations and applying them to documents in communication corpus that have low lexical content. By communication corpus we mean emails, event notification, and other communication documents with data. As shown in Figure 7, given a document in the communication corpus we generate three matrices based on lexical topic analysis and user relationships. These three matrices capture the indexing of topics, keywords, and document/user topic relationships. We use a three step process for computing the three matrices. Broadly, our computation steps are:

1. First, we compute lexical topics for each item in the communication corpus. We use Latent Dirichlet Allocation [23] for topic modeling. LDA considers each document in the corpus as a bag of words and based on their lexical content generates a document distribution and a topic distribution. The document distribution



(a) Distribution for a topic i, τ_i



(b) Document topic distribution for a document j, ΔL_j.

Figure 8: Sample output from Lexical Topic Analysis with 11 topics.

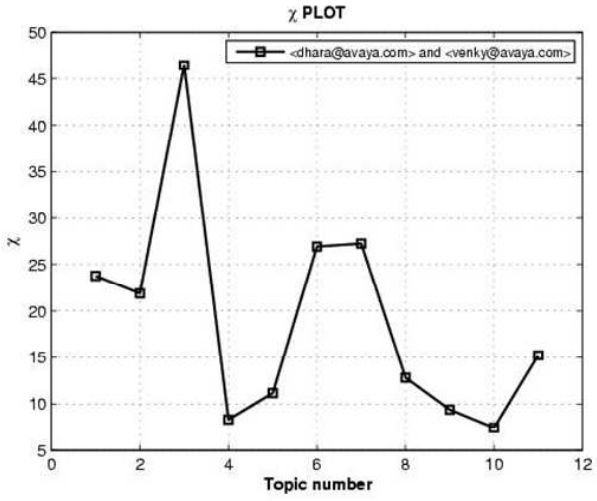


Figure 9: Sample output for χ(`dhara`, `venky`), user = `dhara`, participant in `dhara`'s conversations = `venky`

contains the topic distribution of each document in the corpus. That is for each document d in the corpus, the document distribution is a vector of tuples of the form $\{(t_1, p_1), \cdots, (t_n, p_n)\}$, where (t_i, p_i) gives the probability, p_i, that d has topic t_i and n is the number of topics in the entire corpus. The topic distribution defines the number of topics, for each topic associates a set of keywords, and for each keyword associates a weight. Each topic has a few top words with higher weights that in some sense acts as the definition of the topic. For example, Figure 8a shows a sample topic with top heavy keywords. These top few keywords define this topic. Figure 8b shows the topic distribution

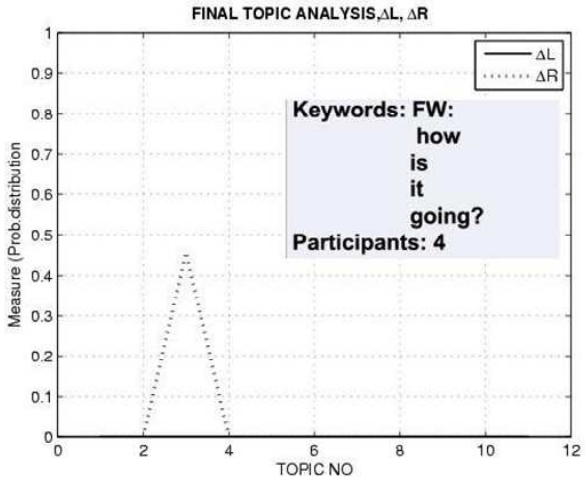


Figure 10: Document Topic Distribution for an email (keywords shown in figure) that has low lexical content. All values of ΔL are below threshold and are along the x-axis.

for one item (email in this example) in the document corpus.

2. Based on the participant list for each of the documents in the corpus and using ΔL and τ, we compute people-topic matrix, χ. With each document χ learns the topic model with respect to people-topic relationship. This helps in identifying documents that have low lexical content and to bring them closer to their topic of discussion. For each document, using χ and τ, we generate user relational document-topic matrix, ΔR. Figure 9 shows a user-participant relationship between two of the authors of this paper obtained by running topic modeling on communication data from a deployed enterprise server. Figure 10 shows topic analyzer grouping an email with low lexical content to a topic using prior communication data between the user and the people related to that document.
3. Finally, we compute Δ using the following formula

$$\Delta R = \alpha_1 \Delta L + \alpha_2 \Delta R,$$

where α_1 and α_2 are constants and can be configured based on the medium of communication data. These constants allow configuration of the system to lean towards lexical content or towards user relationships in determining the final topics for documents in a corpus.

