List of Abbreviations

Abbreviation	**Meaning**
AI	Artificial Intelligent
API	Application Program Interface
Bcc	Blind carbon copy
Cc	Carbon copy
CNRI	Corporation National Research Initiative
E-mail	Electronic mail
EMFA	Email Filtering Agent
FTP	File transfer Protocol
HTTP	Hyper Text Transfer Protocol
IMAP	Internet Message Access Protocol
POP	Post Office Protocol
POP3	Post Office Protocol Version 3
SDK	
SMTP	Simple Mail Transfer Protocol

Abstract

As the number of users connects to the internet increases rapidly, Electronic mail **(E-mail)** is quickly becoming one of the fastest and most economical forms of communication available, since E-mail is extremely cheep and easy to send. As our lives have become ever increasingly tied up to the online world, the volume of E-mails coming into our inboxes has also been increasing, so the problem of email filtering is a critical one. The current solution usually consists of using E-mail filtering program that can filter incoming E-mail according to user specified rules; this program is the **E-mail Filtering Agent (EMFA).**

EMFA system splits E-mails into categories, this would help to automate the process of sorting through E-mail and applying actions (such as deleting unwanted mail or forwarding or replying messages to a specific address).

The agent learns the actions to be performed on E-mail and the features to be used in the classification task from predefined examples in the system. It uses **machine learning** to classify the messages into two lists: **Negative** list (that contains unwanted messages) and **Positive** list (that contains the messages that must be forwarded or replied) and then discard the contents of the negative list and reply or forward the positive list messages.

To implement EMFA system, JAVA language was used. It provides a set of abstract classes defining objects that comprise the E-mail system also supporting the creation of sophisticated user interfaces.

Acknowledgment

I would like to express my sincere appreciation to my supervisor, Dr. Jamal M. Kadhem, for giving me the major steps to go on to explore the subject, sharing with me the ideas in my research "Design and Implementation of Email Filtering Agent" and discuss the points that I felt they are important.

Grateful thanks for the Head of Department of Computer Science, Dr. Taha S. Bashaga,

Also, I wish to thank the staff of Computer Science Department at Al-Nahrain University for their help.

I would like to say "thank you " to my faithful friends for supporting and giving me advises.

Dedication

To my father who supports me

Republic of Iraq
Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Sciences

Design and Implementation of Email Filtering Agent

A thesis

Submitted to the College of Science / AL-Nahrain University in partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science.

By

Huda Fawzi Al-Shahad

(B.Sc.2002)

Supervisor

Dr.Jamal M. Kadhem

Sofar **1429**

February **2008**

Table of Contents

بسم الله الرحمن الرحيم

يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ

صدق الله العلي العظيم

المجادلة(١١)

الخلاصة

مع تزايد عدد مستخدمي الشبكة الدولية للمعلومات (الانترنت) بصورة مطرده .اصبح البريد الالكتروني احد اسرع وارخص انواع الاتصالات المتوفرة حيث ان البريد الالكتروني قليل الكلفه وسهل الارسال . ومع تزايد ارتباط حياتنا بعالم الانترنت، حجم البريد الالكتروني الذي يصل الينا في تزايد ايضا ، ولهذا مشكلة تصفية وترشيح البريد الالكتروني اصبحت مشكله حرجه بعض الشيء . الحل المستخدم في الوقت الحاضر يتضمن برنامج لتصفية وترشيح البريد الالكتروني القادم بالاعتماد على قواعد معينه، هذا النظام يسمى النظام الوكيل لتصفية وترشيح البريد الالكتروني (Email Filtering Agent)(EMFA

هذا النظام يقوم بتوزيع الرسائل الالكترونيه اوتماتيكيا الى فئات معينه، هذا يساعد في جعل عملية تنظيم البريد الالكتروني اوتماتيكية والتي تقوم على تحديد الاسبقيه للرسائل الالكترونية وتقرير افعال معينه (مثل مسح الرسائل الغير مرغوبه وتمرير او اجابة الرسائل الالكترونية القادمة من عنوان معين).

النظام يحتوي على طراز تعليمي بسيط لايستعمل رموز ضمن نص الرسالة الالكترونية، بل يستخدم بدلا عن ذلك مجموعة محددة ومعرفه مسبقا من الرموز المأخوذة من بعض حقول الرسالة الالكترونية (بأستخدام بعض الخواص من الحقول عنوان المرسل والتاريخ).

البرنامج الوكيل يتعلم الافعال التي ينفذها على البريد الالكتروني والميزات المستخلصة والتي تستعمل في عملية التصنيف من امثلة سابقة مخزونة في النظام.

النظام يستخدم طراز تعليمي غير خاضع لسيطرة المستخدم (Unsupervised Machine Learning) لتقسيم الرسائل الالكترونية الى قائمتين : القائمة السلبية Negative List التي تحتوي على الرسائل الالكترونية التي يجب حذفها والقائمة الايجابية Positive List التي تحتوي على الرسائل الالكترونية التي يجب الاجابة عليها او تمريرها بعد ذلك يحذف محتويات القائمة السلبية ويرد على او يمرر الرسائل الموجوده في القائمة الايجابية.

استخدمتJAVA جافا لتطبيق نظام ال EMFA حيث تنها مزوده بمجموعة من المختصرات تعرف العناصر التي تشكل نظام البريد الالكتروني، وايضا تدعم خلق واجهات عرض متطورة.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة النـهرين

كلية العلوم/ قسم علوم الحاسوب

تصميم وتنفيذ نظام وكيل لتصفية وترشيح البريد الالكتروني

رسالة

مقدمة الى كلية العلوم، جامعة النـهرين

كجزء من متطلبات نيل درجة الماجستير في علوم

الحاسوب

ِمنِ قبَل

هدى فوزي الشهد

بكلوريوس

٢٠٠٢

المشرف

د.جمال محمد كاظم

صفر 1428

شباط 200٨

Certification of the Examination Committee

We chairman and members of the examination committee certify that we have studies this thesis "**Design and Implementation of Email Filtering Agent**" presented by the student **Huda Fawzi Al-Shahad** and examined her in its contents and that we have found it worthy to be accepted for the degree of Master of Science in Computer Science .

Signature:

Name: **Dr. Lamia H.Khalid**
Title : **Assistant Professor**
Date : **/ /2008**
(Chairman)

Signature:	Signature:
Name: **Dr. Loay E. George**	Name: **Dr. Taha S. Bashaga**
Title : **Assistant Professor**	Title : **Lecturer**
Date : **/ /2008**	Date : **/ /2008**
(Member)	**(Member)**

Signature:

Name: **Dr. Jamal M. Kadhem**

Title : **Lecturer**

Date : **/ /2008**

(Supervisor)

Approved by the Dean of the Collage of Science, Al-Nahrain University.

Signature:

Name: **Dr. LAITH ABDUL AZIZ AL-ANI**
Title : **Assist. Prof**.
Date : **/ /2008**
(Dean of Collage of Science)

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science/Collage of Science/Al-Nahrain University, by **Huda Fawzi Al-Shahad** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Signature:
Name: **Dr. Jamal M. Kadhem**
Title: **Lecturer**
Date: **/ /2008**

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:
Name: **Dr. Taha S. Bashaga**
Title: **Head of department of Computer Science, Al-Nahrain University.**
Date: **/ /2008**

Chapter One
General Introduction

1.1 Introduction

E-mail communication has become more available, it is being used as a fundamental communication tool by millions of people around the world. E-mail is used to organize meetings, manage virtual work teams, discuss work-related proposal, make announcements and solve problems. A user may receive tens or even hundreds of emails every day. E-mail users commonly try to manage the large amount of emails they receive by using automatic approach to processing the email which is called email filtering [Che06].

Email filtering is the processing of e-mail to organize it according to specified criterion. Most often this refers to the automatic processing of incoming messages, but the term also applies to the intervention of human intelligence in addition to artificial intelligence, and to outgoing emails as well as those being received. Email filtering software inputs email and for its output, it might pass the message through unchanged for delivery to the user's mailbox, it might redirect the message for delivery elsewhere, or it might even throw the message away. Some mail filters are able to edit messages during processing [Wik07].

The problem of email filtering is a very practical one. As our lives become ever increasingly tied to the online world, the volume of email coming into our inboxes has also been increasing steadily. The solution usually consists of using a mail filtering program that can sort incoming mail based on user conditions. Here is where machine learning comes to the rescue. If we can train a machine, or rather a computer program, to learn what email should go where, Junk mail would be discarded.

Instead of simply offering tools that are manipulated by human users, an email agent attempts to organize email massages automatically based on its knowledge about each individual user.

Agent architecture has been developed which observes a user performing tasks, and identifies features which can be used as training data by a learning algorithm. Using the learned profile, an agent can give a device to the user on dealing with new situations. The agent uses sender and recipient field of the message and the keywords from the subject field. Other information such as whether the message has been read, whether it is a reply to a previous message, etc [Pay97].

1.2 Literature Survey

- In 1994, Y. Lashkari *et al.* [Las94] have presented interface agents for the electronic mail domain used in commercial email application. The interface agent monitors the actions of the user over long period of time, finds recurrent patterns and offers to automate them. They had used Memory Based Reasoning algorithm. A problem was occurred when he used it, so collaborative agents is used. Collaborative agents consist of a number of agents belonging to other users. It allows each agent to built a model of each agents area of expertise.

- In 1994, T. Payne [Pay94] had described the development of an agent which employs machine learning techniques to discover rules for filtering email. The agent makes use of a machine learning approach to build up a user model or profile of the users' interests. User actions are observed for use in creating rules, and these rules are used to filter incoming mail messages. The CN2 induction algorithm is used to induce rules based on these observations. He had show that features from the body of an email message can be utilized as well as more traditional features such as the subject or sender fields.

- In 1999, Y.Chang [Cha99] had presented Pine mail system under UNIX that use sort mail algorithm to solve the problem of email filtering by using text classification. Text classification takes a document consisting of a set of words and tries to categorize the document among a specified set of topics, or classes, then put each email as belonging to a particular folder.

- In 2000, J. Rennie [Ren00] had presented a filtering system, ifile, that use to automate classification techniques to filter personal email. Ifile use an efficient supervised Naive Bayes implementation which can both built a classification model and filter a new email messages. Each document in a user's corpus was labeled according to a model built on the rest of the email messages. He has discussed the result of classifying the text of message and selects the appropriate folder.

- In 2000,M. Pazzani [Paz00]had discussed a variety of approaches that could learn a profile of a user's interests for filtering mail. He has used text mining algorithm to create understandable profiles of a user's interests and word pairs as terms increases the acceptance of profiles and learned from noisy training data. He had found that subjects have little confidence in learned rules for text classification and suggest that using word pairs as terms may improve the user acceptance learned prototype profiles.

- In 2003,N.Turenne [Tur03]had discussed the term clustering method evaluated by a text classification task involving user profile. The clustering algorithm is based on the extraction of terms in a training corpus. He had demonstrated the utility of his approach in electronic mail classification. The e-mail folder represents the user areas of interest. Several classification strategies that run over a personal set of text have been compared to form mailing lists.

1.3 The aim of Thesis:

The aim of project is to build an agent that delete, forward and reply mail messages. It is used to assist the user in classifying mail message into different folders, and then filtered them according to specifying conditions.

1.4 Thesis Layout

This thesis was organized as follows:

- ***Chapter two:*** This chapter presents the structure of electronic messages and how to deal with them and how to construct the intelligent agent to be use for email filtering.

- ***Chapter three:*** This chapter presents the design requirements and considerations of the EMFA, and then the structure of the system will be explained with the modules and the algorithms that are used to implement this system.
- ***Chapter four:*** This chapter presents user interface and evaluation of the designed system.
- ***Chapter five:*** This chapter introduces conclusions on this work, with recommendations for future.

Chapter Two
Email and Intelligent Agent overview

2.1 Introduction

Email stands for electronic mail. While it originated as an after thought to the beginnings of the Internet, it is currently one of the most popular services of the Internet. Email does not have to be Internet based. It can be an in-house service that reaches only a certain population. Internet email can be sent to anyone in the world who has an Internet email address [Cam96].

Electronic mail is a natural use of networked communication technology that developed along with the evolution of the Internet. Indeed, message exchange in one form or another has existed from the early days of timesharing computers. Network capable email was developed for the ARPANET shortly after its creation, and has now evolved into the powerful email technology that is the most widely used application on the Internet today [Cro78].

Key events and milestones in the invention of email are described below:

- **Timesharing computers:** With the development in the early 1960's of timesharing computers that could run more than one program at once, many research organizations wrote programs to exchange text messages and even real-time chat among users at different terminals. As is often the case, more than one person at the same time noticed that it was a natural use of a new technology to extend human communications. However, these early systems were limited to use by the group of people using one computer [Cro78].

- **SNDMSG & READMAIL:** In the early 1970's, Ray Tomlinson was working on a small team developing the TENEX operating system, with local email programs called SNDMSG and to extend the addressing to the network, Tomlinson chose the "commercial at" symbol to combine the user and host names, providing the naturally meaningful notation "user@host" that is the standard for email addressing today. These early programs had simple functionality and were command line driven, but established the basic transactional model that still defines the technology email gets sent to someone's mailbox [Cro78].
- **MAIL & MLFL. In 1972,** the commands MAIL and MLFL were added to the FTP (file transfer protocol) program to provide standard network transport capabilities for email transmission. FTP sent a separate copy of each email to each recipient, and provided the standard ARPANET email functionality until the early 1980's when the more efficient SMTP (simple mail transfer protocol) protocol was developed. Among other improvements, SMTP enabled sending a single message to a domain with more than one address, after which the local server would locally copy the message to each recipient [Cro78].

