World Wide Web (2009) 12:125–147
DOI 10.1007/s11280-008-0049-x

An Operable Email Based Intelligent Personal Assistant

Wenbin Li · Ning Zhong · Yiyu Yao · Jiming Liu

Received: 31 March 2008 / Revised: 1 July 2008 /
Accepted: 7 July 2008 / Published online: 23 September 2008

Abstract The recent phenomena of email-function-overloading and email-centricness in daily life and business have created new problems to users. There is a practical need for developing a software assistant to facilitate the management of personal and organizational emails, and to enable users to complete their email-centric jobs or tasks smoothly. This paper presents the status, goals, and key technical elements of an Email-Centric Intelligent Personal Assistant, called ECIPA. ECIPA provides various assisting functions, including automated and cost-sensitive spam filtering based on corresponding analysis, ontology-mediated email classification, query and archiving. ECIPA can learn from dynamic user behaviors to effectively sort and automatically respond email. Techniques developed in Web Intelligence (WI) are adopted to implement ECIPA. In order to facilitate cooperation of ECIPAs of different users, the concept of *operable email*, an extension of traditional

W. Li
Shijiazhuang University of Economics, Shijiazhuang 050031, China

W. Li · N. Zhong (✉) · Y. Yao · J. Liu
The International WIC Institute, Beijing University of Technology, Beijing 100022, China
e-mail: zhong.ning.wici@gmail.com

N. Zhong
Department of Life Science and Informatics, Maebashi Institute of Technology,
460-1 Kamisadori-Cho, Maebashi-City 371-0816, Japan

Y. Yao
University of Regina, Regina, Canada

J. Liu
Hong Kong Baptist University, Hong Kong, China

email with an *operable* form, is introduced. ECIPA can in fact be viewed as a family of collaborative agents working together on the operable email.

Keywords operable email · personal assistant · email filtering · intelligent agents · Web intelligence

1 Introduction

Email is one of the most useful communication tools over the Internet. Sine 1971, the functionality of email has been in a constant evolution, from simple message exchanges to multimedia/hypermedia contents communication to push technologies for direct marketing. For instance, email can work with a Web browser, such as Microsoft Internet Explorer and Netscape, as a useful function of Web-based e-commerce that is popularly used by enterprises to promote products and spread information. In this paper, we refer to this kind of expansion in functionality as *email-function-overloading*. People reply on email to conduct their jobs and tasks in daily life or business. They use emails to plan and coordinate their own work, to track and dispatch tasks, to exchange information and cooperate among group members, and to publish results. In other words, their work is mostly *email-centric*.

Email-function-overloading and email-centricness, as pointed out by Tinapple and Woods, have presented inconveniences and problems to email users [28]. We highlight some of the most serious ones as follows. First, email users often have cluttered inboxes containing hundreds of messages, including outstanding tasks, partially read documents and conversational threads. Attempts by users to rationalize their inboxes by archiving may not be achieved in a straightforward fashion. Important messages usually get overlooked, or lost in archives. Second, as a communication tool, many documents are exchanged as email attachments. Users need to sort out not only emails and resources, but also relationships between them. The existing email facilities are not designed to handle such management tasks. Third, email-mediated communications can in general be categorized into two approaches: automated and non-automated. In existing email facilities, most of the communication tasks are not fully automated. It is necessary for users to deal with those tasks manually. Fourth, receiving, recognizing, or deleting spam messages presents a time-consuming, manual task.

The above mentioned drawbacks in the existing email systems can cause many inconveniences and sometimes serious consequences. For instance, users may frequently forget an important appointment. They may be tired of those repeated tasks which could be automated. They may not be able to remember where an attachment has been stored. Hence, there is a real need for developing an intelligent assistant to improve the management of personal and organizational email, which enables users to complete email-centric tasks smoothly. In fact, developing intelligent applications on various kinds of network is one of the goals of Web Intelligence (WI) [35–37].

This paper presents the status, goals, and key technical elements of an Email-Centric Intelligent Personal Assistant, called ECIPA. ECIPA adopts ontology to store contents from several personalized information sources, local and global information, such as emails, local files, and user's background knowledge, to provide a single gateway to index, manage and search such information. In addition to those tasks which are available in email client applications today, ECIPA provides automated and novel functions. For example, ECIPA can automatically learn the dynamic user behavior by monitoring how the user processes ingoing and outgoing emails. ECIPA can also find out whether or not its owner is on vacation. It can automatically respond senders during the vacation period. In order to reduce the burden of processing the large volume of spam messages, ECIPA incorporates a filtering agent to block spam. We have particularly designed an active email system, called *operable email*, for automated communication between two ECIPAs. The operable email provides an interface for a user or the user's ECIPA to interact with other users' ECIPAs, in order to perform some automated work as permitted by those users.

In the rest of this paper, we first present some background and related work on ECIPA in Section 2. In Section 3, we give an overview and the architecture of ECIPA. Section 4 describes what is the operable email and its main features for automated email-mediated tasks in ECIPA. The main agents in ECIPA, such as extracting agent (EtA), learning agent (LA), filtering agent (FA) are discussed in Section 5. Some implemented snapshot and experimental results of ECIPA are provided in Section 6. Finally, we present concluding remarks and future work in Section 7.

2 Background and related work

The flood of wanted or unwanted messages has aroused a great need for developing intelligent personal assistants that are aimed to automate the classification [4, 10, 15], archiving, filtering [6, 11, 26, 34], storage, representation, and retrieval of emails. An assistant collaborates with its user(s) in the same work environment. It can assist a user in a range of areas [14]: (1) performing tasks on behalf of the user; (2) training or teaching the user; (3) helping users to collaborate; (4) monitoring events and procedures.

In this work we show a more flexible and robust framework of an assistant. This assistant combines various WI-related techniques, such as the operable email, agent-based etc. to produce a highly personal and automated system, with more features than found in a typical system today. The outstanding characteristics of such an assistant are that firstly it adopts the operable email as a basic for automated functions; secondly, it uses agent-based architecture. Thus, in the remainder of this section, we discuss related work with respect to the two aspects.

2.1 The assistants based on variation of email

Although email-mediated tasks can be either automated or non-automated, the email client cannot work for its users automatically without semantic support. To open the door for implementing a wide range of email-mediated applications with automated functions of response by machine, researchers are turning to modify or extend the format of traditional email. Semantic email is a good example [17].

A semantic email process (SEP) should be first represented for a semantic email task. A SEP contains three primary components: originator, manager and participants. A SEP is initiated by the originator, who is typically a person. The manager may be a shared server or a program run directly by the originator. A new SEP invoked by an originator is sent to the manager. Then, the manager sends email messages to the participants. After that, the manager handles responses, and requests changes as necessary to meet the originator's goals. The participants, who are also typically a person, respond to messages received about the process. Surely, the semantic email has the greatest promise for implementing automated functions. Some applications based on such a kind of email have been developed [9]. However, such semantic emails have the following disadvantages and hence cannot be used in ECIPA. First, an automated task needs to be defined as a semantic email process (SEP) by a trained user. Second, it needs an additional server (namely manager) to support the functions of a SEP. Third, many bread-and-butter tasks cannot be defined as a SEP. In order to solve the problems, we describe how to design and apply the operable email for implementing automated functions in ECIPA.

2.2 The assistants based on agent technology

Among many applications, intelligent agent technology has emerged as a development paradigm in which programs are created as proxies for human users in interactions with other humans or computers. For instance, agent-based approaches have been developed to assist users in their own work. Moreale and Watt [20] explored the application of an agent assistant metaphor to mailing list knowledge management. Mitchell et al. [19] pointed that a long-standing goal of artificial intelligence (AI) is the development of intelligent workstation-based personal agents to assist users in their daily lives. By analyzing emails, contact person names, and online calendar meetings, they developed a clustering approach to automatically acquiring a knowledge base describing a user's different activities, e.g., contact people, meetings, and email handling. They studied the distribution of email words, the social network of email senders and recipients, and the results of Google Desktop. Huang et al. [8] discovered some major ongoing activities based on a variety of data available from users' workstations, including emails, files, online calendar, and the histories of Web page accesses. Deng et al. [5] implemented a personal email agent, named P@rty, which can help its user to manage the growing volume of messages. P@rty was designed to classify incoming messages with respect to a user into folders automatically, and prioritize each incoming message based on the user preferences. Bergman et al. [3] studied the capabilities of smart vacation responding, junk mail filtering, efficient email indexing and searching, deleting, forwarding, refiling, and prioritizing of emails. The key contribution of their work is to leverage high-quality open source components for information retrieval, machine learning, agents and rules to provide a powerful, flexible and robust capability. Xia and Liu [29] implemented an adaptive personal email agent (PEA) that can learn the mails handling preferences of its users and automatically categorize and manage the emails. Ho et al. [7] described an email system, called EMMA (email management assistant), that can sort messages into virtual folders, prioritize, read, reply (automatically or semi-automatically), archive and delete certain mail items. Pires and Abreu [22] applied the framework of an evolving logic programs, named EVOLP, to modeling the reactive and updateable rule bases and employing them to define an evolving email Personal Assistant Agent.