Finding Related Documents: Given a document d from a communication corpus, we use the following computation for finding related documents.

1. Find the topic distribution (with topics and weights) row $\vec{t}$ for document d, $\vec{t} = \Delta(d)$. The row vector $\vec{t}$ represents all topics associated with d.
2. Compute the cosine similarity measure (dot product) for all the rows of Δ with respect to $\vec{t}$.
3. Ignore rows with similarity measures that are below a threshold.

4. The row ids of the remaining rows give documents that are similar with respect to topics to the original document d. Order these documents by their similarity measure and rank them.

Finding Documents Related to a Person: As shown in Figure 2 on an incoming call the notion of relation has to consider the participants in the session. Given a person (e.g., caller identity), p, the documents that are related are found by obtaining the topics $\vec{t}$ from χ(p) and using steps 2-4 from above.

As described above, topic analyzer applies lexical topic analysis techniques by augmenting communication user relationship models to correlate discrete items (mails, events, docs, etc) into similar groups. Context-based communication services can use this analysis to offer content-aware services. Some of these are described in the next section.

VI. SERVICE SPECIFIC ANALYSIS AND APPLICATION OF COMMUNICATION CONTEXT: EXAMPLES

In this section, to illustrate service specific communication context analysis, we present details of a few services we built for the unified communication experience shown in Figures 1 and 2. This unified communication experience includes rich communication log (Figure 1), predictive meeting assistant (Figure 4), and one-click conference. We describe these services in detail along with their prototype implementation and individual evaluation.

A. Predictive Meeting Assistant

The key functionality of the Predictive Meeting Assistant service is to predict the future meetings a user may schedule. The service will identify a list of potential meeting subjects and rank them based on the likelihood by which the user will schedule a meeting on the subject. For the given subject, a group of potential participants are identified and the meeting time based on their availability is proposed for the meeting.

There are two types of information sources for the meeting prediction: explicit hints and implicit hints. The explicit hints are sentences proposing or indicating future meetings. The following sentence in an email would be considered as an example of such hint: "*shall I schedule a meeting to discuss about the issue?*" Such explicit hints can be detected primarily from text communications like emails and instant messages. Implicit hints come from two sources: the user's meeting history and the aggregated view of the user's communication.

The explicit meeting hint detection is formulated as a classification problem, i.e., classifying each sentence in the user's communications as a hint or none. When an email contains an explicit hint for a future meeting, the email's subject becomes a candidate of a future meeting subject. The emails' sender or recipients become candidates of the meeting for the subject. Other potential invitees are likely the people who communicated with the user on the same topic to which the email is related to. The flow of identifying such invitees is as follows: find the email containing an explicit hint, extract the topics to which the email is related (the output of the topic analysis described in the previous section), correlate with other communications related to the topics by applying the output of the topic analysis and the thread analysis, and find the people who participated in those communications.

TABLE II.: Distribution of sentences among the authors

Author	Total	TRUE	Author	Total	TRUE
arnold-j	146	20	mclaughlin-e	73	2
bass-e	170	18	pereira-s	40	4
cuilla-m	13	0	quenet-j	6	0
delainey-d	397	103	ruscitti-k	80	7
donohoe-t	3	0	sanders-r	337	74
germany-c	483	16	stokley-c	446	35
grigsby-m	140	25	sturm-f	29	4
horton-s	106	20	tycholiz-b	12	2
jones-t	839	27	weldon-c	48	1
lay-k	179	24			

For explicit hint detection, email sentence are classified with two labels: if the sentence implies a future meeting, label it as TRUE, otherwise, FALSE. It is not sufficient to determine if the sentence contains a certain keyword like 'meeting' or 'conference'. A simple counter example is a sentence like this one, "Yesterday's meeting was quite successful." Even though this sentence contains the keyword 'meeting', it does not imply a future meeting.