- **Commercial Email. In 1988,** Vinton Cerf arranged for the connection of MCI Mail to the NSFNET through the Corporation for the National Research Initiative (CNRI) for "experimental use", providing the first sanctioned commercial use of the Internet. Shortly thereafter, in 1989, the Compuserve mail system also connected to the NSFNET, through the Ohio State University network [Cro78].

- **Online Services. In 1993,** the large network service providers America Online and Delphi started to connect their proprietary email systems to the Internet, beginning the large scale adoption of Internet email as a global standard [Cro78].

As the most popular application on the Internet, electronic mail (email) has become an increasingly critical tool to running a business and/or one's daily life. Many people are overloaded with a large number of emails on a regular basis, and important messages can often get buried under piles of other emails by mistake. Existing email software typically offers functions that help users manage multiple email accounts, organize their emails into folders, specify filters to sort emails automatically by subject, sender, etc. and to block unwanted emails [Den98].

2.2 Artificial Intelligent

The science of artificial Intelligent (AI) is approximately forty years old, dating back to a conference held at Dartmouth in 1958. In the early years, the excitement of both scientists and the popular press tended to overstate the real world prowess of AI system. The early successes were followed by a slow realization that what was hard for people to do but almost impossible for computers to do. The promise of the early years has never been fully realized, and AI research and the term artificial intelligence have become associated with failure and technology.

After 40 years of work, three major phrases of development in AI research can be identified. In the early years, much of the work dealt with formal problems that were structured and had well-defined problem boundaries .This included work on math-related skills such as proving theorems, geometry, calculus, and playing games such as chess.

In this first phase, the emphasis was on creating general "thinking machines" which would be capable of solving broad classes of problems.

A second phase began with the recognition that the most successful AI projects were aimed at very narrow problem domains and usually encoded much specific knowledge about the problem to be solved.

At the third phase, much of the AI community has been working on solving the difficult problems of machine vision and speech, natural language understanding and translation, commonsense reasoning, and robot control. A branch of AI known as connectionism regained popularity and expanded the range of commercial application through the use of neural networks for data mining ,modeling ,and adaptive control. Recently, the explosive growth in the Internet and distributed computing has led to the idea of agents that move through the network, interacting with each other and performing tasks for their users.

Intelligent agents use the latest AI techniques to provide autonomous, intelligent, and mobile software agents, thereby extending the reach of users across networks [Big01].

2.3 Agent

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. A human agent has eyes, ears, other organs for sensors, and has hands, legs, mouth, and other body parts for effectors. A robotic agent substitutes cameras and infrared range finder for the sensors and various motors for the effectors. A generic agent is diagrammed in figure (2.1).

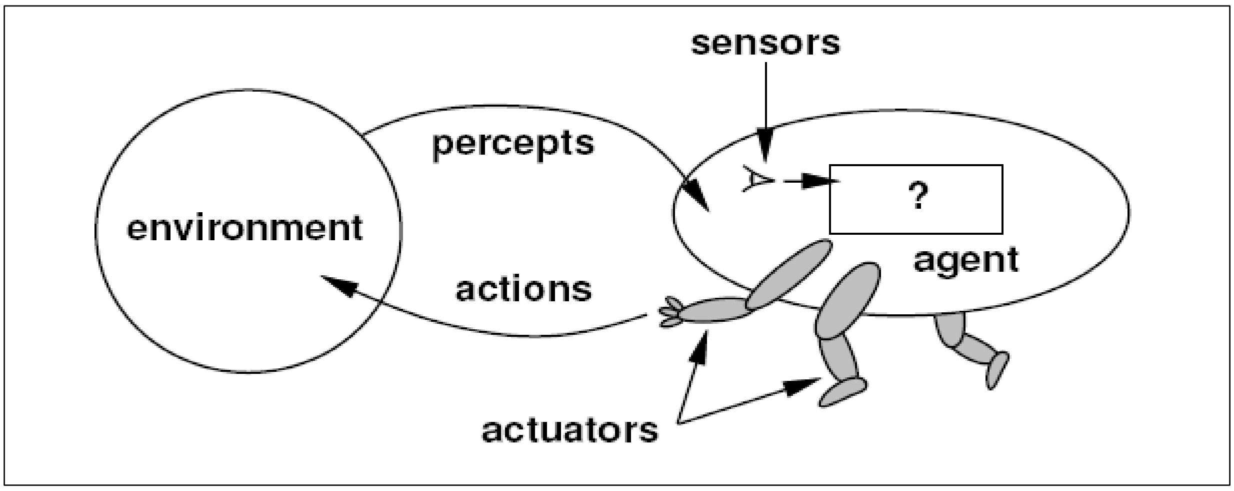


Figure (2.1) Agent diagram

A software agent should display some of the characteristics that are associate with human intelligence: learning, inference, adaptability, independence, creativity, etc. In other words, a software agent is a program that its users can delegate tasks to, rather than commanding it to perform the tasks [Hui00].

Any software agent must have the following characteristics:

- It must be autonomous, which means that it can operate on its own without a direct human command. Software agents have individual internal States and goals, and act in such a manner as to meet their goals on behalf of their users.
- It must be personalized; that is, it acquires the user's interests and adapts as it evolves over time.
- It must be persistent, either run continuously or save its state, so that the user can see the agent as a stable entity and develop a trust relationship with it.

There are some examples of software agents in practice:

• An agent that sorts through e-mail messages and filters out unsolicited commercial e-mail commonly known as spam.

• An agent that searches for information on the Web, either directly by looking at Web sites, or by sending queries to one or more search engines.

• An agent that learns what type of news stories a user is interested in, and fetches suitable content from news sources such as CNN, MSNBC, and other major media sites.

• An agent that compares prices on products for users across a variety of sites, and offers "comparison shopping" through a Web interface.

• Mobile agents that send themselves to a central meeting site to exchange information and barter for prices on products or services, and then return to their users with prices and costs.

• An agent that monitors a source of information (such as a store catalog), for changes relating to the interest of a user, such as movie releases starring a particular actor or actress, or new novels by a particular author. Such an agent might e-mail one or more users to alert them to this change.

Of course, software agents can be written to perform all sorts of tasks, limited only by one's imagination. For some time, artificial intelligence researchers have been predicting that software agents will be the next "killer application". While such a vision may be overly optimistic, software agents are likely to be a significant growth area in the future.

As a language, Java is ideally suited to the development of software agents. With its built-in support for HTTP communication, agent developers can easily make their agents "Web aware," without the need to write a custom HTTP implementation. This helps agent developers concentrate on the application, without being overly burdened by developing the network code [Dav02].

2.3.1 Agent Architectures

All the focusing in the beginning concerned with agent theory the construction of formalisms for reasoning about agents, and the properties of agents expressed in such formalisms. The aim is to shift the emphasis from theory to practice. Considering the issues surrounding the construction of computer systems that satisfy the properties specified by agent theorists. It specifies how the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions and future internal state of the agent. Architecture encompasses techniques and algorithms that support this methodology [Woo98].

Agent theories are specifications which describe how agents are conceptualized, what properties they should have and how these properties should be formally represented and reasoned about.

Agent architectures represent the move from specification to implementation. They can be thought of as software engineering models of agents; researchers in this area are primarily concerned with the problem of designing software or hardware systems that will satisfy the properties specified by agent theorists.

2.3.1.1 Classical Approach: Deliberative Architectures

The term "deliberative agent" seems to have derived from use of the term "deliberate agent" to mean a specific type of symbolic architecture. A deliberative agent or - agent architecture is based on a strong notion of agents and contains an explicitly represented, symbolic model of the world, and decisions (for example about what actions to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation.

In classical AI, symbolic representations provide an interface between the independent information processing units. Reasoning in this systems is realized through logics, mostly second-order module logics.

The idea of deliberative agents based on purely logical reasoning is highly appealing, but there are at least two important problems to be solved:

1. The transduction problem: that of translating the real world into an accurate, adequate symbolic description, in time for that description to be useful.
2. The representation/reasoning problem: that of how to symbolically represent information about complex real world entities and processes (since computers may simulate reasoning but not the interaction with the environment), and how to get agents to reason with this information in time for the results to be useful.

2.3.1.2 Alternative Approaches: Reactive Architectures

A Reactive Architecture is an architecture that does not include any kind of central symbolic world model and doesn't use any complex reasoning. The problems associated with symbolic AI have led some researchers to question the viability of the whole paradigm, and to the development of what are generally known as reactive architectures.

2.4 From AI to Intelligent Agents

As in any AI application, what the agent is expected to do, and in what domain, will have a significant impact on the type of knowledge representation that use. If the agent has a limited number of situations it needs to respond to, maybe hardcoding the intelligence into procedural program code are the solution. If the agent has to build or use sophisticated models of the problem domain and solve problems at different levels of abstraction, then frames or semantic nets are the answer. In many applications, it needs to use a mixture of these knowledge representations.

Whether learning is a desirable function depend on the domain the intelligent agent will work in, as well as the environment. If the agent is long-lived and will perform similar tasks many times during its lifetime, then learning can be used to improve its performance. But adding learning would be overkill if the agent will be used only occasionally.

The intelligent agent must have an equivalent source of information about the world in which it lives. This information comes in through its sensors, which may not be same way that do, but it still has to be able to gather information about its environment. This could be done actively, by sending messages to other agents or system, or it could be done passively by receiving a stream of event messages from the system, the user, or other agents. Just like people, the software agent must be able to distinguish the normal events from the significant events. In a modern GUI environment such as Windows or Macintosh, the user generates a constant stream of events to the underlying windowing system. The agents can monitor this stream and must recognize sequences of basic user actions (mouse movement, pause, click, mouse movement, pause, double-click) as signaling some larger-scale semantic event or user action.

If the agent works in the e-mail or newsgroup monitor domain, it will have to recognize when new documents arrive, whether the user is interested in the subject matter or not, and whether to interrupt the user at some other task to inform him of the newly available information. All of this falls into the realm of perception. Being able to notice or recognize information hidden in data is not easy. It requires intelligence and domain knowledge. Thus being called "perceptive" is a compliment usually reserved for intelligent people. To be useful personal assistants, agent must be perceptive.

The design of agents should be in such a way that the messages don't have to be very perceptive. The user needs to explicitly tell the agent what to do and how to do it. The events that are generated could contain all the information the agent needs to determine the current state and the appropriate action [Big01].

2.4.1 Intelligent Agent Framework

2.4.1.1 Requirements

The first step of any software development project is the collection of requirements from the intended user community. The first requirement is that the intelligent agent framework be practical. No practical in the sense that it is product-level code ready to put into production, but in that the basic principles and thrust of the design is applicable to solving real-world problems. Another requirement is that the architecture must be flexible enough to support the applications.

To summarize the requirements, a simple flexible architecture that is focused on intelligent agent issues is needed. It must be practical so it can solve realistic problems and must have a decent user interface so its functions and limitations will be readily apparent to the user [Big01].

2.4.1.2 Design Goals

Requirements come from the users and tell them what functions or properties the product must have in order to be successful. Having a validated set of requirements is useful, because it focuses the energy on the important stuff. It is just as important to have a clear set of design goals that can use to guide the technical decisions which must be made as the solution that meets those requirements are developed.

There are some fundamental approaches which will drive the design. **The first approach** is that the intelligent agents can be viewed either as adding value to a single standalone application, or as a freestanding community of agents which interact with each other applications. This approach is the least complex because it could view the agent as a simple extension of the application functionality. By providing the intelligent agent functions as an object-oriented framework, intelligent behavior can be easily added to any Java application.

The second approach is more agent-centric, where the agents call the shots and monitor and drive the applications. Here the agent manager is an application in its own right and must interface with other applications which are driven by the agents. The complexity here is that a generic mechanism for application communications should be defined through the agent manager [Big01].

2.4.2 Application of Intelligent Agent

There are several orthogonal dimensions along which agent applications could be classified. They can be classified by the type of agent, by the technology used to implement the agent, or by the application domain itself.

1. Industrial Applications: Industrial applications of agent technology were among the first to be developed, as early in 1987 agents are being applied the contract net task allocation protocol in a manufacturing environment. Today, agents are being applied in a wide range of industrial application [Jen98].

2. Commercial Applications:

- **Information Management:** As the richness and diversity of information available in everyday lives has grown, so the need to manage this information has grown [Jen98].
- **Information filtering:** Every day, enormous amounts of information are presented (via email and newsnet news, for example), only a tiny proportion of which is relevant or important. These information need to be sorted and focused on them [Jen98].
- **Email filtering:** describe an electronic mail filtering agent program that learns to prioritize, delete, forward, sort, and archive mail messages on behalf of a user [Jen98].

3. Medical Applications: Medical informatics is a major growth area in computer science. New applications are being found for computers every day in the health industry [Jen98].

4. Games: It's describing several applications of agent technology to computer games. For example, the designers have developed a version of the popular Tetris computer game [Jen98].