Some of the existing projects on intelligent agents have been criticized for solving only toy problems [3]. Many of the ideas described in the earlier studies are fragmentary. In addition, some assistants were implemented to solve simple out-of-date problems with out-of-date technologies. As the real-world problems become more complex and new technologies continue to emerge, it is imperative that we leverage on and incorporate emerging theories and techniques, such as agent-based computing, heterogeneous information source handling, information retrieval, machine learning, as well as WI-related techniques.

3 The architecture of ECIPA

Web intelligence aims at producing new theories and technologies that will enable us to optimally utilize the global connectivity, as offered by the Web infrastructure, in life, work, and play [13]. ECIPA is an example of the applications in Web intelligence. The main objective of an ECIPA is to design customizable and extensible agents that work together to process ingoing, outgoing, and existing emails.

Figure 1 shows part of use case diagram of ECIPA. It indicates that ECIPA can deal with two kinds of emails, i.e., operable email and traditional email. The former is designed for automated tasks completed by agents in ECIPA, which will be described in the next section. When a new email is detected by MSA (monitoring/sending agent), it firstly identifies the type of the email. Then, MSA informs corresponding agents to deal with this email, by writing a piece of message on the blackboard (all the agents in ECIPA communicate with each other via a blackboard). The blackboard

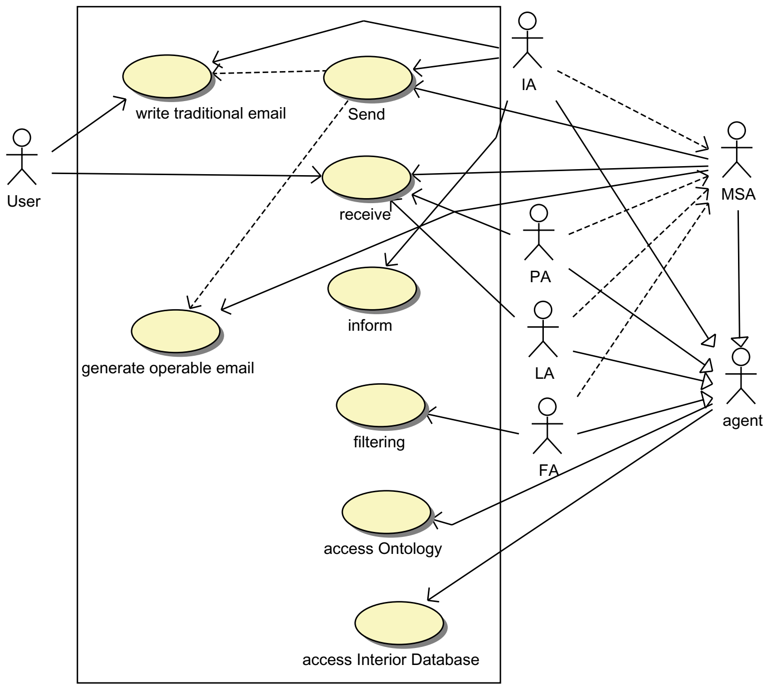


Fig. 1 Part of the use case diagram of ECIPA. *FA* filtering agent, *PA* parsing agent, *LA* learning agent, *MSA* monitoring/sending agent.

is a block of memory that can be accessed by each agent. In fact, it is designed as a message queue for storing blackboard items. All the items are categorized into three types to store, namely, the one from MSA to an agent, the one from an agent to MSA, and the one from an agent to another agent. Main entries in each item include: MESSAGE_ID (the message ID), SOURCE_ID (the sending agent ID), DESTINATION_ID (denotes the receiving agent ID), CONTENT (the message content), USING_OR_NOT (the message is using or not). Most of functions of ECIPA are supported by the agents, such as PA (parsing agent), LA (learning agent), FA (filtering agent) etc. Other agents in ECIPA are: extracting agent (EtA) which distills information from diverse sources; querying/altering agent (QAA) which answers or alerts the user; configuring agent (CA) which sets the running parameters; interface agent (IA) which provides user services; external agent (ExA) that communicates with the archiving system.

Each agent has the following statuses: waiting, running, blocked, suspended. The waiting status denotes that the agent is in the queue waiting for executing. The running means that the agent is running. The blocked denotes that the agent is blocked by MSA. The suspended is that the agent just registers in MSA, it is still stored in the disk. The exchanged diagram of agent status is shown in Figure 2. All agents register their own status in MSA.

The interior database is designed to store various information, e.g., the name of each agent, the capabilities of each agent, the status of each agent, the contact book. The ontology stores built-in concepts and their relationship, and archives emails based those concepts.

The main functions of ECIPA supported by our contributions can be summarized below:

- Operable email-based intelligent cooperation (supported by PA and EtA): In all the email-mediated works with respect to a user, part of them can be carried out automatically [17]. The use of the operable email allows an ECIPA to aid its user to automatically carry out the formalized tasks enclosed in the operable email sent by another user or the user's ECIPA.
- Ontology-mediated email management (supported by EtA, ExA, QAA): An ontology is designed to store background knowledge with respect to the user, the information of local or global resources, emails and attachments within their

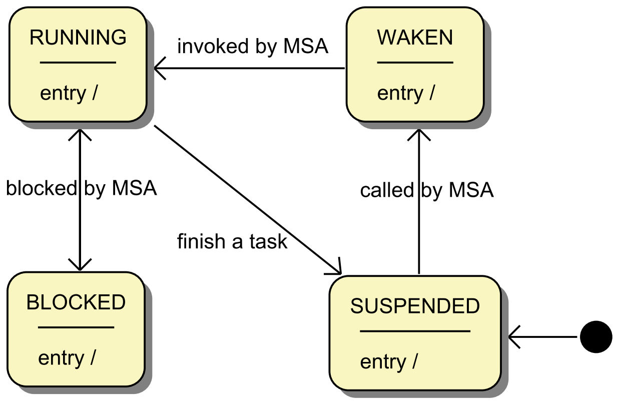


Fig. 2 The statuses of agents in ECIPA.

context. ECIPA provides the flexible functions for classifying emails into virtual folders based on the operations on concepts in the ontology, as well as those for queries based on concepts.

- Dynamic user behavior learning-based sorting/responding (supported by LA): ECIPA learns dynamic user behaviors based on the time window techniques developed by us (see Section 5.2). It prioritizes incoming messages according to the habit processing emails with respect to a user. In addition, by finding and analyzing the user behaviors, ECIPA can identify whether the user is on vacation. Thus, ECIPA responds for the user automatically in the case of vacation.
- Automated and cost-sensitive spam filtering (supported by FA): ECIPA combines multiple Naive Bayesian filters to block unwanted emails. The filtering method is sensitive to the cost of the false positive errors and the false negative errors.

ECIPA demonstrates that useful tasks can be accomplished by means of agent interaction. Furthermore, The key techniques adopted by ECIPA are very useful for developing other intelligent applications on the Web. For example, the dynamic behavior learning approach based on time-window can be adopted in other personalizing recommendation system on the Web. Although the architecture described is developed as a three tiers structure, it can be easily implemented by other ways.

4 Operable email as a basis of ECIPA

4.1 Introduction to the operable email

As an indicator of collaboration and knowledge exchange, email provides a rich source for extracting informal social network across the organization. As a result, it is a highly relevant area for research on social networks. Just as the WWW consists of websites and hyperlinks which *explicitly connect* sites, the Whole World Social Email Network (WWSEN) consists of email addresses and communication relationships which *implicatively connect* users. Hence, among those research topics of WI, recasting email for ubiquitous social computing is a special goal. On the other hand, enhancing email with semantic is an important way for implementing email based intelligent applications. The operable email is our solution for such a goal.