1) Training Data Collection: To build the training data set for the classifier, we took the Enron email corpus [28]. We extracted sentences containing keywords like the following: "*arrange*", "*call*" , "*schedule*", "*contact*", "*discuss*", "*follow up*", and "*meeting*" . The keywords were selected based on heuristics about sentences implying a future meeting. In total, 18,706 sentences were collected. 3547 sentences were manually labeled and 382 sentences were labeled TRUE. It is about 11% of the sample. The 3547 sentences were from 19 people in the Enron data set. We tried to label a sentence TRUE whenever it sounds like the author will need to set a schedule in her calendar regardless of the size of the group. So even if it is clear that it is a one-to-one, if it makes sense for the email sender to mark it in her calendar, we labeled the sentence TRUE. A routine remark such as "*please call me if you have any question*" or a genuine question like "*do you need to meet and discuss?*" was marked FALSE.

To build a classifier for meeting prediction service, we compared three classification algorithms: Maximum Entropy [29] , Naive Baysian, and C45 Decision Tree. The Mallet package was used for the algorithm implementations [30]. Among the three, Maximum Entropy had a better performance.

2) Subject Prediction: The emails containing the sentences implying a future meeting or scheduling task provide significant information about the meeting's subject and the potential participants. A simplistic approach is

TABLE III.: User blair-l's email data summary

Event Type	Count
Outlook meeting notification Emails classified as meeting	992
scheduling by blair-l Total actual meetings	97 1089 (= 992+97)
Emails classified as meeting implication by the future meeting detector (predictions)	528 (=431+97)

TABLE IV.: Precision of subject prediction on blair-l's email data.

	Matched predictions	Prediction match ratio
Strict subject match	107 (=10+97)	20%
Modest subject match	367 (=250+97)	70%
Loose subject match	432 (=335+97)	82%

to take the email's subject as the subject of the future meeting. To measure this simplistsectionic approach and the performance of the classified trained as described in the previous section, we need to compare the subjects of the emails to the subjects of the actual meetings. Among the users in the Enron email corpus, blair-l (Lynn Blair) produced a very well organized mail records. Among her email folders are four folders containing 992 meeting appointment notifications from the Outlook Calendar (the Exchange Server) and 97 emails she used to arrange meetings. The classifier scanned every sentence of all the emails (except the Outlook Calendar notifications) of hers and identified sentences implying a future meeting. The 97 emails in the meetings folders were all classified as emails implying future meeting. 431 additional emails were classified as implying a future meeting. We compared the subjects of the 431 emails with the subjects of the Outlook Calendar meeting notifications and the 97 emails. We removed "FW:" and "RE:" from the message subjects.

Three levels of match criteria were applied.

- Strict subject match: complete string match
- Modes subject match: minimum two words (unless one of the message has a single word subject) match
- Loose subject match: starts with the same first word

Table IV shows the results of subject match. The 97 emails in the meetings folders were counted as the predictions as well as the actual meetings. The data shows that 20% of the predictions proposed the exactly same subject with one or more actual meetings. 70% of the predicted subjects have a subject with two or more common words with subject of one or more actual meetings. 82% of the predicted subjects have the same first word with the subject of one or more actual meetings.

Table V shows how many actual meeting subjects were predicted from the emails. Only 10% of the meeting subjects were identical to the subject of one or more emails (predictions). The subjects of 69% of the meetings shared two or more words with the emails. The number,

TABLE V.: Recall of subject prediction on blair-l's email data.

	Predicted meetings	Prediction ratio
Strict subject match	105 (=8 + 97)	10%
Modest subject match	751 (= 654+97)	69%
Loose subject match	378 (281+97)	35%

751, is larger than the number of predictions because there are multiple meetings on the same subjects. Only 35% of the first words of the meeting subjects were in the predicted subjects. False positive occurs when the email classified as implying a future meeting does not match any actually established meeting's subject. It is partially due to the error in classification and also due to the possibility that the user chooses a quite different expression for the subject of the meeting. A more sophisticated approach will be looking at the text in the email body in addition to the email subject to predict the meeting subject.