2.5 Email Agent

Electronic mail provides an essential communication media for people all over the world. Many people are overloaded with a large number of emails on a regular basis. Users today want the ability to automatically prioritize and organize their e-mail, and in the future, they would like to do even more automatically, such as addressing mail by organizational function rather than by person.

Intelligent agents can facilitate all these functions by allowing mail handling rules to be specified ahead of time, and letting intelligent agents operate on behalf of the user according to those rules. Usually it is also possible (or at least it will be) to have agents deduce these rules by observing a user's behavior and trying to find patterns in it [Her97].

Messages are exchanged between hosts using the SMTP (Simple Mail Transfer Protocol) with software programs called mail transport agents. Users can download their messages from servers with standard protocols such as the POP (Post Office Protocol). IMAP (Internet Message Access Protocol) provides the user more capabilities for retaining e-mail on the server and for organizing it in folders on the server [Wik07].

2.5.1 Message Format

Internet e-mail messages consist of two major components:

Headers: Message summary, sender, receiver, and other information about the e-mail.

Body: The message itself, usually containing a signature block at the end.

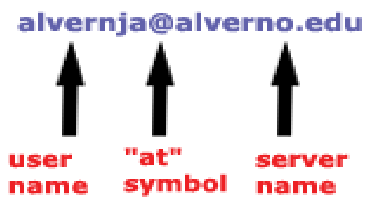
The headers usually have at least four fields [Res01]:

1. **From:** The e-mail address of the sender of the message.
2. **To:** The e-mail address of the receiver of the message.
3. **Subject:** A brief summary of the contents of the message.
4. **Date:** The local time and date when the message was originally sent.

An email address is unique, just like a Post Office street, city, state and zip address. Email addresses have two parts:

- The user name
- The email server or host address

The host address is similar to a post office address. When you send mail to someone in another city, the address on the envelope is read and that piece of mail is directed to a post office for delivery. That is the purpose of the host or email server. The user name is separated from the host address by the @ (pronounced "at") sign. Each user name is unique to a particular host address [Wik07]. For example Jane A. Alverno’s email address at Alverno would be:



The information supplied in the headers on the recipient's computer is similar to that found on top of a conventional letter. The actual information such as who the message was addressed to is removed by the mail server after it assigns it to the correct user's mailbox. Also note that the From field does not have to be the real sender of the e-mail. It is very easy to fake the From line and let an e-mail seem to be from any mail address. It is possible to digitally sign an e-mail. This is much harder to fake.

Other common header fields include:

1. Cc: Carbon copy (because typewriters use carbon paper to make copies of letters). It’s contains the addresses of others who are to receive the message, though the content of the message may not be directed at them.

2. Bcc: Blind carbon copy (the recipient of this copy will know who was in the **To** field, but the recipients cannot see who is on the Bcc: list).
It contains addresses of recipients of the message whose addresses are not to be revealed to other recipients of the message [Res01].

3. Received: Tracking information generated by mail servers that have previously handled a message.

4. Content-Type: Information about how the message has to be displayed, usually a MIME type.

Due to an artifact of the notational conventions, the syntax indicates that, when present, "Date", "From", "Sender", and "Reply-To" fields must be in a particular order. These header items must be unique (occur exactly once). However header fields, in fact, are NOT required to occur in any particular order, except that the message body must occur AFTER the headers. For readability and ease of parsing by simple systems, it is recommended that headers be sent in the order "Date", "From", "Subject", "Sender", "To", "cc", etc. This specification permits multiple occurrences of most optional-fields. However, their interpretation is not specified here, and their use is strongly discouraged [Cro97].

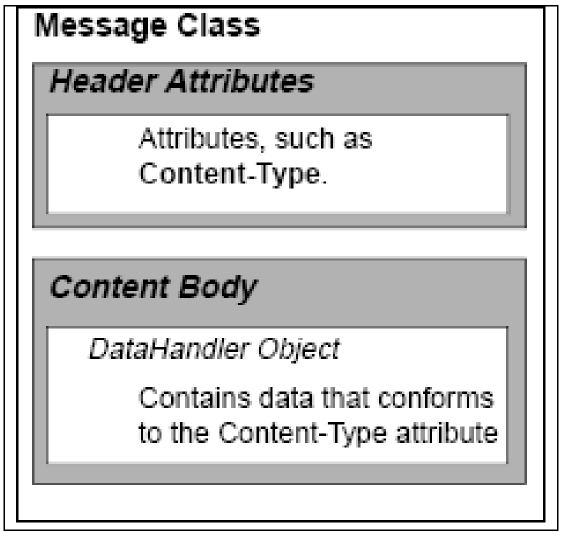


Figure (2.2) Structure of Message Format

2.5.2 Email Filtering

Active users of electronic mail may receive dozens or even hundreds of messages every day. To facilitate retrieval of messages weeks, months or years after their original receipt, most mail reader applications allow users to organize their messages into user-defined folders [Ric99].

The term “mail filtering” is used in different contexts and requires some discussion before the meaning can be cleared. Mail filters, or, sets of rules that users put together to file incoming e-mail into different mailboxes or folders. It is call personal mail filtering because it pertains directly to a single person’s organizational preferences. As e-mail use has grown, some regularity has come about the sort of e-mail that appears in users’ mail boxes. In particular, un-solicited e-mail, such as “make money fast” schemes, chain letters and porn advertisements, is becoming all too common. Filtering out such unwanted trash is known as junk mail filtering [Ren00].

There are three types of filtering [Pay94]:

- ***Cognitive Filtering:*** this characterizes a message by the contents and meaning of the message. Participants in the survey looked for certain keywords or phrases to classify messages.
- ***Social Filtering:*** this complements the cognitive approach by concentrating on the personal and organizational interrelationships between sender and receiver. For example, more attention may be given to messages from a superior such as a supervisor.
- ***Economic Filtering:*** this is based on a cost-benefit assessment of a message, such as the length of a message.

Instead of simply offering tools that are manipulated by a human user, an email agent attempts to organize email messages automatically based on its knowledge about each individual user. In particular, the agent should be able to classify incoming email messages into folders and to prioritize them so that the user can focus on more important emails first [Den98].

2.6 Learning System

Learning means that the agent is capable of using its past experience in order to improve its future behavior. Ways to implement this property may vary from simple repetitive observation of a certain behavior in the environment to heavyweight tools such as neural networks [Jen98].

The idea behind learning is that percepts should be used not only for acting, but also improving the agent's ability to act in the future. Learning takes place as a result of the interaction between the agent and the world, and from observation by the agent of its own decision-making processes

One central element of intelligent behavior is the ability to adapt or learn from experience. Adding learning or adaptive behavior to an intelligent agent elevates it to a higher level of ability. A learning agent can adapt to your likes and dislikes. It can learn which agents to trust and cooperate with, and which ones to avoid. A learning agent can recognize situations it has been in before and improve its performance based on prior experience [Big01].

The following strategies can be used by a learning agent [Pfe01]:

- Rote learning: direct implementation of knowledge and skills without requiring further interface or transformation from the learner.
- Learning from instruction and by advice taking: operationalization transformation an internal representation and information like an instruction or an advice that is not directly executable by learner.
- Learning from examples and by practice: extraction and refinement of knowledge and skills a general concept or a standardized pattern of motion from positive and negative examples or from practical experience.
- Learning by discovery: gathering new knowledge and skills by making observations, conducting experiments, and generating and testing hypotheses or theories on the basis of the observational and experimental results.

2.6.1 Learning Paradigm

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning [Kur05].

2.6.1.1 Supervised Learning

Supervised learning the most common form of learning and is sometimes called programming by example. The learned agent is trained by showing it examples of the problem state or attributes along with the desired output.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation).

2.6.1.2 Unsupervised Learning

Unsupervised learning is used when the learning agent needs to recognize similarities between inputs or to identify features in the input data. The data is presented to the agent, and it adapts so that it partitions the data into groups.

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

In unsupervised clustering, unlabelled collection of documents is available. The aim is to cluster the documents without additional knowledge or intervention such that documents within a cluster are more similar than documents between clusters. Clustering techniques can be categorized into two major groups as partitional and hierarchical [Ozg04].

There are two types of unsupervised clustering:

1- Partitional Clustering Techniques.

2- Hierarchical Clustering Techniques

2.6.1.3 Reinforcement learning

Reinforcement learning is a type of supervised learning where the error information is less specific. It can also be used in cases where is a sequence of inputs and the output or action is only taken after the specific sequence occurs.

Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

Another important distinction in learning agents is whether the learning is done on-line or off-line. **On-line** learning means that the agent is sent out to perform its tasks and that it can learn or adapt after each transaction is processed. On-line learning is like on-the-job training and places severe requirements on the learning algorithms. It must be very fast and very stable. **Off-line** learning, on the other hand, is more like a business seminar. You take your salespeople off the floor and place them in an environment where they can focus on improving their skills without distractions. After a suitable training period, they are sent out to apply their newfound knowledge and skills. In an intelligent agent context, this means that the data will be gathered data from situations that the agents have experienced. This data will be augmented with information about the desired agent response to build a training data set. Once this database is obtained, it could be used it to modify the behavior of the agents [Big01].

Unsupervised machine learning for user modeling has been mostly used in non-educational applications. For example, collaborative filtering (CF) systems employ unsupervised learning techniques to model user preferences and make item recommendations based on user similarities. Other research has demonstrated the use of unsupervised learning on words in a document to model and automatically manage email activities [Ame07].

2.7 Design of Email Filtering agent

As information requirements vary greatly from user to user, the filtering system should be highly personalized to satisfy the users' needs. This model may be used over a long period of time, yet it cannot be assumed that the requirements of the user will remain static, so the agent must be able to notice when these requirements change and revise its model accordingly [Pay94].

2.7.1 System Requirements for email agent

The requirement is to develop a mail agent which aids its user in handling mail. The agent sits above the mail tool observing the users interactions with the tool. It can interact with the mail tool in order to perform actions automatically for the user.

The following is a list of the types of actions the agent will attempt to learn from observing the user:

• Messages which are filed away in different mailboxes for later browsing.
• Junk mail which the user is never interested in reading.
• Messages which are forwarded to other users.

The agent comprises of a number of modules, each of which are responsible for certain tasks. Each time the rule base is generated, the existing rules are discarded and new ones are created. This process can be processor intensive, so it occurs as a regular batch process. Classification of new mail messages is performed each time a new message is delivered; again to reduce processing time when the mail tool is invoked.

The agent has to be as transparent as possible. Mail agents such as that which communicates with the mail tool rely on the user interacting with the agent in addition to the mail tool. The only time the user is aware of the agent is when the user requests help. Because of this the agent has to interact with an existing mail tool [Sun98].

2.7.2Clustering algorithm

The objective of the project is to build a module for an office agent that clusters emails into folders. The primary question that arises is whether an agent can do the above since such a clustering (as any other clustering operation) requires a metric of similarity/dissimilarity as the centric operator. However, there is no gold standard available in such a scenario. In fact, each user decides his own gold standard; in other words a user himself decides as to what he wants to be considered as similar/dissimilar. For example, someone might want his emails to be foldered based on the month of receipt whereas someone else would want the clustering to be done based on subject and content. List emails provide another such example.

As input the agent shall receive emails of its client, along with a set of examples, both similar and dissimilar, in order to develop a sense of what the user requires in his clustering. The agent has the task to group the emails into folders based on the similarity between them in the form of common subjects, common email threads etc. The goal is to design a learning strategy for the agent to learn how to cluster [Sah03].

Chapter Three
Design and Implementation of EMFA

3.1 Introduction

This chapter will describe the system of email filtering agent which utilize clustering algorithm that classify the messages into two lists negative and positive according to the features that extracted from each message. Then, the filtering algorithm will be applied on the two lists. The messages in the negative list will be deleted and the messages in the positive list will be replied to the same sender or forwarded to a new one.

3.2 Email Filtering Agent (EMFA) System

Email Filtering Agent (EMFA) modules deal with the messages in the Inbox that need to be filtered, so the first step in the system is to fetch the messages from Inbox, then extract the features from these messages, and then apply clustering algorithm that will divide the messages into two lists *negative* and *positive* lists. Negative list contains unwanted messages while Positive list contains wanted messages. After that, the filtering algorithm will be applied on the two lists. The messages in negative list will be deleted and the messages in the positive list will be either replied or forwarded according to the features that extracted .The architecture of proposed Email Filtering Agent (EMFA) is shown in Figure (3.1).