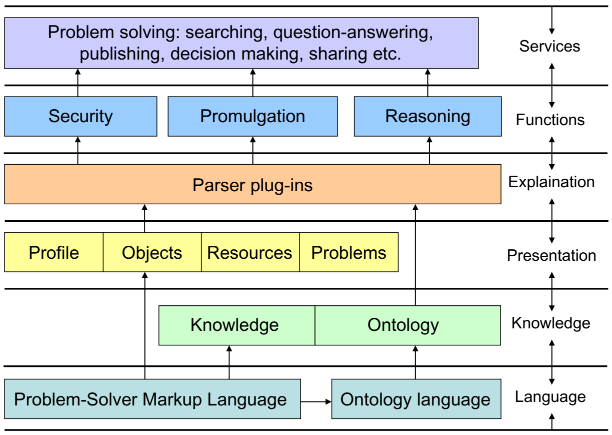
Different from the traditional, merely human-readable email, the *operable email* instead provides a language as well as an infrastructure where agents on the WWSEN can automatically cooperate and deal with some tasks by expressing some information in an email with a machine-processable form. We are not sure if a formal definition of the operable email is useful or desirable. Nevertheless, we suggest the possible best way of summing up the operable email as follows. It is a technology for enabling machines to make more sense of some specifically emails sent from an agent to another, with the result of implementing automated functions or services on the WWSEN.

It is well know that the traditional email was originally designed as a tool for the communication of *one to one* user or *one to multiple* users. For simplification, we only consider *one to one* communication. In our view, the communication between two users can be categorized into two types, namely automated and non-automated. However, current emails without semantic do not support any automated application, although many new functions have been added to email. Instead of providing a simple tool for manual communication between two users, the operable email focuses on offering a mechanism for those two kinds of communications. Consequently, with the support of the operable email, many intelligent functions, such as searching, question-answering, and problem-solving, become possible on the WWSEN. Sustained by automated communication based on the operable email between any two nodes, the WWSEN provides another Internet application environment that enables people to effectively publish, share, and search information or resources. Concretely, we view each node as an autonomy agent entity, this allows more functionality of a node, as well as additional connections and interactions between different nodes. The operable email is viewed as a soft communication media between any two entities. That is, for automated tasks, the semantic information from agent x to agent y is enclosed into an operable email. Imaging that user A wants to search a resource on the WWSEN, A first inputs relative information about that resource into A's agent x. After that, x automatically generates an operable email o_e enclosing information that can be automatically understand by other agents. With an appropriate mechanism, o_e is promulgated on the WWSEN. Then, the received agents parse, understand and respond o_e. Although many details in the illustrative example are ignored, we still see that the WWSEN provides asynchronous services (while WWW provides synchronous ones) to support intelligent problem solving, decision making, question-answering, cooperative teamwork and so on.

In fact, the WWSEN absorbs the Web's ideals, and it is a complementarity for current Internet computing. Thus, the current and future WI-related theories and techniques serving for the wisdom Web provide possible implementation platforms for developing the operable email and the WWSEN [13, 36]. Figure 3 shows the stack of operable email, in which the operable email consists of six layers, namely *language layer*, *knowledge layer*, *representation layer*, *explaination layer*, *functions layer* and *services layer*.

- The *language layer* aims at designing Problem Solver Markup Language (PSML) [27] for representing globally distributed resources and knowledge from

Fig. 3 The stack of the operable email.



the WWSEN, as well as problems, tasks, queries, results, profiles and objects. Note that PSML is a language which bears a machine-understandable form.

- In the *knowledge layer*, various knowledge, and ontologies including concepts and their relationship are represented and stored.
- Using PSML, the *representation layer* provides a mechanism with which profiles, objects, resources etc. are expressed. A profile is used to denote personalized information related to a user. An object refers to a mobile agent defined by PSML. A sharing file, a piece of publishing information etc. on the WWSEN are called by a joint name, namely resources.
- The *explaination layer* provides multifarious plug-ins for parsing messages enclosed into an operable email. For example, a query initiated by a sender is enclosed into an operable email which will be sent to receivers. At the receivers' side, the corresponding parser plug-in in the agents is invoked to "understand" the intention of that query. After that, an answer is enclosed into another operable email, and then is sent to the sender. Different plug-ins parse different messages of the operable email. In this way, the operable email affords a flexible and extensible architecture. That is, when we want to the agent to understand a new kind of message of the operable email, only installing (or developing) a corresponding plug-in is needed.
- Just as its name implies, the *functions layer* offers some functions such as the function of security mechanism, the promulgation function and the reasoning function. Agents, the sharing resources, etc. on the WWSEN are especially vulnerable to new threats from a security viewpoint. The security mechanism aims at studying countermeasure for that. Through the promulgation function, an operable email can penetrate over the WWSEN when it is necessary. Carrying out the reasoning function using knowledge is required for intelligent services.
- The *services layer* includes many user-oriented services which aid users to solve sundry problems, such as querying, sharing/searching, publishing etc.

In our view, the operable email opens the door for implementing a wide range of email-mediated applications with automated functions on the WWSEN that are neglected or infeasible before. By providing a more efficient and effective communication means with semantic, the operable email exploits research towards a long-standing goal of WI, i.e., the wisdom Web [13, 36].

4.2 The operable email in ECIPA

From the description in Section 4.1, we can see that designing such operable email is a complex engineering. To demonstrate some automated functions for ECIPA's users, we here provide a simple PSML in language layer of the operable email. Using a special field in the header of an operable email, the MSA in ECIPA can distinguish an operable email from a traditional email. The content in an operable email is generated by ECIPA (or written by users) according to the PSML syntax as shown in Table 1.

According to the PSML syntax for providing the automated tasks in ECIPA, we define some commands as shown in Table 2. Due to the limited space of this paper, the parameters of those commands are omitted. To simplify the parsing process, we represent each script of the commands as an XML document according to a

Table 1 The command syntax of the operable email in BNF.

⟨message⟩:: = ⟨command⟩(⟨blank⟩⟨para-value⟩)*
⟨command⟩:: = ⟨identifier⟩
⟨para-value⟩:: = ⟨paraName⟩⟨blank⟩⟨value⟩
⟨paraName⟩:: = ⟨identifier⟩
⟨value⟩:: = (⟨ascii⟩)*
⟨identifier⟩:: = ⟨alphabetic⟩(⟨alphabetic⟩|⟨numeric⟩)*
⟨blank⟩:: = ⟨whitespace⟩(⟨TAB⟩|⟨ENTER⟩)$^+$

"*" indicated any number of occurrences.

"+" means that the number of occurrences should be greater than 1.

scheme of the body of the operable email before sending [12]. In other words, ECIPA encodes message-as-XML documents into operable emails, and decodes the message-as-XML documents back into messages that represent the commands as shown in Table 2.

Hence, in ECIPA, we can see that the operable email provides an email-based communication means, facilitated by an assistant of the users. This means that the operable email supports an email-based agent infrastructure where agents can automatically deal with some tasks that are impracticable currently. There are several reasons why using the operable email as a communication media for "bear the weight" of the communication script. First of all, email clients are lightweight and available on most of computational devices. In the next place, emails are peer-to-peer and symmetric communication protocols. Hence, an email-based agent communication channel does not need a router component. Again, firewalls are not an issue for email-based communication. Agents on either side of a firewall can communicate more easily with an email-based infrastructure than with a TCP/IP infrastructure. Finally, the ISP mailbox acts as a message queuing facility, obviating the need for a specialized message queuing component. Consequently, the operable email or its transmutation to be developed in the future is an imperative, simple-seeming but very useful tool for implementing WI applications on the next generation of the Web (i.e., the Semantic Web).

In practice, operable emails can be generated by either a user or a program. It is obvious that the user-generated case is tedious, time-consuming, and error-prone. Thus, in ECIPA, we adopt the latter to form operable emails for the user according to the input in the IA of the assistant.

Table 2 The main commands as used in ECIPA.