B. Rich Communication Log

When there are no active events, rich communication log service composes a user view that groups their communication sessions by threads and sorts the sessions based on time or relevance. When there is an active event, such as an incoming call or a scheduled appointment, rich communication log composes an user view with sessions in the log based on the relevance of the event and groups the sessions by the topic of the event. Rich communication log consumes communication context mined by thread analyzer and topic analyzer. This combined view distinguishes our solution from existing work which correlat emails or documents only based on content or metadata. Our solution correlates communication units, including emails and attached documents, not only based on the communication units' own syntax or semantic information, but also on users' communication histories, including sessions of different modalities and derived person relationships. Further, the correlation can be presented in different views for the following user activities or user context.

1. **No Activity** (default): There is no specific event happening in this context. The only factor that affects the content is the "current time".
2. **Incoming/outgoing session:** In this context, Rich Communication Log shows related sessions based on the relevance values to the content or participants of the incoming/outgoing session.
3. **Ad-hoc/meet-me conference:** In this context, conference participants and conference appointment can all be used to calculate the relevance values of related sessions.
4. **Upcoming events:** In this context, event time, participants, and topics can be used to calculate importance values.
5. **On-demand (query):** In this context, user inputs, such as keywords, selected people or events, are used to

perform context-based searching.

6. **Post-communication analysis:** Some information become available only when a session being terminated, e.g., the duration of a session, the offline tagging, and sometimes, the speech-to-text transcripts. In post-communication context, Rich Communication Log can help to retrieve related sessions and schedule future follow-up sessions.

Given one of these user activities, a view is a combination of the sorting, grouping, and weighting factors. These factors further depend on starting time of communication session, their importance and relevance that indicate how likely will users use the information, on topic of communication sessions, on people in the communication sessions, and so on. To compute such views rich communication log uses different analyzers that interact with the core thread and topic analyzers and on its service specific analyzers such as message importance analyzer, message feedback analyzer, and on user feedback handling.

1) Message Importance Analyzer: Message Importance Analyzer dynamically calculates the importance value of a user's communication sessions. The calculation consists of two parts, one that generates pre-computed importance value and the other that handles aging or decaying operation, which adjusts importance value based on temporal factors. The formula is shown below:

$$\begin{aligned} w(s) &= w(raw) + a * w(topics) \\ &\quad + b * w(threads) + c * w(peoples) \\ wt(s) &= w(s) * (1 - d * (t - ts)/T) \\ &\quad (t - ts \leq T, a, b, c, d \leq 1) \end{aligned} \tag{1}$$

In this formula, the first equation calculates pre-computed importance value of a session "s". The importance value is a composition of the importance values of related topics, threads, and people. The a, b, c, d are the weighting factors that can be customized by users or adaptively learnt by the system. T represents the aging time limit, t is the current time, and ts is the session time. In the formula, w(raw) represents the importance value of the session extracted from raw session data, e.g., the value of the "Importance" header of a session, or the duration of a call, or the length of an email. In the formula, w(peoples) represents the importance value of all the participants of session "s".

$$w(peoples) = \Sigma f(w(p), n) \tag{2}$$

where n is the number of participants. For each participant p,

$$\begin{aligned} w(p) &= w(msgsp) + w(globalp) \\ &\quad + w(hierarchy) + w(distance) \end{aligned} \tag{3}$$

In which, w(msgsp) is calculated based on the messages exchanged between the person "p" and the user "u"; w(globalp) shows how popular the person "p" is for all the people in an enterprise; w(hierarchy) shows the position of "p" in a corporate hierarchy, while w(distance) shows the distance between "p" and the user in the corporate hierarchy. In formula (1), w(topics) represents the importance value of all the related topics to this specific communication session. In formula (1), w(threads) represents the importance value of the thread in which the communication session "s" belongs. For every new communication session, the pre-computed importance value of the session is calculated when the session is terminated. The importance value of a new session will affect the importance value of its related topics, threads, or people, and in turn affect other related sessions and the new session itself. This causes a circular dependency that should be avoided. To solve the problem, we stop updating the importance values of sessions if they have already been updated by introducing the new session.