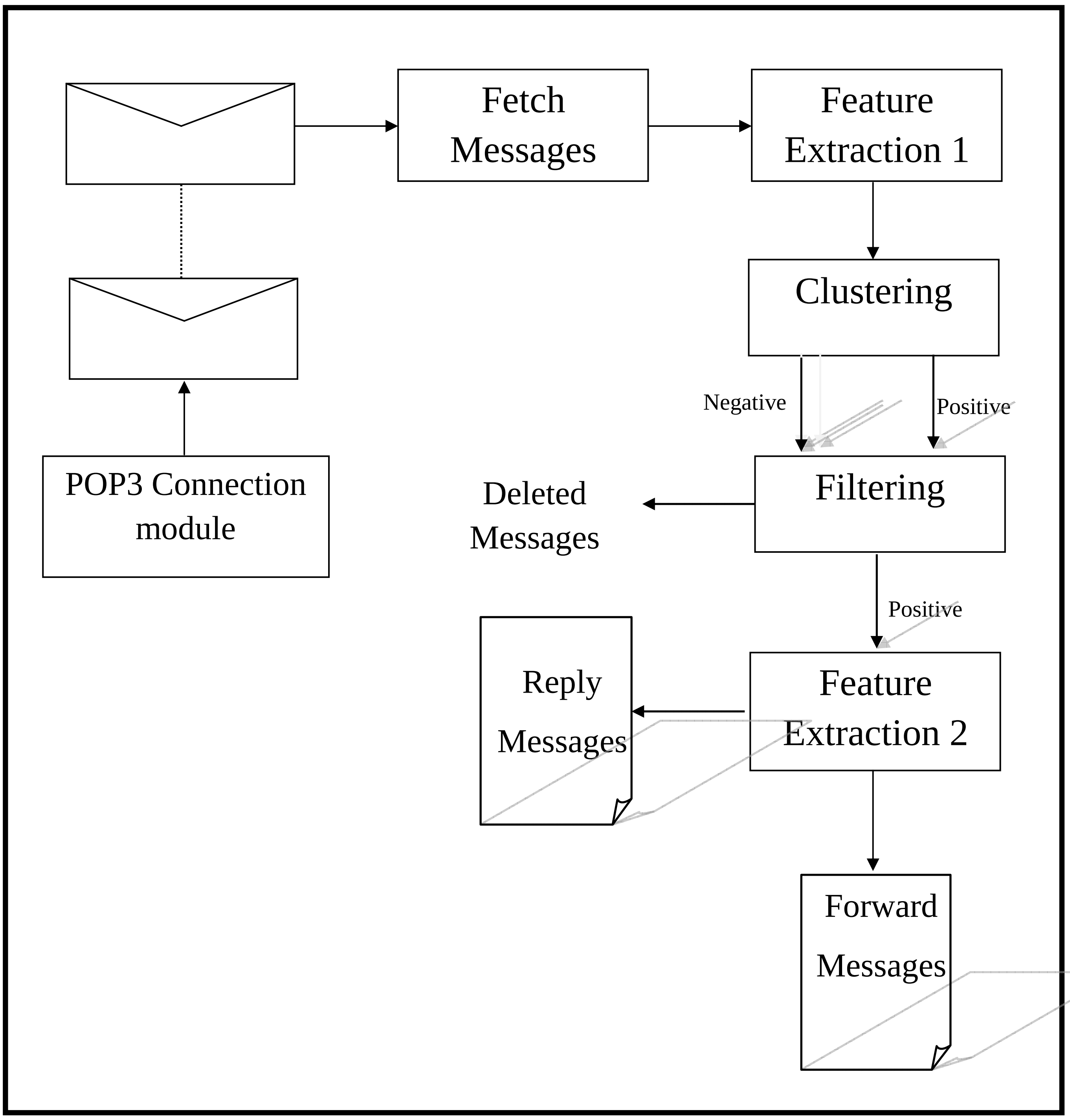


Figure (3.1) EMFA system model

3.3 Connection to POP3

The design of POP3 and its procedures supports end-users with the disconnected state (such as dial-up connections), allowing those users to retrieve e-mail when they are connected and then view and manipulate the retrieved messages without the need to stay connected. To fulfill the connection with the POP3 server multi functions are used such as **smtpmailer 1.4**, **mail-1.4** and **activation-1.1**.Although most clients have an option to leave mail on server, e-mail clients using POP3 generally connect, retrieve all messages, store them on the user's PC as new messages and then disconnect. In contrast, the newer, more capable Internet Message Access Protocol (IMAP) supports both connected (online) and disconnected (offline) modes of operation.

When connected, the interface between a mail-client and the network must be defined, this is called ***Session***. The mail ***Session*** object manages the configuration options and user authentication information that are used to interact with messaging systems. There are simultaneous multiple sessions. Each session can access multiple message stores and transports. Any desktop application that needs to access the current primary message store can share the default session. Typically the mail-enabled application establishes the default session, which initializes the authentication information necessary to access the user’s Inbox folder. Other desktop applications then use the default session when sending or accessing mail on behalf of the user. When sharing the session object, all applications share authentication information, properties, and the rest of the state of the object.

After defining session, the database that holds the messages in the Inbox must be defined, this is called ***Store***. The Store also defines the access protocol used to access folders and retrieve messages from folder. After defining Store, the connection with the POP3 will be fulfilled. Figure (3.2) shows how to connect with POP3 by entering the host, username and password and save them as **Properties.** The properties will then be passed to the ***Session*** that opened with the network and then POP3 will be used to define ***Store*** that takes the information from the session. At the end, when all the information is available, the connection is fulfilled with the POP3 to open the Inbox and retrieve messages from it.

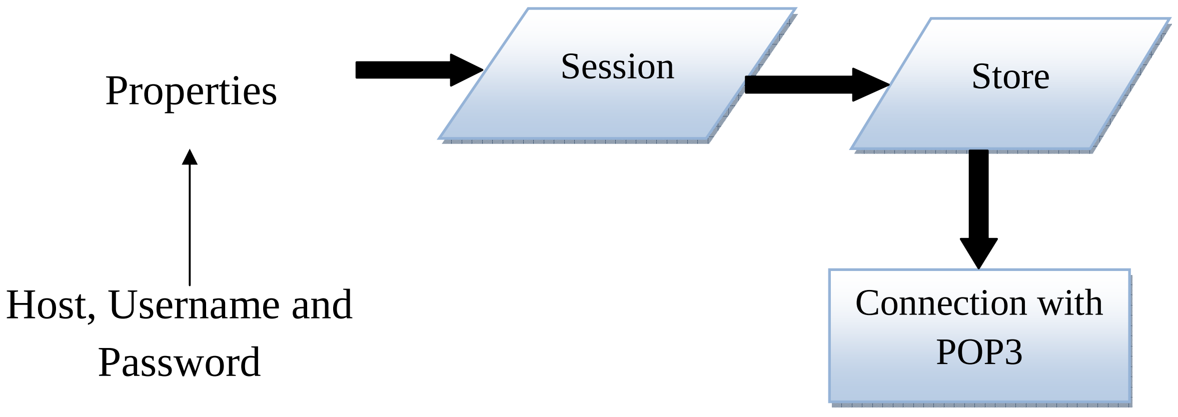


Figure (3.2) connection with POP3

3.3.1 Open Session

The Session class provides access to the protocol providers that implement the Store, Transport, and related classes. The Session class allows messaging system implementations to use the Authenticator object that was registered when the session was created. The Authenticator object is created by the application and allows interaction with the user to obtain a user name and password. The user name and password is returned in a **PasswordAuthentication** object. The messaging system implementation can ask the session to associate a user name and password with a particular message store using the **setPasswordAuthentication** function.

There are two ways to create a Session in Java language as shown below:

- **Session session = Session.getInstance (props, authenticator)**
- **Session defaultSession = Session.getDefaultInstance(props, authenticator)**

The Properties object that initializes the Session contains default values and other configuration information. It is expected that clients using the values for the mail.host, mail.username, and mail.password. Algorithm (3.1) shows the steps to open session.

Algorithm (3.1) open session

```
Input: username and password
Output: flag is true
Procedure
Step1: Set Username and Password.
Step2: Define props as properties.
Step3: Set authentication on username and password.
Step4: Pass properties and authentication as parameters of session.
Step5: Set flag of session as true to open session.
```

3.3.2 Store class

The Store class models the message database and its access protocol. A client uses it to connect to a particular message store, and to retrieve messages in Inbox. Clients gain access to a Message Store by obtaining a Store object that implements the database access protocol. Most message stores require the user to be authenticated before they allow access; the connection method performs that authentication.

For many message stores, a host name, user name, and password are sufficient to authenticate a user. Store provides a default connect method. In either case, the client can obtain missing information from the Session object's properties, or by interacting with the user by accessing the Session's Authenticator object. Typically, the client retrieves the default Store or Transport object based on properties loaded for that session as shown below:

- **Store store = session.getStore( )**

The client uses the Session object's getStore method to connect to the default store. The getStore method returns a Store object subclass that supports the access protocol defined in the user properties object, which will typically contain per-user preferences as shown in appendix (A). After that, the **Inbox** is opened to read the messages from it. Figure (3.3) shows the message-handling process, and algorithm (3.2) illustrates how to use store class to connect with POP3 and read messages.

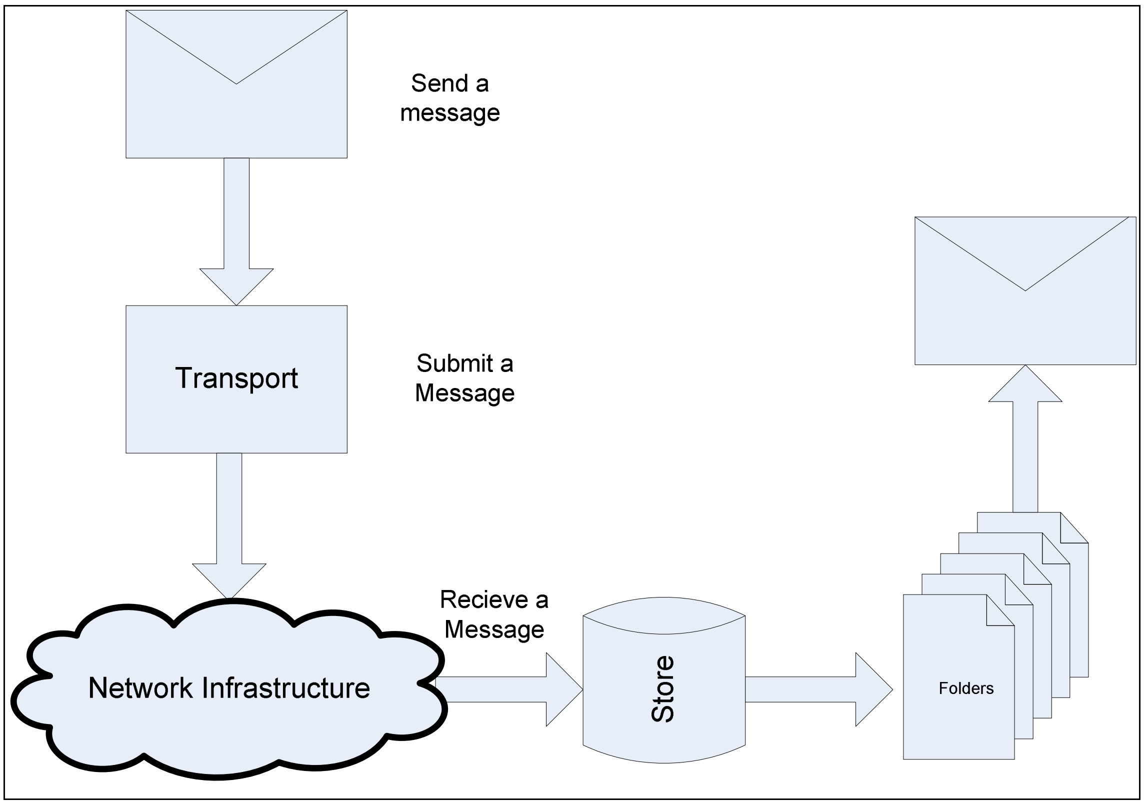


Figure 3.3 message-handling processes

Algorithm (3.2) store class

```
Input: properties and POP3
Output: create Inbox
Procedure
Step1: Define pstore and folder as new variables.
Step2: Pass properties.
Step3: Open session with POP3 server as a parameter.
Step4: Open connection to connect the store with the server.
Step5: Open Store as read_only mode and save messages in Inbox.
Step6: Read messages from Store.
Step7: Close folder after finished from it.
Step8: Close the connection
```

3.4 Fetch messages

The **Folder** class represents a folder containing messages. Folders can contain subfolders as well as messages, thus providing a hierarchical structure. Each user has a folder that has the case-insensitive name INBOX. The Inbox can be viewed as presenting a resizable array of messages to a client. This allows the client to access a message based on its index within this array. The index is the message's sequence-number. Sequence numbers begin at one (1) and continue incrementing by one, through the total number of messages in the folder.

The **Store** class provides abstract functions for allowing the user to retrieve a folder:

- **getDefaultFolder**
- **getFolder**

The ***getDefaultFolder*** function for the corresponding Store object returns the root folder of a user's default folder hierarchy. This is the default initial state of a folder. The ***getFolder*** function returns the specified folders **Folder getFolder (String name).** The folder class provides a specific function to get one or more messages from Inbox, this function is ***getMessage***. The ***getMessage*** function, when given no parameters, returns all of the message objects in the folder. The ***getMessage*** function takes the number of messages in the **Inbox** that is known by using ***getMessageCount*** function. Algorithm (3.3) shows how to open the Inbox and how to fetch messages from it.

Algorithm (3.3) Fetch message

```
Input: Store
Output: Messages
Procedure
  Step1: Define a variable such as x to compute the number of
         messages in the Inbox.
  Step2: Open store to get Inbox.
  Step3: Open Inbox for read_write mode.
  Step4: Save number of messages in x parameter.
  Step5: Define a counter to facilitate fetching messages.
  Step6: Fetch the first messages.
  Step7: Check if the counter is larger than number of messages, If No
         then increment the counter, If Yes then close the Inbox.
```

3.5 Feature Extraction

The agent should be able to classify the incoming email message into folders according to the features that were extracted from the message headers. The header contains information about who was the sender, time and date of sending, subject of the message, recipients, etc. The functions ***getFrom***, ***getSubject*** and ***getDate*** are used to extract the features: **From** field, **Subject** field and **Date** field from the headers**.** Figure (3.2) illustrates the message hierarchy and the functions that deal with it and algorithm (3.4) shows the steps that are followed to extract the features from messages.