Command	Meaning
download	Download a file from another assistant
listfile	Ask the receiver to list the sharing files the sender can access
sendfile	Ask the receiver to send a file
appointment	Make an appointment with the receiver at a given time
meeting	Hold a meeting with the receivers at a given time
bulletin	Publish a bulletin to receivers
subscribe	Subscribe a piece of information from the receiver
ask	Ask the receiver about something

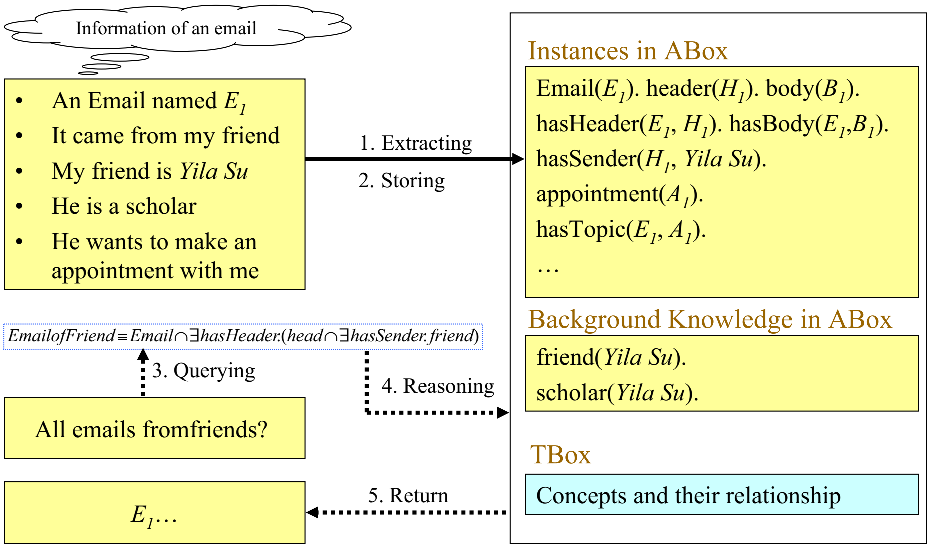
5 An agent based design of ECIPA

In this section, we describe in detail the agent based architecture of ECIPA. More specifically, the functions and working principles of some of the agents are examined in relation to the concept of operable emails.

5.1 Ontology-based processing of operable emails

Ontologies provide many useful capabilities [21], such as (1) sharing domain information among people and software; (2) enabling reuse of domain knowledge; (3) analyzing domain knowledge and making it more explicit; (4) separating domain knowledge from its implementation. These useful capabilities motivate us to adopt the ontology-based method for email management. Specifically, our objective is to archive email information, email context, user's background, and other resources into an ontology represented by OWL (Web Ontology Language: http://www.w3.org/2004/OWL/) whose logical footstone is Description Logic (DL) [2]. In this way, ECIPA provides the retrieving, classifying and managing functions based on concepts.

In DL, the complex concepts are constructed by using the atomic concepts, relationship, and constructors. According to the ontology, the EtA in ECIPA extracts the relevant information of each email, and then stores them as the instances into the ontology's ABox. as shown in Figure 4. Hence, an email is decomposed into much piece of information, i.e., the instances in the ABox of the DL-based ontology. After the atomic and complex concepts are defined to obtain the instances of the atomic concepts and their relationship, the instances of a concept can be queried through



Information of an email
An Email named E_1
It came from my friend
My friend is *Yila Su*
He is a scholar
He wants to make an appointment with me
1. Extracting
2. Storing
Instances in ABox
Email(E_1). header(H_1). body(B_1).
hasHeader(E_1, H_1). hasBody(E_1,B_1).
hasSender(H_1, *Yila Su*).
appointment(A_1).
hasTopic(E_1, A_1).
…
$EmailofFriend \equiv Email \sqcap \exists hasHeader.(head \sqcap \exists hasSender.friend)$
3. Querying
4. Reasoning
Background Knowledge in ABox
friend(*Yila Su*).
scholar(*Yila Su*).
All emails fromfriends?
TBox
Concepts and their relationship
5. Return
E_1…

Fig. 4 The concept-based query process in ECIPA. *1, 2: ExA extracts and stores an instance into the ontology; 3: QAA generates a formal query according to user's query given in IA, and send it to an inference engine; 4: reasoning; 5: the results are shown in IA.*

the reasoning in an inference engine of DL. For example, “sender” and “Email” are two atomic concepts in the ontology, “hasSender” is an atomic relationship between these two concepts. “friend” is a sub-subclass of “sender”. Figure 4 shows that the ABox of this ontology has the following instances: *Email(E_1); hasSender(E_1, Yila Su); friend(Yila Su)*. Then we can use the following concept to search “emails from friends”:

$$EmailofFriend \equiv Email \cap \exists hasHeader.(head \cap \exists hasSender.friend).$$

In this example, the inference engine returns the instance “E1” of the above concept, and shows it to the user via IA. The searching method based on concepts can return the results that the user really needs. However, the precision and recall of the traditional searching based on keywords are not perfect.

The main purpose of a concept-based query is to provide the easy, accuracy, and fast access to all information related to email messages (i.e. message contents, header fields, attachments etc.). However, the correspondence between a user’s intension of queries and concepts needs to be established to support such a function, as shown in Figure 4. To solve this problem, we set several built-in concepts maintained by the QAA in ECIPA. For example, the following concepts can be defined to distinguish sender’s status: $family$, $colleague$, $friend$, $businessman$, and $scholar$, whose instances are set by the user.

$$\begin{aligned}
&EmailofFamily \equiv Email \cap \exists hasHeader.(head \cap \exists hasSender.family)\\
&EmailofConfrere \equiv Email \cap \exists hasHeader.(head \cap \exists hasSender.confrere)\\
&EmailofFriend \equiv Email \cap \exists hasHeader.(head \cap \exists hasSender.friend)\\
&EmailofBusiness \equiv Email \cap \exists hasHeader.(head \cap \exists hasSender.merchant)\\
&EmailofScholar \equiv Email \cap \exists hasHeader.(head \cap \exists hasSender.scholar).
\end{aligned}$$

Thus, an input of a query accepted by the IA is automatically mapped by QAA into a built-in concept or that one defined by multiple concepts. For example, if a user wants to get those emails sent from friends who are scholars, ECIPA returns the instances synchronously defined by the two concepts, namely *EmailofFriend* and *EmailofScholar*, as follows.

$$\begin{aligned}
&EmailFromScholarAndFriend \equiv Email \cap \exists hadHeader.(head \cap \exists hasSender\\
&.scholar) \cap (head \cap \exists hasSender.friend)).
\end{aligned}$$

By means of executing a query, messages can be classified into a virtual folder. A virtual folder contains the set of messages defined by an atomic or complex concept at any given moment. An analogy can be established between a virtual folder and a view in databases, which is a virtual relation defined by a query over existing, concrete relations. This enables a user to define virtual folders based on the combination of built-in concepts. The expression for a folder is executed every time as a user wishes to retrieve the set of messages represented by the folder criterion. Such a method guarantees the consistency between the classification criteria specified for a folder, and the set of messages associated to it. At the same time, the logical association of emails to virtual folders has some additional advantages; for instance, the same email is related to one or multiple folders without message duplication.

As described above, an ontology is also designed to archive the user's background knowledge, the information about emails and attachments within their context. In the archiving method, ECIPA indexes all the contents without user's involvement. The ExA in ECIPA acts as the external interface that interacts with the archiving agent of the organization. The enterprise agent can tell the ExA what are the required emails for archiving. For those required messages, the ExA protects them from tampering.

5.2 Learning and adaptation

ECIPA is a personal assistant. A prerequisite for developing systems providing the personalized services is to understand user behavior [1]. However, it is very difficult to learn the behaviors of a user because they are dynamic. That is, the behavior in a period is different from that one in another period. In order to capture the dynamic behaviors, we adopt a technique, namely *time window* that refers to a period in which LA observes behaviors of its users. Figure 5 shows an illustrative example of the time window. In each window, LA observers behaviors of its users, such as determining what kinds of emails the user read with a higher priority, the deleting behavior, and the forwarding or replying behavior. Hence, when the window is sliding step by step, the learned behaviors change synchronously.

Supported by LA, ECIPA provides the functions with respect to behavior learning: (1) vacation responding, and (2) prioritizing incoming messages. It can be guessed that a user is on vacation if ECIPA observed no outgoing emails. However, some receivers may be waiting for a user's reply. In this case, based on the analyzing results of LA, the MSA in ECIPA automatically sends a reply to tell the senders that its user may be on vacation, and to prompt them try to contact the user in other ways if they have urgent things. The length of a sliding step (denoted by ℓ below) and the size of a time window (denoted by s below) are the two crucial parameters for this function, where $s > \ell$. For the service of vacation responder, we suggest that the user sets ℓ to 1, and sets s to $\lfloor m/n \rfloor$, where m is the maximum length of vacations, and $2 <= n < m$. For example, the first author often has a short vacation less than 7 days, then the author sets s to 3, where m=7, n=2.

The architecture of ECIPA shows that all agents in ECIPA run on the server side. Hence, even the user is not online, ECIPA still provides some of the automated functions (e.g., the vacation responding) for its user.