2) Message Relevance Analyzer: Message Relevance Analyzer calculates the relevance value of a message to a people, an event/session, or a keyword. This is more complicated than message importance analyzer because it has one more input parameter to consider.

Relevance to a people Usually, this value is dynamically calculated when a user requires information that is related to the sessions of a specific person. For example, for an incoming call, Rich Communication Log can show the sessions related to the caller. For a person *p* and a communication session *s*, *r(p-s)* represents the relevance value between *p* and *s*.

$$\begin{aligned} r(p - s) &= r(senders) + r(recipients) + k * w(p) \\ rt(p - s) &= r(p - s) * (1 - d * (t - ts)/T) \\ &\quad (t - ts \leq T, k, d \leq 1) \end{aligned} \tag{4}$$

r(sender): the relevance value between the sender of *s* and *p*. The simplest algorithm is r(sender) = 1 if the sender is *p*, otherwise r(sender) = 0. A more sophisticated way is to find out how many sessions *p* and the sender of *s* participated together. The more the sessions, the bigger the r(sender) is. We can use the same way to calculate the relevance value between a recipient of *s* and *p*. If a session has more than one recipient, we have:

$$r(recipients) = \Sigma f(r(recipients), n) \tag{5}$$

w(p) is the same as what defined in formula (3). We can also make formula (4) more sophisticated by considering shared topics between person *p* and session *s*, but that makes the computation too heavy to scale.

Relevance to an event Usually, this relevance value is useful when a scheduled event is happening, such as a meet-me conference. A message and an event may be associated in multiple ways.

$$\begin{aligned} r(e - s) &= a * r(e - topic - s) + b * r(e - thread - s) \\ &\quad + c * r(e - people - s) \\ rt(e - s) &= r(e - s) * (1 - d * (te - ts)/T) \\ &\quad (te - ts \leq T, a, b, c, d \leq 1) \end{aligned} \tag{6}$$

For an event *e* and a session *s*, r(e-thread-s) checks whether the event and the session are in the same thread

or not, r(e-people-s) checks whether the event and the session have the same participants, and r(e-topic-s) checks whether the event and the session share the same topic. (3) Relevance to keywords This is used for searching sessions based on keywords.

$$r(k-s) = a * r(k-topic-s) + c * r(k-people-s) + s(k-content-s)$$
$$rt(k-s) = r(k-s) * (1-d*(t-ts)/T) \quad (t-ts \leq T, a, c, d \leq 1) \tag{7}$$

r(k-topic-s) checks whether the keyword *k* is part of a topic of the session *s*, r(k-people-s) checks whether *k* is a participant of *s*, and s(k-content-s) is the regular search result by searching *k* in the content of *s*.

User feedback handling Our system handles user feedback in two ways. One is by allowing user to configure or tune various parameters for the services described above, such as rich comunication log or predictive contacts. That is, users can tune the service analyzers by indicating their weightage preferences. For example, a user that considers events or conferences more important than calls or emails, can tune the results of predictive analyzers towards events.

Another kind of feedback is by monitoring user activity. For example, when a user picks up a session in Rich Communication Log to review, this user action implies that the selected session is more important than the other unselected sessions. We should then adjust the importance value of this session. The adjustment will be applied to the underlying weighting factors (a, b, c, d in formula (1)). Considering a specific time range T, we use ST to represent all the selected sessions in T. The adjustment is to maximize the overall important value of selected sessions by updating the value of a, b, c, and d. Some of these mechanisms are currently deployed in the trial system while some parts are under investigation.