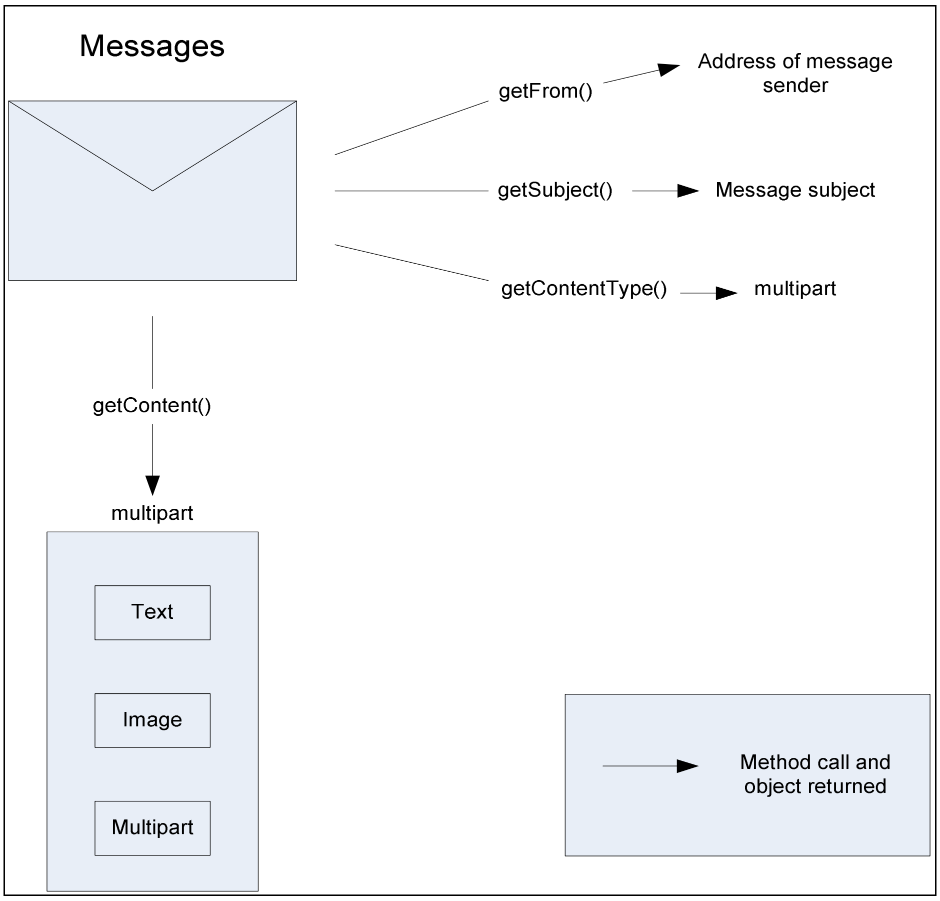


Figure (3.2) message hierarchy

```
Algorithm (3.4) Feature extraction

Input: Message
Output: Date, Subject and From
Procedure
Step1: Define three arrays for the features that extracted from messages.
Step2: Open Inbox for read _write folder.
Step3: Save the number of messages in a counter.
Step4: Fetch a message from Inbox.
Step5: Extract the Subject field, From field and Date field from the
       header of the message and save them in the arrays.
Step6: Increment the counter.
Step7: Check the counter if it is less than the number of messages in the
       Inbox, If No then fetch another message, If Yes then close the
       Inbox.
```

3.6 Clustering algorithm

Clustering is the algorithm of organizing objects into groups whose members are similar in some way. The designing of clustering algorithm is to choose how to classify emails. It represents each email using both header features that include the subject line, email addresses, and date of sending message.

A clustering algorithm creates a set of rules for each class in a concept. Each rule covers many examples of the class in question (***positive*** examples), yet selects examples of other classes (***negative*** examples). According to these examples, the messages will be filled into two lists, **Negative** and **Positive** lists. The **Negative** list contains messages that have the date which is specified by the user under specific condition and the **Positive** list contains messages that will be filtered by using filtering algorithm after extracting other features from it. Figure (3.3) shows how to use the features in clustering algorithm and algorithm (3.5) illustrates the steps of clustering algorithm.

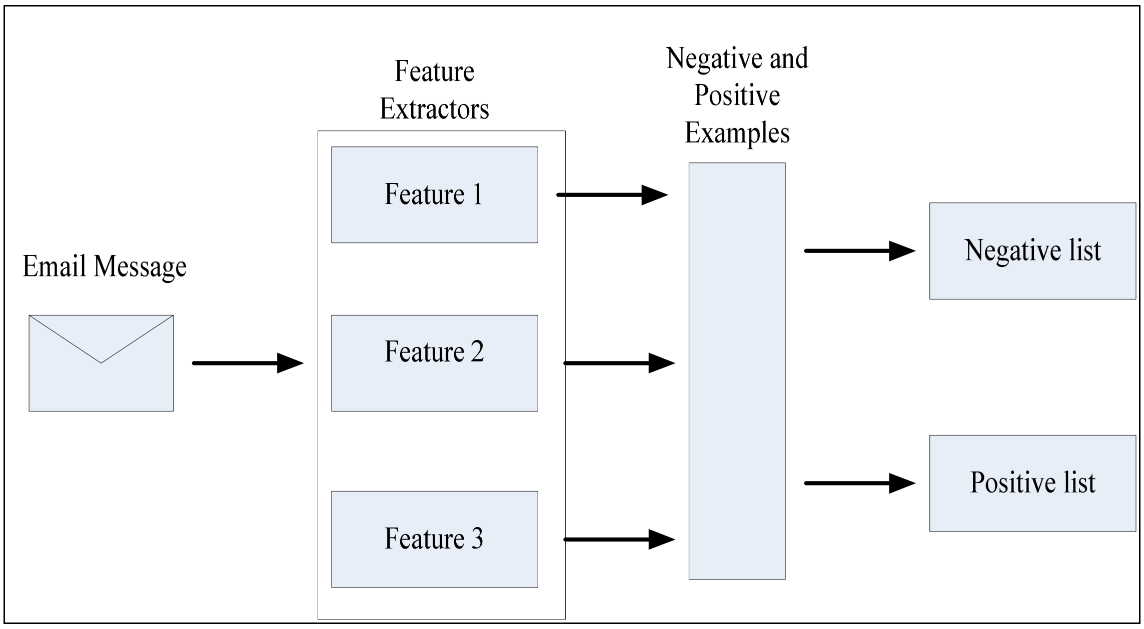


Figure (3.3) clustering algorithm

Algorithm (3.5) Clustering

Input: *Message Date, Subject and From*

Output: *negative and positive lists*

Procedure

```
Step1: Define two arrays: positive and negative.
Step2: Fetch the Date field from Date arrays.
Step3: Split the day, month and year from the Date field.
Step4: Check the month with a specific value that the user was entered.
Step5: If the year is less than the predefine value then consider the message as
        negative and save it in the negative array.
Step6: If the month is larger than the value then consider the message as
        positive and save it in the positive array.
Step7: Increment the counter.
Step8: Check the counter with the length of Date array, if it is less than
        the number of messages in the array, If No then fetch another
        date, If Yes then close the array.
```

3.7 Filtering Algorithm

After clustering, filtering algorithm will be applied on ***negative*** and ***positive*** lists. In this project the social filtering is used. This type of filtering depends on the interrelationships between sender and receiver. The negative list contains the unwanted messages that were classified depending on the date of the message, these messages will be deleted. While in the positive list, the messages will be either replied or forwarded according to the **From** field and **Subject** field. The features that are extracted from the messages in filtering algorithm are the same features that were extracted in clustering algorithm but there is no need to extract the **date** field here. The performing of reply and forward is decided when the **From** field is extracted. According to **From** field the agent will decide which messages will be forwarded and which will be replied. For the replied messages the **Subject** field will decide which form of the two predefined user forms should be used for the body. Figure (3.4) illustrate the filtering algorithm diagram.

3.7.1 Forward Message

Email forwarding involves passing email from one address to another. The address of the message in the positive list will be compared with the addresses in the **user defined forward list** which is set by the user. **User defined forward list** contains the specific email addresses that are specify if the message will be forwarded or not. If the email address is matches with one of these addresses, the message will be forward to all addresses in the address book. The counter of the number of forwarded messages will be increased with every sent message. Algorithm (3.6) shows the steps of forward message.

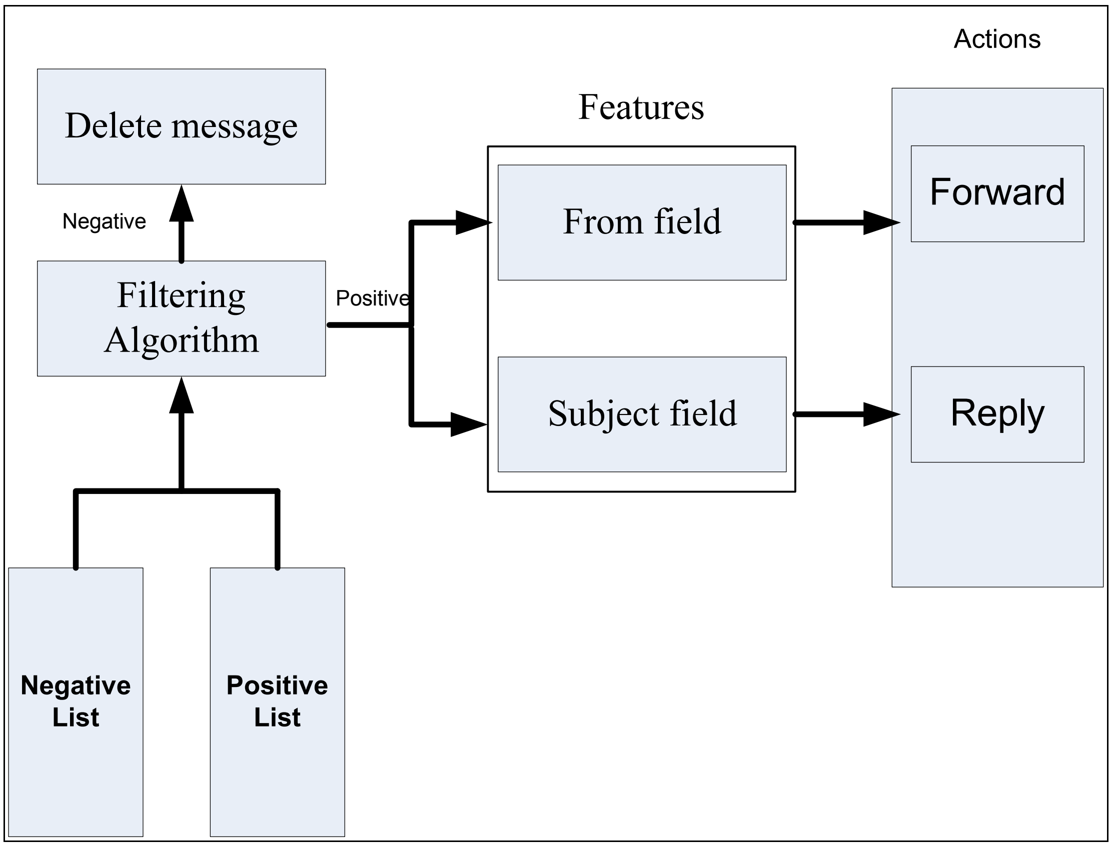


Figure (3.4) Filtering algorithm

Algorithm (3.6) Forward Message

Input: *Positive list*

Output: *Forward Messages*

Procedure

```
Step1: Define an addresslist1 (user defined forward list) that has favorite
       email addresses.
Step2: Open positive list.
```

Continue

```
Step3: For each message in positive list, take the From field from them.
Step4: Check if the email address is match with one of the addresses in
       the address boo, if not match then send a warning message.
Step5: Check the counter of messages in positive list, if it is less than
       the number of messages in the list, If No then fetch another
       message, If Yes then close the array.
```

3.7.2 Reply Message

In the reply message, the agent will compare the address of the message in the positive list with a **user defined reply list**. The **user defined reply list** contains the sender email addresses of the messages that the agent must send a reply to them. There are two standard forms defined by the user, these forms will be used for the message body and will be answered to the senders. The agent will depend on the **Subject** field for these messages to choose one of the two forms, and then a message will be sent to the sender using the same email address. The counter of the number of replied messages will be incremented by 1 each time a message is sent. Algorithm (3.7) shows the above steps.