As mentioned above, ECIPA also learns user behavior to prioritize incoming messages. The user may then view emails sorted by priority. That is, the priority

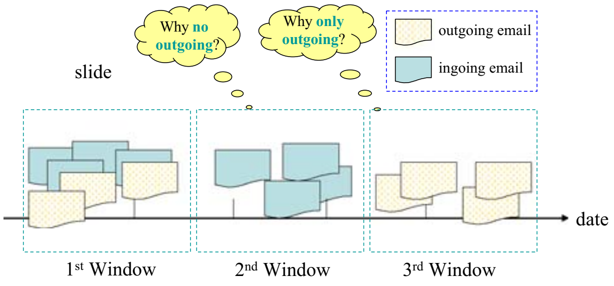


Fig. 5 An illustrative example of the time windows.

is used to represent the degree of importance of new emails. In ECIPA, we define 4 types of priorities for each message:

- Read-based Priority (*RP*) that is based on the principle "more early read, more important";
- Sender-based Priority (*SP*) that is based on the principle "more frequency, more important";
- Similarity-Based Priority (*SIP*) that is based on the principle "more similar to important emails, more important";
- Combined Priority (*CP*) that is a combination of the above three priorities.

Suppose that there are n new emails, e_1 is the first email that a user has read, e_2 is the second one, ..., e_n is the last one. Then, the *RP* of the ith email is defined as $RP(e_i) = (N - i + 1)/(N * (N - 1)/2)$. Furthermore, suppose that, in the *full time window*, there are N emails in total, and among them, there are N_1 ingoing emails, N_0 outgoing emails, N_{j1} emails are sent by the jth sender, N_{j0} are sent to the jth sender. In the *recent time window*, the jth sender has sent M_{j1} emails, and the ECIPA user has sent M_{j0} emails to this sender. Then, *SP* of the jth sender is defined as

$$SP(j) = \sum_{m=0}^{1} \left(\frac{N_{jm}}{N_j \sum_{i=1}^{N_s} (N_{im}/N_i)} + \frac{M_{jm}}{M_j \sum_{i=1}^{N_t} M_{im}/M_i} \right) + N_j/N \tag{1}$$

where N_s is the count of all the senders, N_t is the count of all the time windows, $N_j{=}N_{j1}{+}N_{j0}$, $M_j{=}M_{j1}{+}M_{j0}$.

SIP of the jth new email e_j is defined as follows. Assume that $e_1, \ldots, e_K$ are e_j's K nearest neighbors computed by k-NN method [18]. Then, *SIP* is

$$SIP(e_j) = \sum_{i=1}^{K} RP(e_i)/K. \tag{2}$$

ECIPA sorts new emails according to the *CP* value of each email. For a new email e, $CP(e)$ is computed as

$$CP(e) = a * SP(e_s) + b * SIP(e) \tag{3}$$

where e_s denotes the sender of e, a and b are set by the user, $0< a, b \leq 2$. In fact, in the *CP* formula, a feedback mechanism is hidden. From the above descriptions, we can see that the reading order of a user in this time affects *SP* value in the next time. Thus, the *CP* value changes with the reading behavior of a user.

5.3 Spam filtering

The spam filtering is a very important function of ECIPA. As Segal et al. [24] pointed out, "*the best approach is a multifaceted one*, combining various forms of filtering ...". ECIPA combines rule-based and statistical methods. The left side of Figure 6 shows the working flow of the FA in ECIPA. FA consists of two types agents: rule-based and statistic-based. When a new email arrives, the rule-based agent uses default rules to judge whether it is a spam or not. In fact, the rules refer to some of the concepts and the corresponding instances stored in the ontology of ECIPA. The purposes using

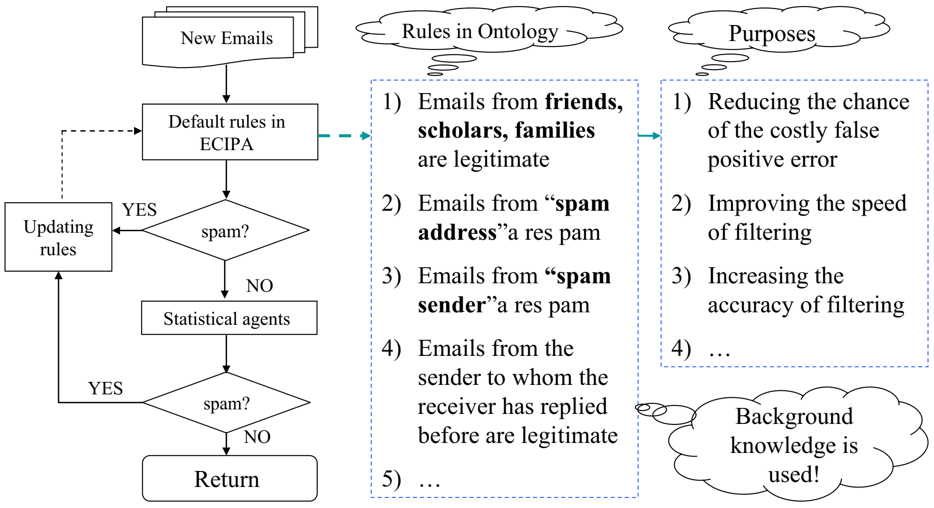

New Emails
Default rules in ECIPA
spam?
YES
Updating rules
NO
Statistical agents
YES
spam?
NO
Return
Rules in Ontology
1) Emails from **friends, scholars, families** are legitimate
2) Emails from "**spam address**"a res pam
3) Emails from **"spam sender**"a res pam
4) Emails from the sender to whom the receiver has replied before are legitimate
5) …
Purposes
1) Reducing the chance of the costly false positive error
2) Improving the speed of filtering
3) Increasing the accuracy of filtering
4) …
Background knowledge is used!

Fig. 6 The filtering process of FA.

such rules are given in the right-hand of Figure 6. If the rule-based agent fails to classify that email, ECIPA passes it to the statistic-based agent. For simplification, we still call the statistic-based agent as FA.

Similar to ATC tasks described in [23], the main issues, when building a filtering agent, include dataset preparation, feature selection, email representation, etc. [25, 32, 33] give the detailed and formal definition about such a process. Before the filtering agent is available, we conduct a training phase to train multiple statistic-based filtering agents. The training phase is divided into four steps: firstly, dividing the training dataset into Q partitions; secondly, training the kth Naive Bayesian (NB) [16] agent on the kth subset ($k = 1, 2, \ldots, Q$); thirdly, constructing the *training matrices* (includes the *spam matrix* and the *legitimate matrix*); fourthly, computing a weight for each agent using correspondence analysis. After that, when a new email arrives, Q agents classifies it according to their own weights. The key problem in FA lies in how to build the training matrices. For simplification, we only consider the *spam matrix* below. Suppose that $TM_{(N\times(Q+1))}$ is the matrix, N is the total count of the training spam. The ith line of TM represents the ith training spam, which mainly reflects the performance of each filter on the ith training email. Let the ith line vector in TM be v_i, and $v_i = \langle p_{i,1}, p_{i,2}, \ldots, p_{i,Q-1}, p_{i,Q}, p_{i,Q+1}\rangle$, where $p_{i,k}(k = 1, \ldots, Q)$ is the posterior probability of the ith training email belonging to spam, which is computed by the kth agent, and $p_{i,Q+1} = 1$.

The NB algorithm is adopted to learn a filter for each statistic-based agent in FA. Each statistic-based agent votes a new email according to its weight. Correspondence Analysis (CA) is adopted to generate the weight for each of such agents. The main idea of correspondence analysis is to develop simple indices that show the relations between the row and column categories. These indices tell us simultaneously which column category has a greater weight in a row category and vice-versa. Correspondence analysis is also related to the issue of dimension reduction of the table. Algorithm 1 shows the CA process on the spam or legitimate training matrix.

Here we describe CA process in FA for the *spam matrix*. For the matrix $TM_{(N\times(Q+1))}$, the calculations of correspondence analysis may be divided into three main stages (see Algorithm 1). The first stage (Steps 1.1-1.6 in Algorithm 1) consists of some preprocessing calculations performed on $TM_{I\times J}$ $(I = N, J = Q + 1)$, which leads to the standardized residual matrix. In the second stage (Step 2 in Algorithm 1), a singular value decomposition (SVD) is performed on the standardized residual matrix to redefine it in terms of three matrices: $U_{I\times K}$, $\sum_{K\times K}$, and $V_{K\times J}$, where $K = min(I - 1, J - 1)$. In the third stage (Steps 3.1-3.2 in Algorithm 1), we use U, $\sum$, V to determine $Y_{I\times K}$ and $Z_{J\times K}$, the coordinates of the rows and columns of TM, respectively, in the new space.