C. One-Click-Conferencing

One-click-conferencing is a convenient feature used for appointment handling. As shown in Figure 4, when a user clicks the call button, the system can automatically parse the location string, dial `17203334444`, once the call is connected, it then dials `111111`. This is different from traditional click-to-call feature, which does not distinguish and identify bridge numbers and conference pin codes. To enable this feature, we need to handle the following tasks:

- training data collection
- machine learning and categorizing numbers
- dialing and collecting feedback information

1) Training data collection: By observing appointments in Avaya and in Enron email archive [28], we noticed that the non-number words prior to each number are very important to determine what the number is used for. Some other observations are:

- people in the same company use similar patterns
- numbers at the end of long emails not for meetings
- numbers in subject line likely for meetings

Based on the above observations, we construct our training set to include the following information and represent a training entry as a comma separated string:

`<sender>, <size>, <position>, <word1>, <word2>, <workd3>, <number>, <class>`

In this training entry, `<word1>`, `<word2>`, and `<word3>` refer to the three non-number words prior to a number, and `<class>` refers to the classification of the `<number>`, which we got from the training set collection program. We call each comma separated value an attribute for learning. We choose to use the C4.5 [31] algorithm to build the decision tree based on the following three requirements:

- containing tree quality measurement mechanism
- allowing incremental learning
- supporting tree pruning

Note that training data is collected from many different users. This in fact crosses our system's privacy boundary by allowing one user to use data generated from other users. However, since the generated decision tree just represents commonly used patterns that are available in enterprise directories, and since this service will not release users' event specific data, users' privacy is still protected.

2) Validation and retraining: It is inevitable that the machine learning algorithm may sometimes misclassify wrong numbers for dialing (false positive) or cannot identify the right numbers for dialing (false negative). In this case, it is important to collect users feedback with the right information and re-train the decision tree. Even if there is no misclassification, this action should also be put into training process to enforce the rule. This enforcement can change the structure of the decision tree, for example, move one node closer to the root of the tree to minimize the entropy of the entire decision tree.

3) Performance evaluation: We measured the performance of the J48 implementation (Java based) of the C4.5 algorithm and found that the time taken for retraining increases linearly with the increase of the size of the dataset. The J48 performance was measured on a laptop with moderate configurations: Intel Core 2 Duo 2.4GHz CPU and 2GB RAM. We also measured the time for classification. It is around 0.03 s in average on the same machine. The classification time can vary largely depending on the depth of the decision tree. In our deployment, we choose to use J48 implementation simply for license issue. As shown in the figure, training 19200 samples takes about 0.386 ms by using J48, which is sufficiently quick for our usage.

To measure the accuracy of J48, a pair of datasets with instances of 321 for training, 100 for testing, and 422 for training, 51 for testing respectively were taken. The classification was done on weka with 10-fold cross

validation and the accuracy was 83% in the first case and 100% for the second case.

D. Recommendations and Observations from our Trial System

This section presents what we have learned from our implementation and deployment experiences in three categories: feature, architecture, and implementation.

1) Features: Three things were stressed in the feedbacks from the trial users and audiences of our demonstrations in conferences and tradeshows: efficiency, simplicity, and privacy.

The goal of providing context-based features is to improve users' efficiency and not to distract users with unnecessary information. Any new feature should follow the five-three rule: *the most needed information should be in the top five results, and any action should take less than three clicks*.

Efficiency also requires a simple and familiar user interface for the users to control the way the data is presented or computed. As discussed in Section II-A, we present contacts, conversations, documents, and events as the entry points for users to use our system because users are familiar with these services. We expect that further new features should also be presented in user familiar ways. For simplicity, we also try to hide server side computation from users. For example, we use a slider interface to allow users to adjust different co-efficients and re-arrange context-related results as shown in Figure 11. A user can just move the slider and see result updates.

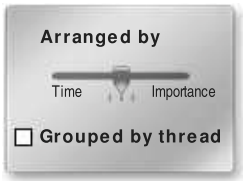


Figure 11: Slider interface

For privacy, three guidelines were proposed. First, storing raw content of the communications such as email body outside their primary storages such as the email server is discouraged. We only store computed meta data for users' conversation history in a separate storage. Second, all the context-based computations in our system are based on each user's own communication history. Communication data is not referenced across the user boundary. Third, if sharing any context-based data is desired in a collaboration session, permission must be explicitly granted by users.

2) Architecture and algorithms: We understand that the context-based services we provided in our system are just a small fraction of potential usage of communication context information. We expect that our system can be greatly extended to host more context-based features. Therefore, an extensible architecture and reusable algorithm implementation is very important. As we discussed in Section IV, our system consists of four layers. We expect many new features can use the output of our analyzers and reside in the top layer of our architecture. Some new features requiring additional data analysis may put new analyzers in the third layer. In most cases, the bottom two layers will not be affected by new features.