Algorithm (3.7) Reply Message

```
Input: Positives list
Output : replied Messages
Procedure
Step1: Open an addresslist2 that has an important email address.
```

Continue

```
Step2: Open positive array.
Step3: For each message in the positive list, get the From field and
       Subject field.
Step4: Check if the email address is match with one of the addresses in
       the addresslist2, then fetch the from that must be write in the
       message body.
Step5: Take the subject to write in the Subject field.
Step6: Send the message.
Step7: Check the counter of messages in positive list, if it is less than
       the number of messages in the list, If  No then fetch another
       message, If Yes then close the array.
```

3.7 Delete Message

Deleting messages from Inbox is a two-phase operation. The first phase is setting the **DELETED** flag for messages to mark them as deleted messages, but they will not be removed from the list. The **setFlags** function is responsible for setting this flag. Flags objects carry flag settings that describe the state of a message object within the list. The second phase is deleting the messages by using the **expunge** function. When the expunge function returns, the sequence number of those messages will be renumbered. Algorithm (3.8) shows the deletion steps.

Algorithm (3.7) Deleting Message

Input: *Negative list*

Output : *Deleted Messages*

Procedure

```
Step1: Open negative list.
Step2: Define a counter to the number of messages in the negative array.
Step3: Set flag of each message to True.
Step4: Delete messages.
Step5: Refresh the Inbox to renumber the messages in it.
Step6: Check the counter with the number of messages in the negative
         list, if it is less than the number of messages in the list, If No then
       fetch another message, If Yes then close the array.
```

Chapter Four
Email Filtering Interface and Result

4.1 Introduction

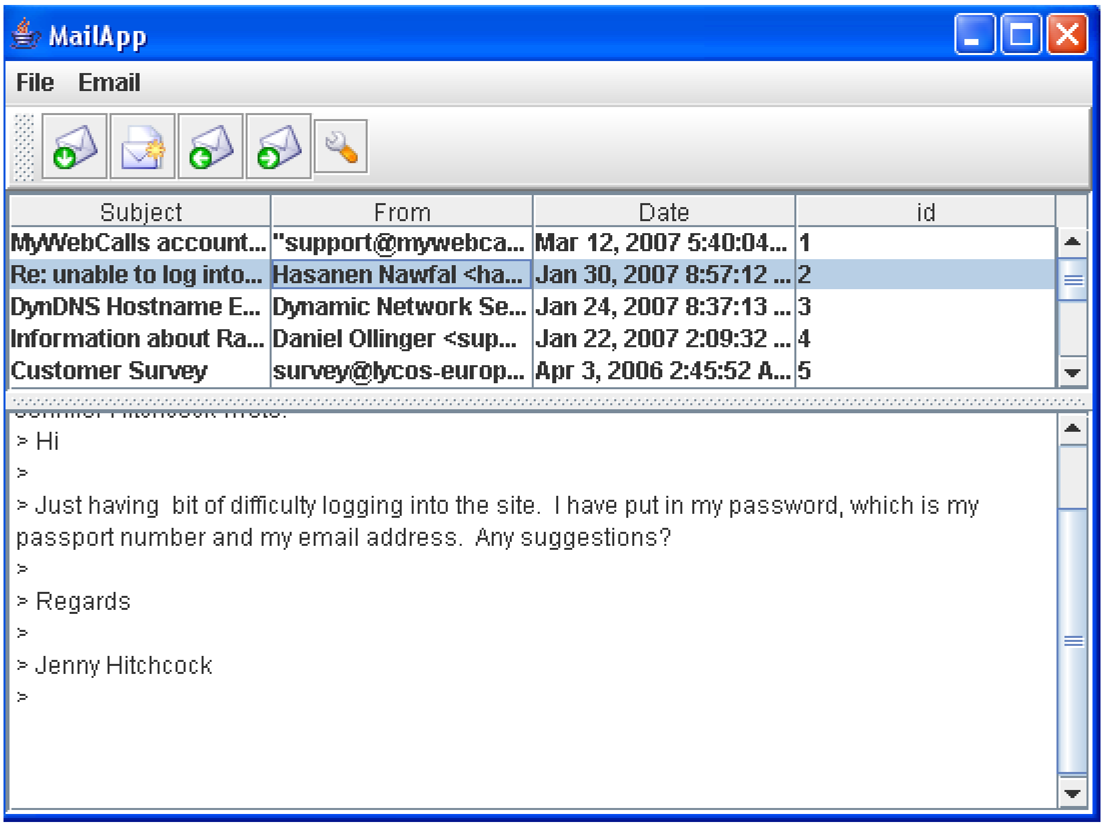
The EMFA system described in chapter three was implemented here using **JAVA (version 1.4) and Eclipse (SDK 3.2.2)** running under windows XP operating system. The experiments are performed on Pentium 4, 1.6 GHz with 256 MB of RAM.

The algorithms described in chapter three for reading messages, feature extraction, clustering, and filtering algorithm are applied on a set of messages that are stored in the Inbox and the obtained results are illustrated in the following sections.

4.2 EMFA system

EMFA system is designed with many frames to enable the agent to filter the messages stored in the Inbox. The first frame that appears on the screen is the main frame that contains the inbox and the messages header. The header of the messages is divided into four fields: **Subject**, **From**, **Date** and **ID** (that is a unique number associated with each message)**.**

When any message is selected, the body of the message appears in the same frame as shown in figure (4.1).



Subject	From	Date	id
MyWebCalls account...	"support@mywebca...	Mar 12, 2007 5:40:04...	1
Re: unable to log into...	Hasanen Nawfal <ha...	Jan 30, 2007 8:57:12 ...	2
DynDNS Hostname E...	Dynamic Network Se...	Jan 24, 2007 8:37:13 ...	3
Information about Ra...	Daniel Ollinger <sup...	Jan 22, 2007 2:09:32 ...	4
Customer Survey	survey@lycos-europ...	Apr 3, 2006 2:45:52 A...	5

```
> Hi
>
> Just having  bit of difficulty logging into the site.  I have put in my password, which is my
passport number and my email address.  Any suggestions?
>
> Regards
>
> Jenny Hitchcock
>
```

Figure (4.1) EMFA system

4.3 EMFA frame

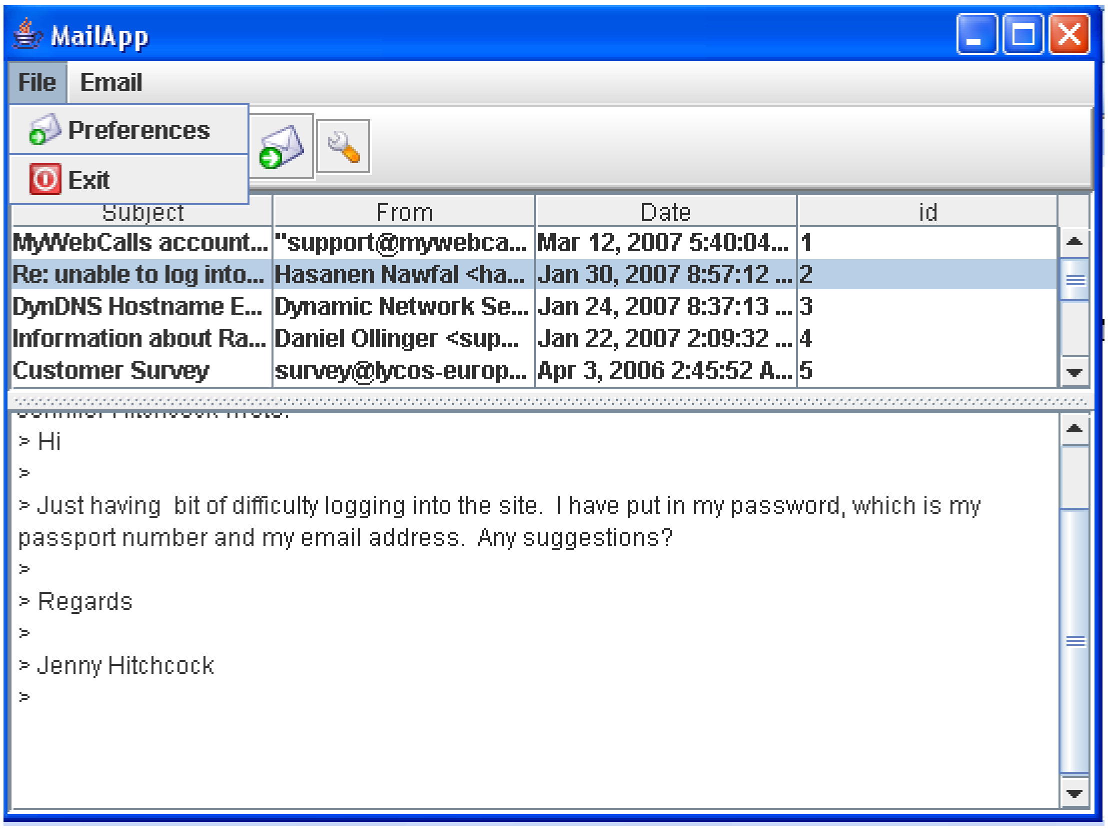
The EMFA frame contains a menu of two options **File** and **Email**, and those options are illustrated below:

1-**File** option: This option contains two items, **Preferences** and **Exit** as shown in figure (4.2).

- **Preferences**: It contains three fields, **Account Settings**, **POP3 Server Settings** and **Outgoing mail Setting**. These fields should be filled correctly so that the user can have access to his account.
- The **Account Settings** field should be filled with the name and the Email address of the user. Then the username and password must be entered in the **POP3 Server Settings** fields. If the server requires

authentication, the username and password of server must be entered in the **Outgoing Mail Settings** as shown in figure (4.3).

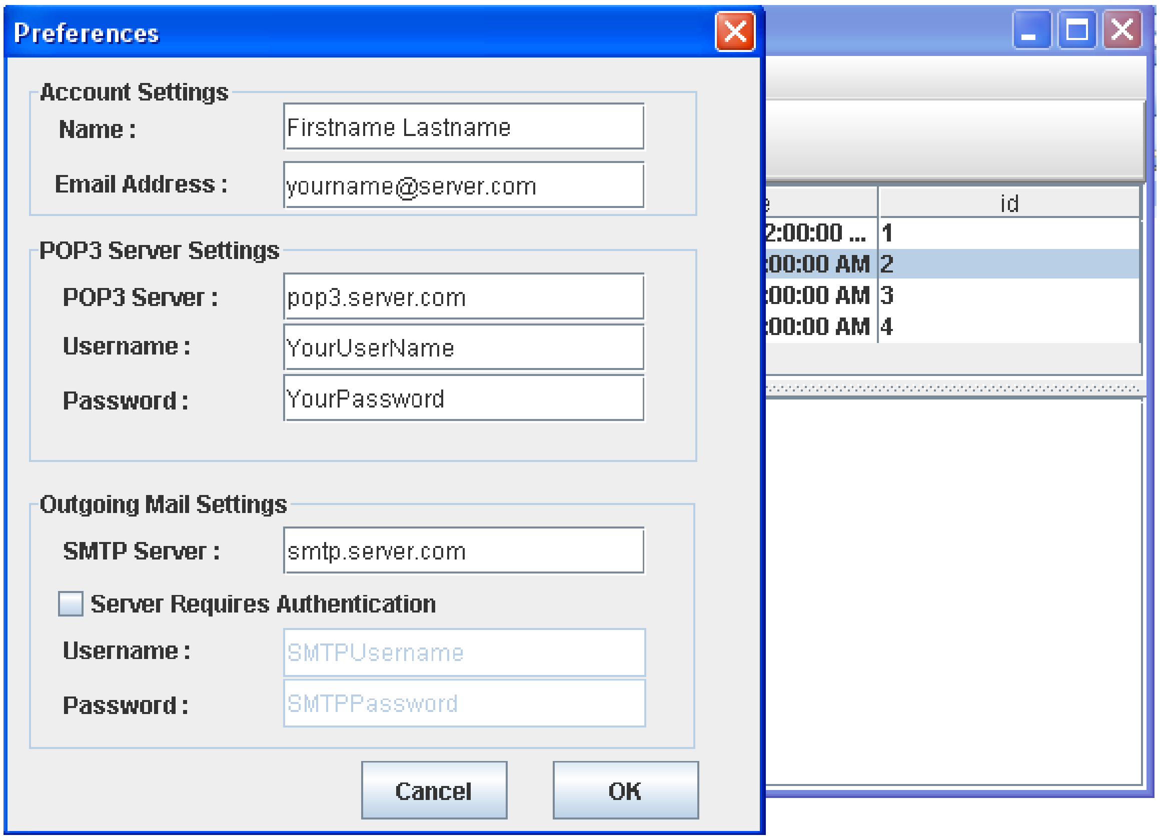
- **Exit**: this button ends the execution of the monitoring system.