$Y_{I\times K}$ is the principal coordinates of the rows of TM, and $Z_{J\times K}$ is the principal coordinates of the columns of TM. Note that not all K dimensions are necessary. Hence, we can reduce the dimension to $\widetilde{K}$, while some information is lost. Definition 1 is used to reflect the degree of information loss.

Definition 1 Information loss (IL) is defined as follows.

$$\mathrm{IL} = 1 - \sum_{i=1}^{\widetilde{K}} \lambda_i / \sum_{i=1}^{K} \lambda_i \tag{4}$$

where λ_i is the diagonal element in $\sum$.

According to Definition 1, given a value of IL, the user can compute the $\widetilde{K}$. In the new geometric representation, the rows z_i and $z_j (i, j = 1, 2, \ldots, Q)$ in Z, corresponding to columns i and j in TM, lie close to one another when filters i and j receive similar predictions from the collection of training emails. Similarly, Z_{Q+1} is corresponding to the Q+1 column of TM, which represents the real categorization of the training emails. Hence, we can define:

Definition 2 The *i*th Agent's Weight on Spam (WJ$_i$) is defined as follows.

$$WJ_i = \sum_{j=1}^{\widetilde{K}} |Z_{ij} + Z_{(Q+1)j}| / \sum_{j=1}^{\widetilde{K}} |Z_{ij} - Z_{(Q+1)j}| \tag{5}$$

where Z_{ij} is the *j*th element in Z_i, $i = 1, 2, \ldots, Q$.

Similarly, we can use correspondence analysis on the legitimate matrix. Then, we can work out the *i*th Agent's Weight on Legitimate (WL_i).

In the filter combination phase of the FA, the *i*th statistic-based agent's prediction or vote for a spam has a strength proportional to its assigned weight WJ_i. Thus, we can compute the posterior probability of a spam as follows:

$$p(c_1/e) = \sum_{i=1}^{Q} WJ_i * p_i(c_1/e) \tag{6}$$

where $p_i(c_1/e)$ is the posterior probability of a spam computed by the *i*th filter, e is a new email, c_1 denotes spam.

Algorithm 1 The CA process on training matrix TM

```
Data: $TM$. //training matrix
Result: $U$, $\Sigma$, $V$, $Z$. //intermediate variables
Process:
    Step 1.1. $sum = \Sigma_{i=1}^{I}\Sigma_{j=1}^{J} TM_{i,j}$;
    Step 1.2. $P = (1/sum)TM$;
    Step 1.3. $r = \langle r_1, r_2, \ldots, r_I\rangle, r_i = P_i + \ (i = 1, 2, \ldots, I)$;
    Step 1.4. $c = \langle c_1, c_2, \ldots, c_J\rangle, c_i = P_+ i \ (i = 1, 2, \ldots, J)$;
    Step 1.5. $D_r = \text{diag}(r_1, r_2, \ldots, r_I)$, $D_c = \text{diag}(c_1, c_2, \ldots, c_J)$;
    Step 1.6. $\widetilde{P} = D_r^{-1/2}(P - rc^T)D_c^{-1/2}$;
    Step 2.   $\widetilde{P} = U\Sigma V^T$;
    Step 3.1. $Y = D_r^{-1/2}U\Sigma$;
    Step 3.2. $Z = D_c^{-1/2}V\Sigma$;
    Step 4.   return $U$, $\Sigma$, $V$, $Z$.
```

Similarly, we can work out the posterior probability of legitimate(c_0) for e:

$$p(c_0/e) = \sum_{i=1}^{Q} WL_i * p_i(c_0/e). \tag{7}$$

In general, the decision rule selects the class for which the posterior probability, $p(c_j/e)$ ($j = 0$ or 1), is the largest one. This way minimizes the probability of making an error. While for the email filtering problem, we should consider a somewhat different rule that minimizes an expected loss or risk. Let c_{01} be the cost assigning an email e to c_0 (legitimate) when $e \in c_1$ (spam), i.e., c_{01} is the cost of *receive-error* (namely, false negative error); and c_{10} is the cost of *reject-error* (namely, false positive error). In practice, it can be very difficult to assign costs. In some situations, both c_{01} and c_{10} may be measured in monetary units that are quantifiable. However, in many situations, costs are a combination of several different factors measured in different units, such as money, time, quality of life. As a consequence, they may be the subjective opinion of an expert.

The overall expected cost (OEC) is used to denote the error cost of a filter:

$$\text{OEC} = c_{10}p(c_1/c_0)p_0 + c_{01}p(c_0/c_1)p_1 \tag{8}$$

where p_0 and p_1 are the prior probabilities of c_0 and c_1, respectively, and $p(c_1/c_0)$ is the probability of making a *reject-error*, and $p(c_0/c_1)$ is the probability of making a *receive-error*.

In order to minimize OEC, we can deduce the decision rule as follows:

Rule 1. Given a new email e, we classify it into spam if and only if:

$$\frac{p(c_1/e)}{p(c_0/e)} \geq \frac{c_{10}p_0}{c_{01}p_1}. \tag{9}$$

where the right hand side of the inequality is a constant, which is called α below.

6 Implementation and evaluation of ECIPA

This section reports a prototype implementation of the ECIPA system, as well as some initial evaluations of the filtering agent.

6.1 An illustrative example using ECIPA

This subsection describes how to use ECIPA to implement automated appointment based on the operable email. The left side of Table 3 shows a simple example of using the "appointment" command with its parameters. As shown in this table, the sender wants to make an appointment with the receiver at 9:00am on Oct 28, 2006. The participants also include another people, namely Wenbin.

Figure 7 shows the snapshot of ECIPA relative to the function of making appointment. First, the sender should login (Figure 7a). Then, the email type of appointment should be chosen (Figure 7b). After that, IA automatically shows an interface to input corresponding information about that appointment (Figure 7c). Once the sender clicks the "OK" button, an operable email enclosed this piece of information shown in the right side of Table 3 is generated automatically. When the operable email is sent to the receiver from the sender. The assistant of the receiver parses that command first. If the assistant finds that its owner is not free at that time, it will automatically ask the sender a new time. Otherwise, it responds automatically if and only if its owner endued the right to it in advance. If the ECIPAs of a sender or a receiver agree with the time, IA will show the results for the users (Figure 7d). QAA will alter the user before the appointment.