3) Implementation: One of the biggest concerns during our implementation was performance and scalability, considering that a regular knowledge worker may receive 50 - 100 emails and several or tens of phone calls a day. Some people may even receive several hundreds emails daily. Extracting communication context based on all these conversations in large enterprise environments is computationally quite expensive. Since audio/video communications are usually time-sensitive, users may expect to get their needed information in a session with in a couple of seconds. Therefore, new feature developers need to balance the requirements of performance and accuracy. In our system, we have periodically invoked analyzers to compute time-insensitive meta data and save those data into database. We also have real-time analyzers that directly respond to users' requests for time-sensitive context-based information. To accelerate the performance of real-time analyzers, we may pre-compute context-based information using periodically invoked analyzers and save the computation result in database. Then the real-time analyzers can retrieve the saved data and apply time-based decay functions to provide time-sensitive results. Using decay functions on pre-computed data may be less accurate than real-time computation, but it generates temporal results that improve the performance of our system.

We also noticed that different features in our context-based services may be bounded to different latency requirements. For example, when an incoming call comes, our application shows the callers name and relevant information. Caller's name is required information for call handling, so it should be bound to telecommunication's delay constraint. E.721 [32] recommends an average delay of no more than 3.0 seconds, 5.0 seconds, and 8.0 seconds, for local, toll, and international calls, respectively. On the other hand, getting relevant information of a user can be bound to web domain delay constraint. For web pages, studies [33] show that users' intention, attitude, and performance will quickly decrease as delays increased above 8 seconds.

We did some preliminary test in our trial system. We ran our context-engine on a Linux server with four Intel(R) Xeon(TM) CPU 2.40GHz and 4G memory. The test data is on real users' email and call data with about 8000 contacts for 5 users that has wide connections and heavy email and call loads. The test is on one month data of 5 users with about 8000 emails and 100+ calls (some call data are missing). Our test generates about 40,000+ thread-person relationship entries. The time of getting

relevant information in real-time is about 0.6s, and 1.5s if grouped as threads. It is an engineering challenge to handle thousands of users in such a system. Since we limit the context-based calculation upon users' own data, the computation complexity is linear to the number of users. We are working towards a testing platform of supporting 100+ users with the same performance boundaries.

VII. Conclusions

In this paper we discussed new unified communication services with communication context that enhance end user experiences. While we defined communication context as user's communication history or patterns that are mined from their data, we discussed several services that could use this notion of context in real-time. We presented challenges in building such servics and proposed a framework that captures several commonalities that are both structural and lexical. We described our core analyzers along with novel communication services with data from a deployed enterprise communicaton server that mines communication context and offers new communication services.

Acknowledgment

We thank all the enterprise communication users who participated in our trial.

References

[1] M. Steyvers and T. Griffiths, *Probabilistic Topic Models*. Lawrence Erlbaum Associates, 2007.

[2] T. L. Griffiths and M. Steyvers, "Finding scientific topics." *Proceedings of the National Academy of Sciences*, vol. 101, no. Suppl. 1, pp. 5228–5235, April 2004.

[3] G. Kossinets, J. Kleinberg, and D. Watts, "The structure of information pathways in a social communication network," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2008, pp. 435–443.

[4] D. Gruhl, D. Liben-Nowell, R. Guha, and A. Tomkins, "Information diffusion through blogspace," *SIGKDD Explor. Newsl.*, vol. 6, no. 2, pp. 43–52, 2004.

[5] A. McCallum, X. Wang, and A. Corrada-Emmanuel, "Topic and role discovery in social networks with experiments on enron and academic email," *Journal of Artificial Intelligence Research*, vol. 30, pp. 249–272, 2007.

[6] V. T. Babu, K. K. Dhara, and V. Krishnaswamy, "Augmenting topic models with user relations in context based communication services," in *Proceedings of the Third International IEEE Conference on Communication Systems and Networks (IEEE COMSNETS'11)*, Jan 2011.