Subject	From	Date	id
MyWebCalls account...	"support@mywebca...	Mar 12, 2007 5:40:04...	1
Re: unable to log into...	Hasanen Nawfal <ha...	Jan 30, 2007 8:57:12 ...	2
DynDNS Hostname E...	Dynamic Network Se...	Jan 24, 2007 8:37:13 ...	3
Information about Ra...	Daniel Ollinger <sup...	Jan 22, 2007 2:09:32 ...	4
Customer Survey	survey@lycos-europ...	Apr 3, 2006 2:45:52 A...	5

```
> Hi
>
> Just having  bit of difficulty logging into the site.  I have put in my password, which is my
passport number and my email address.  Any suggestions?
>
> Regards
>
> Jenny Hitchcock
>
```

Figure (4.2) EMFA frame



Preferences

Account Settings

Name :	Firstname Lastname
Email Address :	yourname@server.com

POP3 Server Settings

POP3 Server :	pop3.server.com
Username :	YourUserName
Password :	YourPassword

Outgoing Mail Settings

SMTP Server :	smtp.server.com

☐ Server Requires Authentication

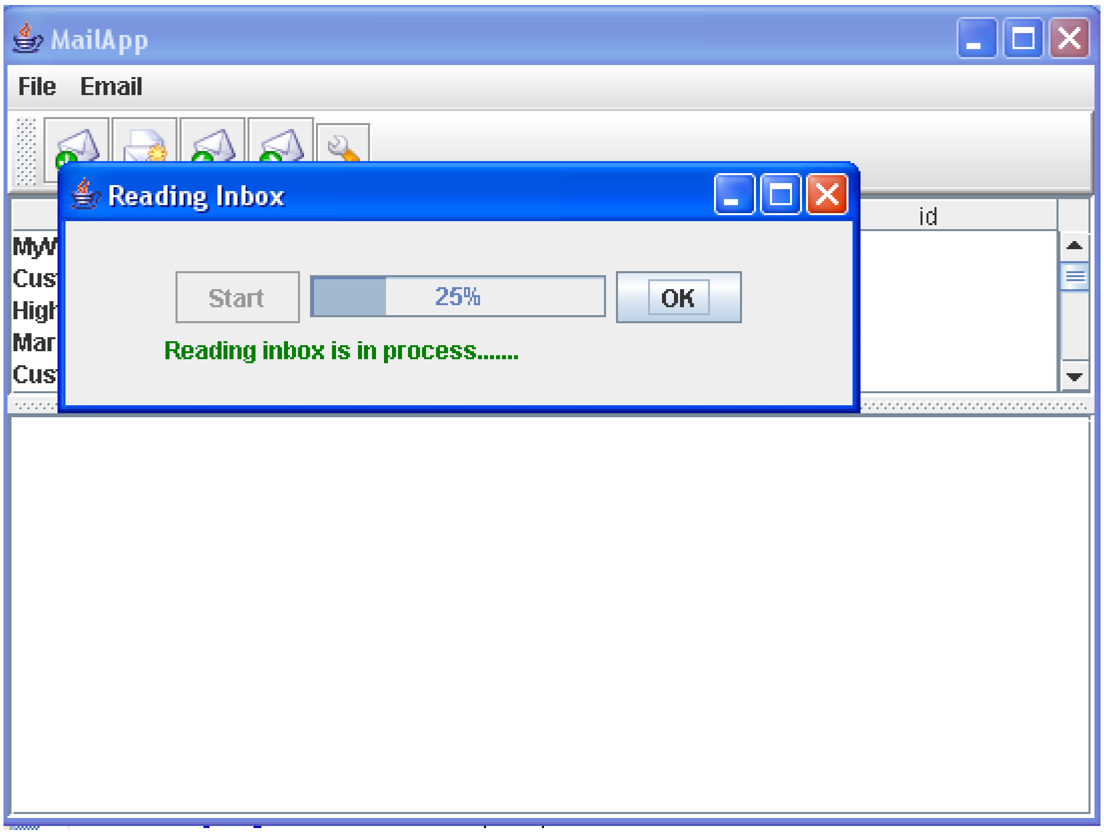
Username :	SMTPUsername
Password :	SMTPPassword

Figure (4.3) Preference frame

2- **Email** option: this option contains many items, Compose, Reply, Forward, Delete and Agent.

4.4 Running Agent

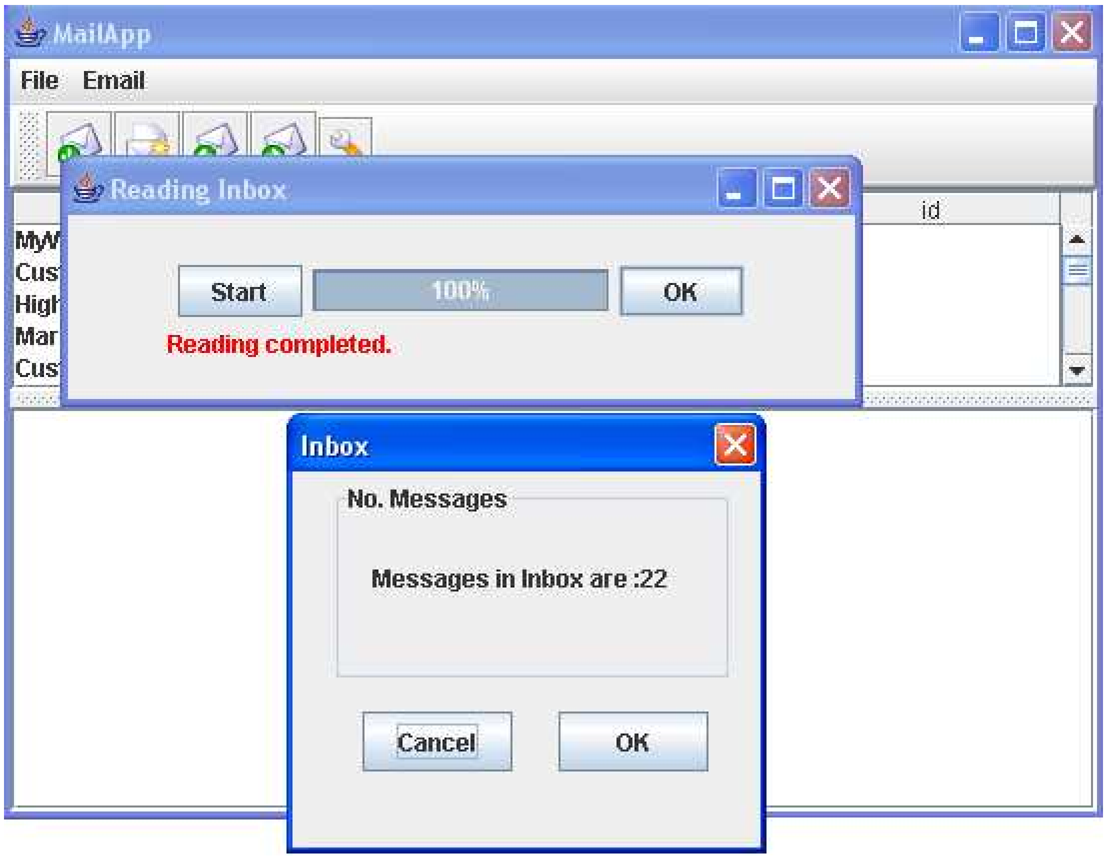
When an agent button is clicked, then the program will be run automatically. A reading Inbox frame will appear that contains the progress bar, Start and OK buttons. If start button is pressed, then progress bar will indicate the progress of reading the messages in the inbox one by one as shown in figure (4.4).



Reading inbox is in process.......

Figure (4.4) Reading Inbox frame

When the reading inbox process is done, a new report window will appear reporting the number of messages in the inbox, as shown in figure (4.5).

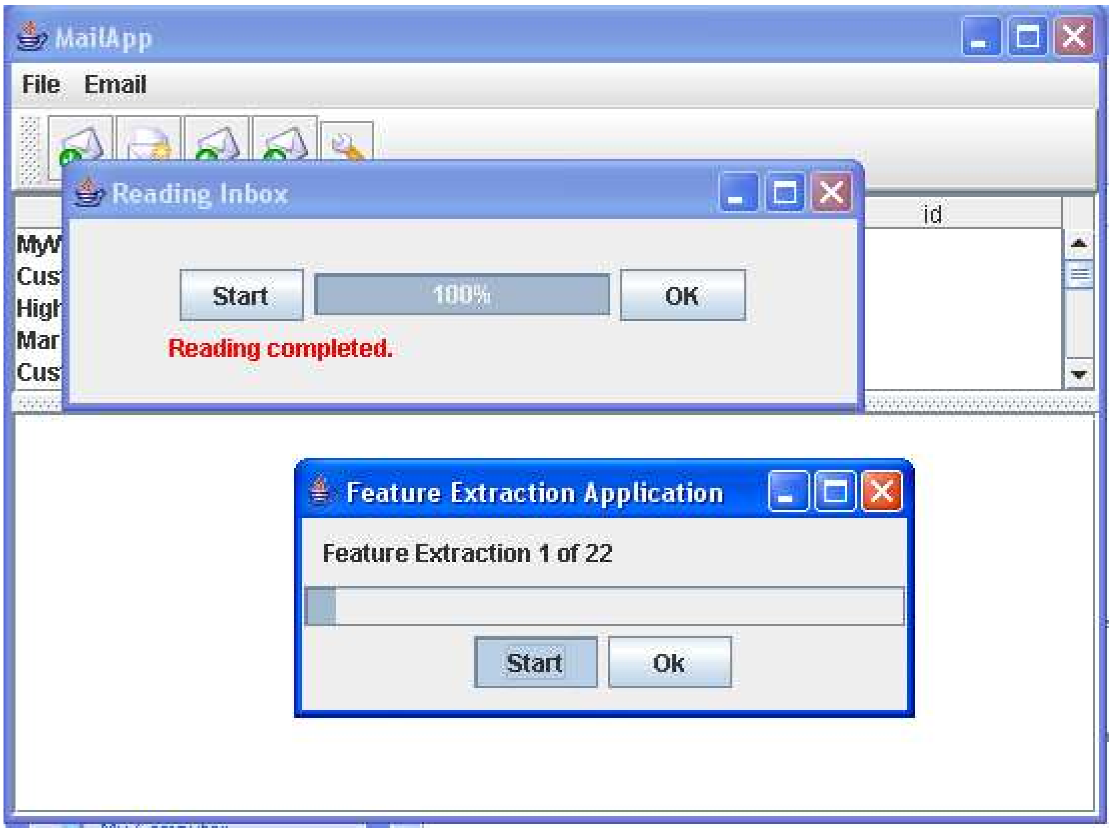


Reading completed.

Messages in Inbox are :22

Figure (4.5) Reading Inbox report

After that, the agent will apply clustering algorithm on all of the messages as shown in figure (4.6). When the progress bar starts, the agent will extract the features from messages such as **Subject**, **From** and **Date**.

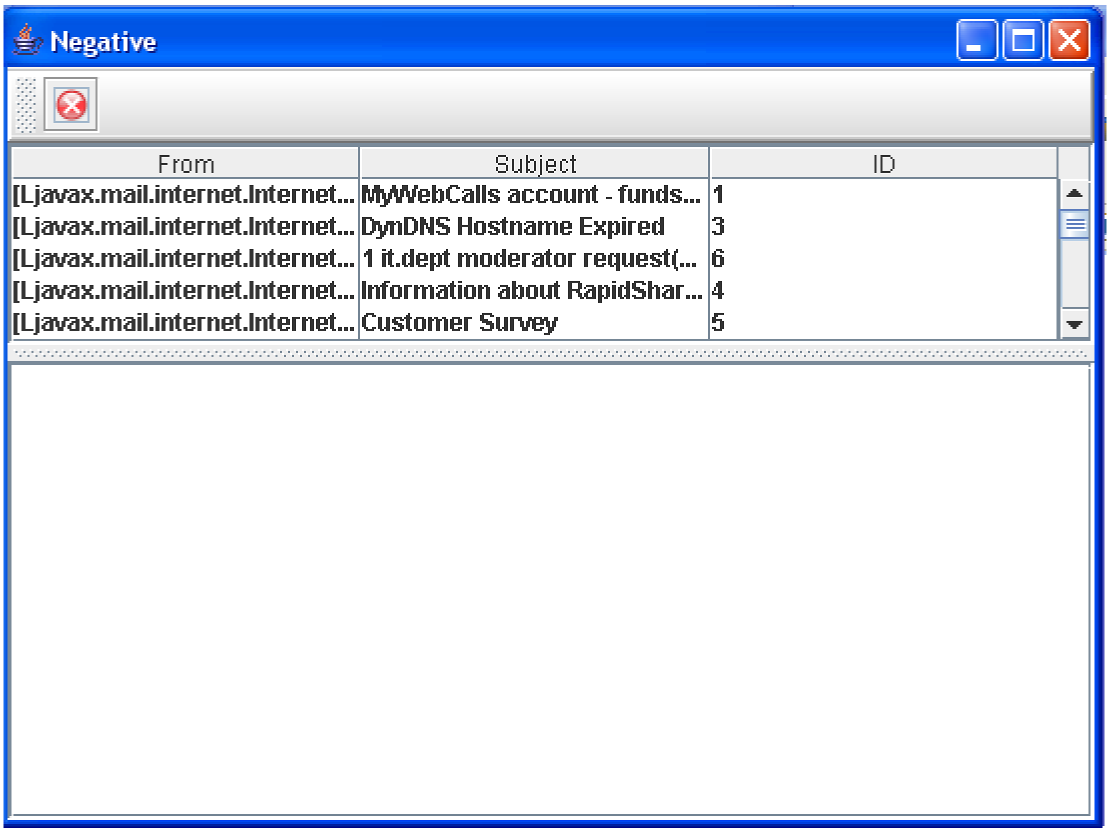


Reading completed.