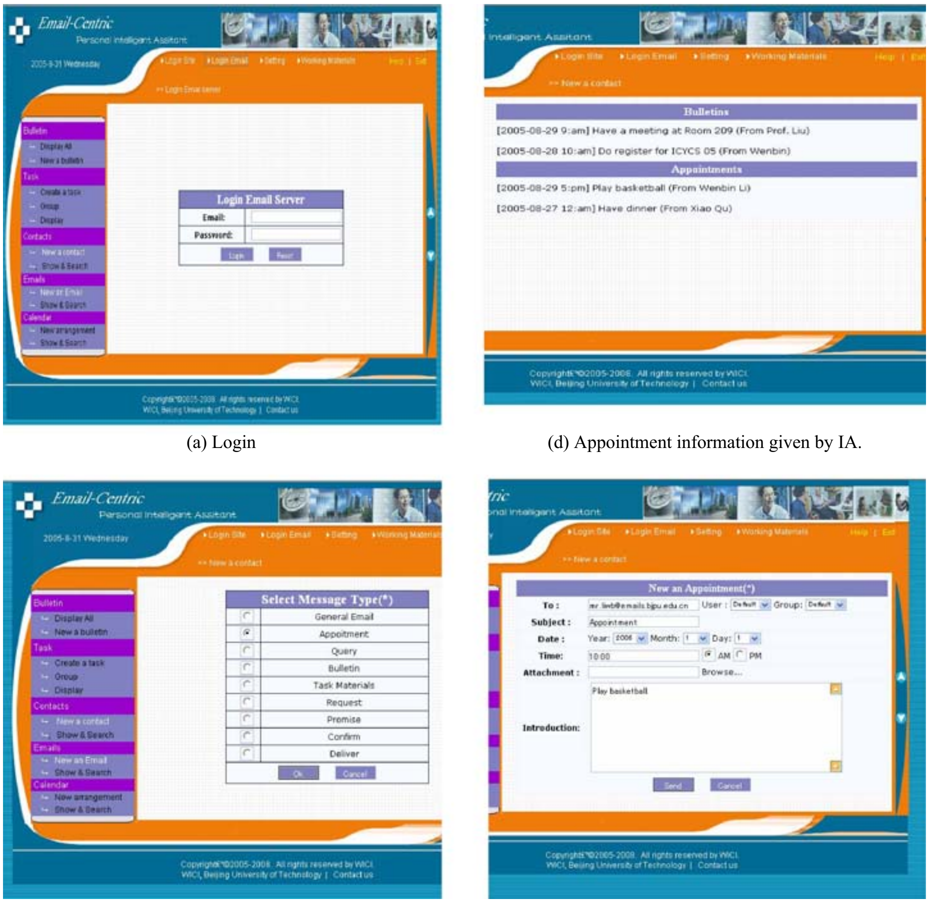
6.2 Filtering performance of FA

This subsection discusses the comparative performance of FA and other filtering algorithms. Below, we first depict the configuration of relative experiments. In the cost-insensitive classification tasks, two commonly used evaluation measures are *precision* and *recall*. These measures do not take into account the cost of two types of errors. In our opinion, an ideal filter should have the following features: (1) the *reject-error rate* is in a sustainable range $[0, \beta]$, where $\beta > 0$; (2) the *receiver-error rate* is a low value; and (3) the total error cost is very low.

Table 3 An example of using the "appointment" command.

(appointment)	⟨appointment⟩
:date Oct 28, 2006	⟨date⟩ Oct 28, 2006 ⟨/date⟩
:time 9:00am	⟨time⟩ 9:00am ⟨/time⟩
:event Netmeeting	⟨event⟩ Netmeeting ⟨/event⟩
:participant Wenbin	⟨participant⟩ Wenbin ⟨/participant⟩
	⟨/appointment⟩

The user wants to make an appointment with Wenbin at 9:00 am, on Oct 28, 2006.



Login Email Server

Email:

Password:

(a) Login

Bulletins

[2005-08-29 9:am] Have a meeting at Room 209 (From Prof. Liu)

[2005-08-28 10:am] Do register for ICYCS 05 (From Wenbin)

Appointments

[2005-08-29 5:pm] Play basketball (From Wenbin Li)

[2005-08-27 12:am] Have dinner (From Xiao Qu)

(d) Appointment information given by IA.

Select Message Type(*)

- General Email
- [✓ Appoitment]
- Query
- Bulletin
- Task Materials
- Request
- Promise
- Confirm
- Deliver

(b) Select email type. User can choice to write a traditional email or an Operable Email.

New an Appointment(*)

To: mr.liwb@emails.bjpu.edu.cn User: Default Group: Default

Subject: Appointment

Date: Year: 2006 Month: 1 Day: 1

Time: 10:00 [✓ AM] PM

Attachment:

Introduction: Play basketball

(c) Inputing information of an appointment, then ECIPA will generate an Operable Email automatically, and send it to relative sender.

Fig. 7 An illustrative example to show how to make an appointment via ECIPA.

In order to judge whether or not a filter has a good performance, we define three evaluation criteria as follows:

Definition 3 The reject-error rate (RJER):

$$\text{RJER} = F_{10}/(F_{10} + F_{00}). \tag{10}$$

Definition 4 The receive-error rate (REER):

$$\text{REER} = F_{01}/(F_{01} + F_{11}). \tag{11}$$

Definition 5 The total error cost (TEC):

$$\text{TEC} = c_{10}p_0 * RJER + c_{01}p_1 * REER. \tag{12}$$

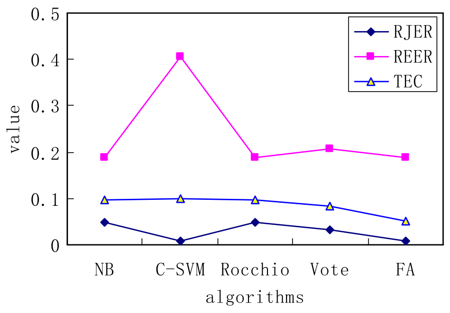
Table 4 The distribution of testing and training emails of the two mentioned corpora.

	The count of training emails		The count of testing emails	
	Legitimate	Spam	Legitimate	Spam
PU1	488	384	122	96
Ling-Spam	1929	384	483	97

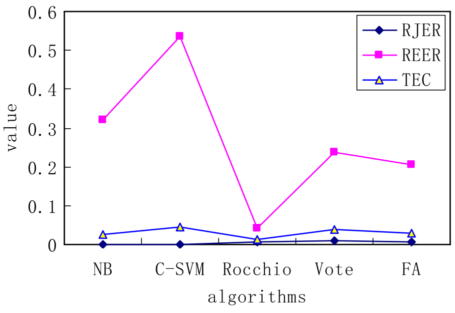
In the three definitions, F_{10} is the count of reject-error, F_{01} is the count of receive-error, F_{00} is the times correctly classifying legitimate emails, and F_{11} is the times correctly classifying spam, respectively. RJER and REER reflect the ratio of two types of errors, respectively. TEC represents the total cost generated by these two types of errors. A larger RJER shows that the filter makes the *reject-error* more often, and a larger REER indicates that the filter makes the *receive-error* more often. In general, the effect of RJER on TEC is much stronger than REER since c_{10} is larger than c_{01}. A larger TEC indicates a worse performance.

Several experiments have been carried out on two benchmark datasets for testing email filters' performance: Ling-Spam and PU1 (http://iit.demokritos.gr/skel/i-config/downloads/). Table 4 gives the distribution of testing and training emails of those two corpora. In our experiments, the ratio of feature subset selection is 1%; $Q = 8$; the feature subset selection method is based on information gain (IG) [31]; we adopt $c_{01} = 0.5$ and $c_{10} = 2$ for evaluating the filters.

Figure 8 shows the comparative results of five filtering algorithms with those two corpora. These methods are NB [11, 30], Rocchio [23], Vote [23], cost-SVM (C-SVM) [11] and FA, respectively. In this experiment, α is set to 1 for FA. For both corpora, all the RJER values of FA are very small. That is, FA makes very few positive errors on these two corpora. Also, C-SVM makes few reject-errors on both two test datasets. However, the REER and TEC in C-SVM are higher than the ones in FA with Ling-Spam and PU1. Figure 8a shows that FA has the lowest RJER, REER and TEC on PU1. For Ling-Spam, it seems that Rocchio filter is the best one. While this filer is not a cost-sensitive method on the one hand, and on the other hand, it makes more reject-errors than FA for PU1. From Figure 8b, we can see that all filters make few positive errors on Ling-Spam. Hence, for Ling-Spam, most of legitimate test emails may reside on the side represented by the legitimate training data, and are far way from the classifying hyperplane between legitimate and spam.



(a) PU1



(b) Ling-Spam

Fig. 8 The results of five filters on PU1 and Ling-Spam.

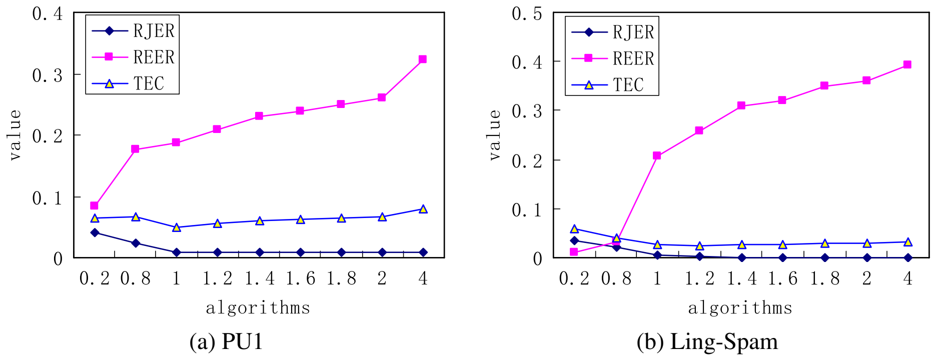


Fig. 9 The performance of FA on PU1 and Ling-Spam when α is changing.

In other words, it is difficult for filters to incorrectly classify legitimate test emails for Ling-Spam.

Figure 9 gives the performance of FA when α is changed. Figure 9 shows the performance on PU1 and Ling-Spam, respectively. From Figure 9a, we can see that RJER becomes much lower when α is changed to larger, while REER is opposite. When we adjust α to be greater, TEC becomes much lower at first, when α arrives at a threshold, TEC starts to become larger adagio. Figure 9b displays the same pattern.