[7] J. Putrevu, X. Wu, and V. Krishnaswamy, "Automate session setup based on machine learning." in *in proceedings of IEEE IMSAA 10*. IEEE, 2010.

[8] E. Shim, K. K. Dhara, X. Wu, and V. Krishnaswamy, "Communication data based user activity recommendataions." in *proceedings of ACM ICUIMC*. ACM, 2011.

[9] K. K. Dhara, V. Krishnaswamy, and T. Singh, "Reconsidering social networks for enterprise communication services," in *Globecom'10: the IEEE Global Communications Conference*, 2010.

[10] X. Wu and V. Krishnaswamy, "Widgetizing communication services," in *in proceedings of IEEE International Conference on Communications*. IEEE, May 2010.

[11] "Google priority inbox," http://mail.google.com/mail/help/priorityinbox.html. [Online]. Available: http://mail.google.com/mail/help/priorityinbox.html

[12] "Xobni," http://www.xobni.com. [Online]. Available: http://www.xobni.com

[13] B. Schilit, N. Adams, and R. Want, "Context-aware computing applicatis," in *1st International Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85 – 90.

[14] P. Brown, "The stick-e document: a framework for creating context-aware applications," in *Electronic Publishing*, 1996, pp. 259–272.

[15] A. K. Dey and G. D. Abowed, "Towards a better understanding of context and context-awareness," in *Conference on Human Factors in Computing Systems*, April 2000.

[16] A. K. Dey, "Context-aware computing: The cyberdesk project," in *AAAI Symposium on Intelligent Environments, Technical Report SS-98-02*, 1998, pp. 51–54.

[17] A. K. Dey, M. Futakawa, D. Salber, and G. D. Abowed, "The conference assistant: Combining context-awareness with wearable computing," in *the 3rd International Symposium on Wearable Computers*, 1999.

[18] "Cyberdesk: Framework for providing self-integrating context-aware services," *Knowledge Based Systems*, vol. 11, no. 1, pp. 3–13, September 1998.

[19] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a virtual community of use," in *Conference on Human Factors in Computing Systems*, 1995.

[20] P. Resnickm, N. Iakoyou, M. Susha, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Computer Supported Cooperative Work Conference*, 1994.

[21] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating 'word of mouth'," in *Conference on Human Factors in Computing systems*, 1995.

[22] M. Balabanovic and Y. Shoham, "Fab: Content-based collaborative recommendation," *Comm. ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[23] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation." *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[24] D. Ramage, E. Rosen, J. Chuang, C. D. Manning, and D. A. McFarland, "Topic modeling for the social sciences," in *NIPS 2009 Workshop on Applications for Topic Models: Text and Beyond*, Whistler, Canada, December 2009. [Online]. Available: pubs/tmt-nips09.pdf

[25] J. Chang and D. Blei, "Relational topic models for document networks," in *AAISTATS '09: Proceedings of the Conference on AI and Statistics*, 2009.

[26] X. Wei and W. B. Croft, "Lda-based document models for ad-hoc retrieval," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2006, pp. 178–185.

[27] Y. Liu, A. Niculescu-Mizil, and W. Gryc, "Topic-link lda: joint models of topic and author community," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 665–672.

[28] "Enron email dataset," http://www.cs.cmu.edu/ enron. [Online]. Available: http://www.cs.cmu.edu/ enron

[29] E. T. Jaynes, "Information theory and statistical mechanics," *Physical Review Series II*, vol. 106, no. 4, pp. 620–630, 1953.

[30] A. K. McCallum, "Mallet: A machine learning for language toolkit." *http://mallet.cs.umass.edu*, 2002.
[31] J. R. Quinlan, *C4.5: Programs for machine learning*. Morgan Kaufman, 1993.
[32] T. S. S. of ITU, "International Telecommunication Union Network grade of service parameters and target values for circuit-switched services in the evolving isdn, recommendation E.721," RFC 4730 (Proposed Standard), Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1999.
[33] D. F. Galletta, R. Henry, S. Mccoy, and P. Polak, "Web site delays: How tolerant are users?" *Journal of AIS*, 2003.