From Figure 9a, we note that TEC and RJER reach the lowest point when α adopts the threshold value 1. Figure 9b shows that α should be set to the threshold value 1.2 if we expect to gain the lowest TEC and RJER on Ling-Spam. In real applications, we suggest that users should adopt the value of α which is a little larger than the threshold at which TEC and RJER gain their lowest value. For example, if PU1 is used as a dataset, α can be set to 1.1, and if Ling-Spam is used as a dataset, α can be set to 1.3.

7 Concluding remarks

The main contributions in this work can be summarized as follows. Firstly, we presented how to recast traditional emails for implementing intelligent applications and ubiquitous computing based on email. Email is indispensable to most user's work, and has significant impacts on both academic research and ordinary daily life. However, when people benefit from email, they often suffer from spam and too much manual work. To solve this problem completely, we proposed the operable email. Secondly, we developed an Email-Centric Intelligent Personal Assistant named ECIPA, by drawing on a variety of WI-related techniques. This assistant combines various WI-related techniques, such as the operable email, agent-based means etc. to produce a highly personal and automated system, with more features than found in a typical system today. The key and novel technical features of such an assistant include automated and cost-sensitive spam filtering; ontology-mediated classification, query and archiving; sorting/responding based on dynamic user behavior learning; intelligent cooperation based on the operable email. Thirdly, a proposed filtering method is adopted in such an assistant, which combines multiple NB filters based on its own dynamic weight of voting.

Acknowledgements The authors would like to thank all the reviewers. The work is (partially) supported by the NSFC major research program: "Basic Theory and Core Techniques of Non-Canonical Knowledge" (no. 60496322), NSFC project: "The Operable Email: Theory and Its Applications" (no. 60673015), Open Foundation of Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology, and Project (no. 06213558) of Department of Science and Technology of Hebei Province.

References

1. Amato, G., Straccia, U.: User profile modeling and applications to digital libraries. In: Proc. of ECDL'99, pp. 184–197 (1999)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Schneider, P.P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, UK (2002)
3. Bergman, R., Griss, M., Staelin, C.: A personal email assistant. Technical report HPL-2002-236, HP Labs Palo Alto http://citeseer.ist.psu.edu/bergman02personal.html (2002)
4. Cohen, W., Carvalho, V.R., Mitchell, T.M.: Learning to classify email into "speech acts". In: Proc. of the EMNLP'04, Hong Kong, pp. 309–316 (2004)
5. Deng, Y.H., Tsai, T.H., Hsu, J.: P@rty: a personal email agent. In: Proc. of Agent Technology Workshop, National Taiwan University, pp. 61–64 (1999)
6. Fawcett, T.: "In vivo" spam filtering: a challenge problem for data mining. KDD Explorations **5**(2), 140–148 (2003)
7. Ho, V., Wobcke, W., Compton, P.: EMMA: an email management assistant. In: Proc. of IEEE/WIC International Conference on Intelligent Agent Technology, IAT'03, pp. 67–74. IEEE, Los Alamitos, CA (2003)
8. Huang, Y.F., Govindaraju, D., Mitchell, T.M., Carvalho, V.R.D., Cohen, W.W.: Inferring ongoing activities of workstation users by clustering email. In: Proc. of CEAS'04 (2004)
9. Kassoff, M., Petrie, C., Zen, L.M., Genesereth, M.: Semantic email addressing: sending email to people, not strings. In: Proc. of AAAI 2006 Fall Symposium on Integrating Reasoning into Everyday Applications (2006)
10. Li, W.B., Zhong, N., Liu, C.N.: Design and implementation of an email classifier. In: Proc. of International Conference on Active Media Technology, AMT'03, pp. 423–430. Chongqing, China (2003)
11. Li, W.B., Liu, C.N., Chen, Y.Y.: Combining multiple email filters of Naive Bayes based on GMM. Acta Electronica Sinica **34**(2), 247–251 (2006)
12. Li, W.B., Zhong, N., Liu, J.M., Yao, Y.Y., Liu, C.N.: Perspective of applying the global e-mail network. In: Proc. of IEEE/WIC/ACM Web Intelligence, WI'06, pp. 117–120. Hong Kong (2006)
13. Liu, J.M.: Web intelligence (WI): What makes wisdom Web? In: Proc. of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03, pp. 1596–1601. Morgan Kaufmann, San Francisco (2003)
14. Maes, P.: Agents that reduce work and information overload. Comm. ACM **37**(7), 30–40 (1994)
15. Martin, S., Sewani, A., Nelson, B., Chen, K., Joseph, A.D.: Analyzing behaviorial features for email classification. In: Proc. of the IEEE Second Conference on Email and Anti-Spam, CEAS'06 (2005)
16. McCallum, A., Nigam, K.: A comparison of event models for Naive Bayes text classification. In: Proc. of AAAI-98 Workshop on Learning for Text Categorization, pp. 41–48 (1998)
17. McDowell, L., Etzioni, O., Halevy, A., Henry, L.: Semantic email. In: Proc. of the Thirteenth Int. WWW Conference, WWW'04, pp. 244–254. New York, USA (2004)
18. Mineau, G.W.: A simple KNN algorithm for text categorization. In: Proc. of ICDM'01, pp. 647–648
19. Mitchell, T.M., Wang, S.H., Huang, Y.: Cheyer: Extracting knowledge about users' activities from raw workstation contents. In: Proc. of AAAI'06, pp. 181–186 (2006)
20. Moreale, E., Watt, S.: An agent-based approach to mailing list knowledge management, vol. 2926, pp. 118–129. Springer, LNAI (2003)
21. Noy, N., McGuinness, D.L.: Ontology development: a guide to creating your first ontology. Technical report SMI-2001-0880, Standford Medical Informatics, Stanford University, Stanford, CA 94305 (2001)

22. Pires, F.M., Abreu, S.: An evolvable rule-based email agent, vol. 2902, pp. 394–408. Springer, LNCS (2003)
23. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002)
24. Segal, R., Crawford, J., Kephart, J., Leiba, B.: SpamGuru: an enterprise anti-spam filtering system. In: Proc. of the First Conference on Email and Anti-Spam, CEAS'04 (2004)
25. Stephen, R., Hugo, Z., Michael, T.: Simple BM25 extension to multiple weighted fields. In: Proc. of CIKM'04, pp. 42–49 (2004)
26. Sun, D., Tran, Q.A., Duan, H., Zhang, G.: A novel method for Chinese spam detection based on one-class support vector machine. J. Inform. Comput. Sci. **2**(1), 109–114 (2005)
27. Su, Y.L., Zheng, L., Zhong, N., Liu, C.N., Liu, J.M.: Distributed reasoning based on problem solver markup language (PSML) - a demonstration through extended OWL. In: Proc. of EEE'05, pp. 208–213 (2005)
28. Tinapple, D., Woods, D.: Message overload from the inbox to intelligence analysis: how spam and blogs point to new tools. In: Proc. of Human Factors and Ergonomics Society 47th Annual Meeting, pp. 419–423. Denver, CO (2003)
29. Xia, F., Liu, W.Y.: An agent for semi-automatic management of emails. In: Proc. of 5th Asia Pacific Conference on Computer Human Interaction, APCHI'02, PRC, pp. 709–719 (2002)
30. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proc. SIGIR'99, pp. 42–49 (1999)
31. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proc. of 14th International Conference on Machine Learning, ICML'97, pp. 412–420. Nashville, TN, US (1997)
32. Wu, S.T., Li, Y.F., Wu, Y.: Automatic pattern-taxonomy extraction for web mining. In: Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence, pp. 242–248 (2004)
33. Wu, S.T., Li, Y.F., Wu, Y.: Deploying approaches for pattern refinement in text mining. In Proc. of ICDM'06, 1157–1161 (2006)
34. Zhang, L., Zhu, J.B., Yao, T.S.: An evaluation of statistical spam filtering techniques. ACM Trans. Asian Lang. Inform. Process. **3**(4), 243–269 (2004)
35. Zhong, N.: Developing intelligent portals by using WI technologies. In: Proc. of Wavelet Analysis and Its Applications, and Active Media Technology, vol. 2, pp. 555–567. World Scientific (2004)
36. Zhong, N., Liu, J.M., Yao, Y.Y.: In search of the wisdom Web. Computer **35**(11), 27–31 (2002) (special issue)
37. Zhong, N., Liu, J.M., Yao, Y.Y.: Envisioning intelligent information technologies from the standpoint of Web intelligence. Commun. of the ACM **50**(3), 89–94 (2007)