Feature Extraction 1 of 22

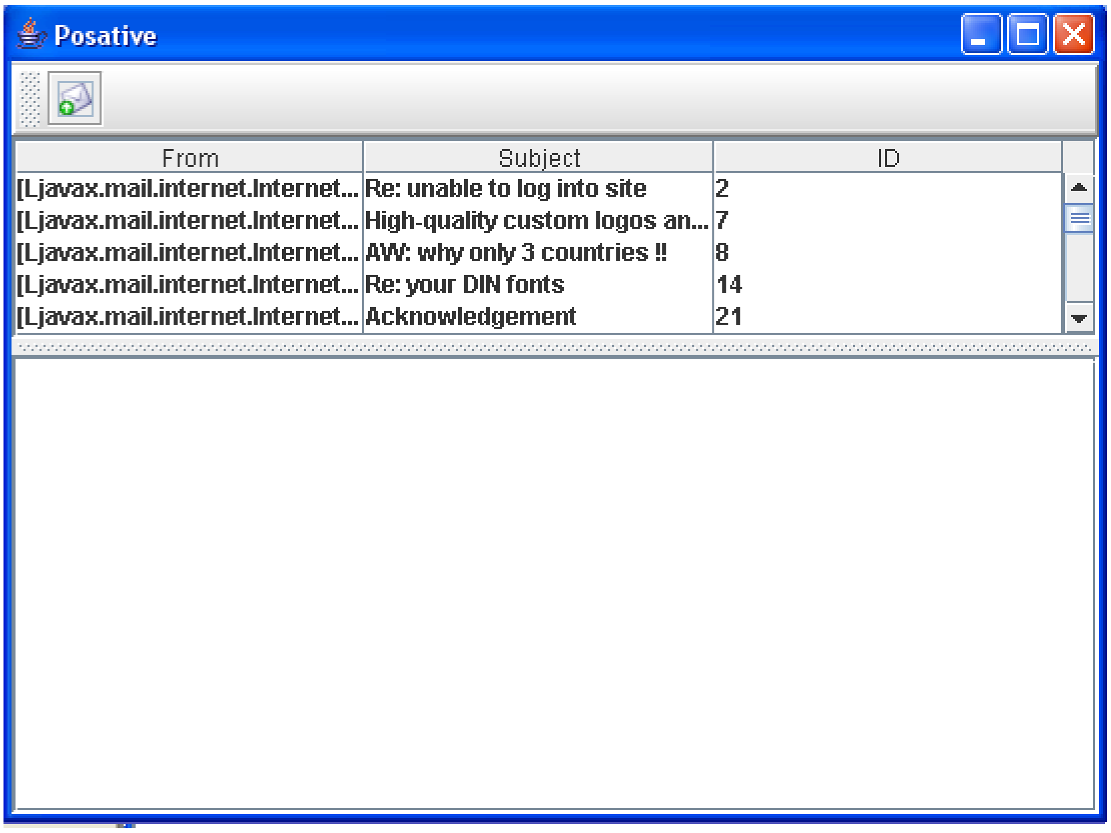
Figure (4.6) Feature Extraction

When extraction is finished, messages will be divided into two lists, Positive list and Negative list according to the extracted features. These lists will be shown in figure (4.7) and figure (4.8).



From	Subject	ID
[Ljavax.mail.internet.Internet...	MyWebCalls account - funds...	1
[Ljavax.mail.internet.Internet...	DynDNS Hostname Expired	3
[Ljavax.mail.internet.Internet...	1 it.dept moderator request(...	6
[Ljavax.mail.internet.Internet...	Information about RapidShar...	4
[Ljavax.mail.internet.Internet...	Customer Survey	5

Figure (4.7) Negative Lists



From	Subject	ID
[Ljavax.mail.internet.Internet...	Re: unable to log into site	2
[Ljavax.mail.internet.Internet...	High-quality custom logos an...	7
[Ljavax.mail.internet.Internet...	AW: why only 3 countries !!	8
[Ljavax.mail.internet.Internet...	Re: your DIN fonts	14
[Ljavax.mail.internet.Internet...	Acknowledgement	21

Figure (4.8) Positive Lists

The last step of the agent work is to filter the messages in the Positive list and Negative list, filtering process will be represented by a feature extraction progress bar as shown in figure (4.9).

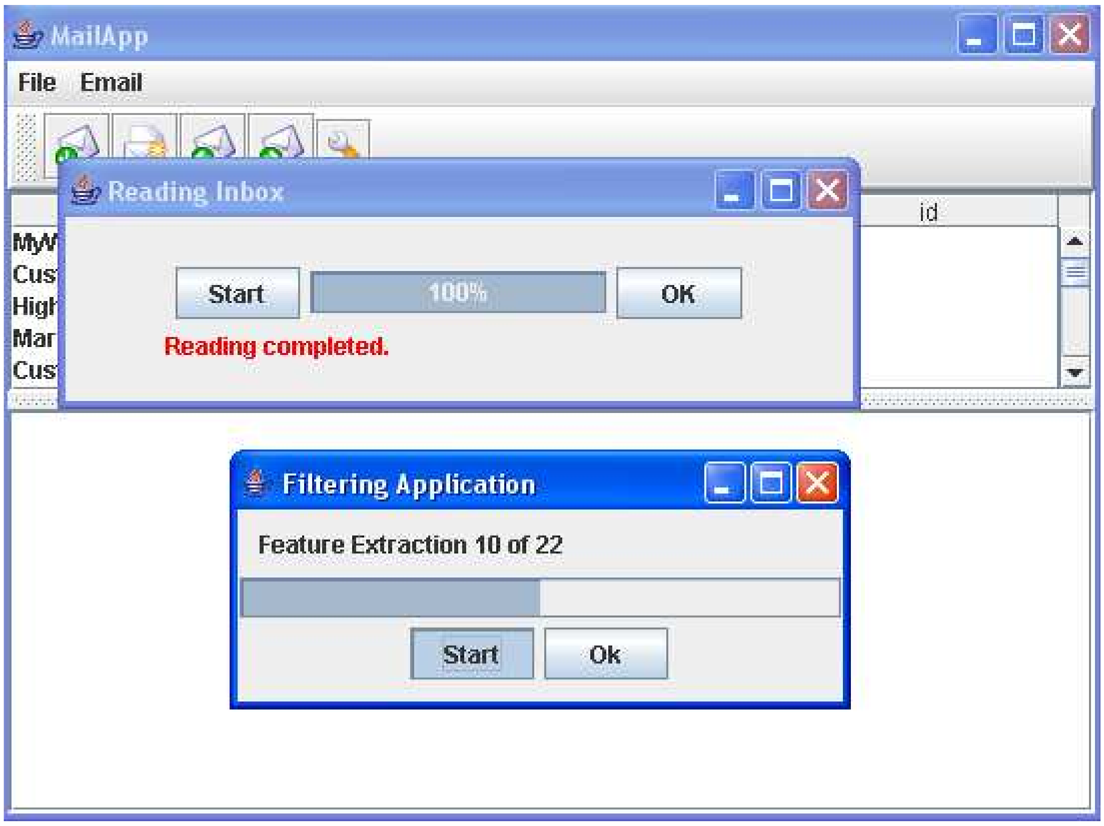
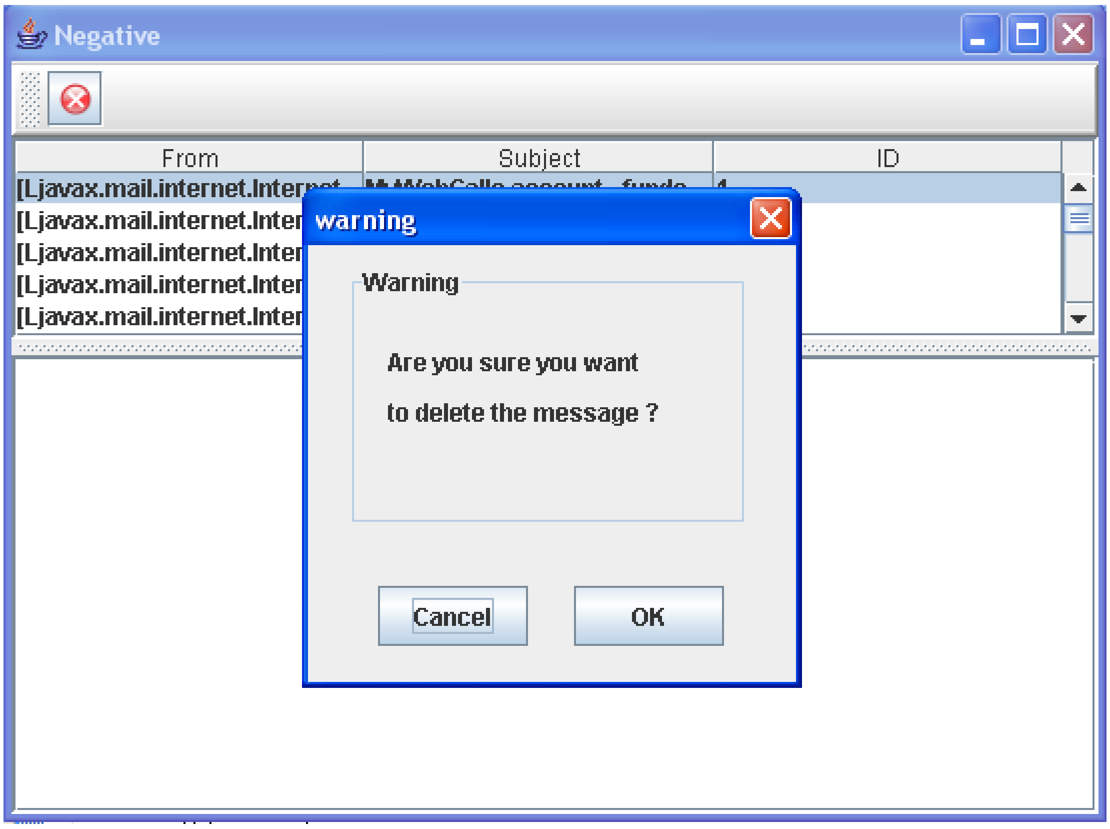


Figure (4.9) Filtering Application

Filtering will be applied on the negative and positive lists after extracting the features. The messages in the negative list will be deleted. The agent consults the user to confirm the deletion before completing the process. A dialog window will appear asking the user to agree on deleting the contents of the negative list, this is shown in figure (4.10).



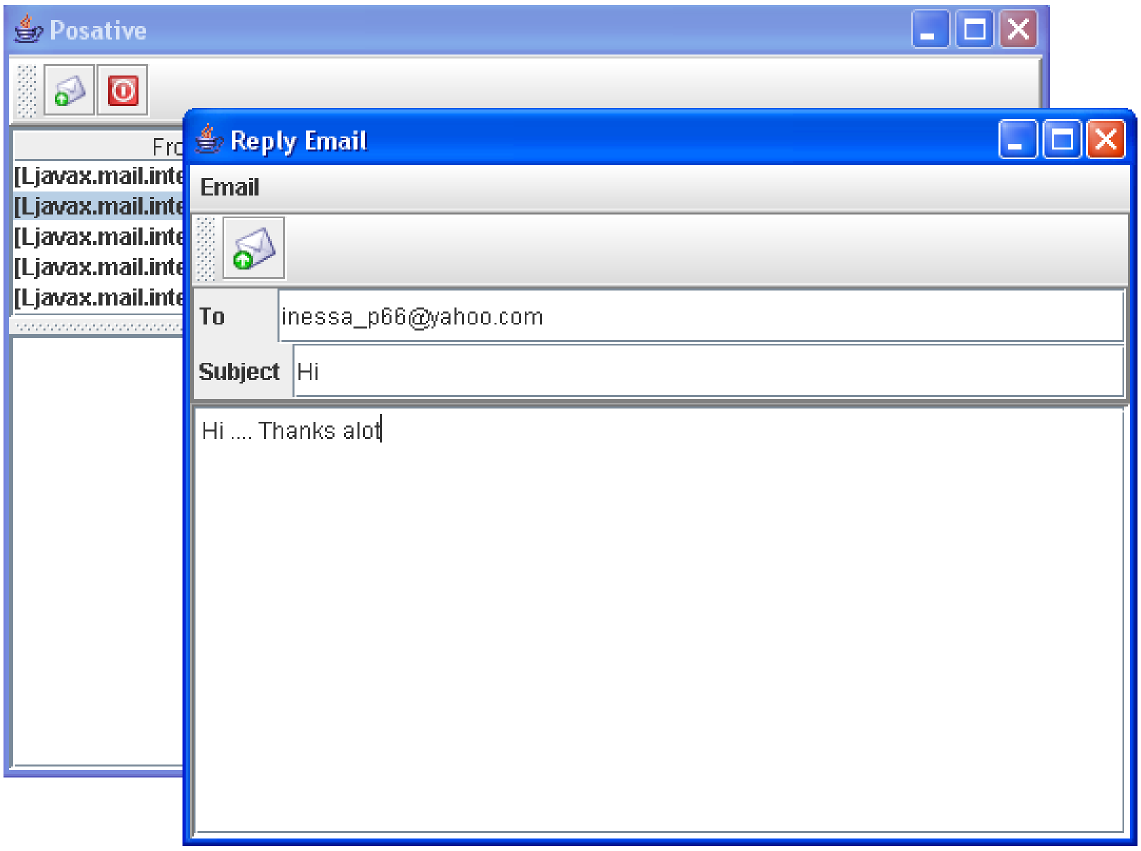
[Ljavax.mail.internet.Internet
[Ljavax.mail.internet.Inter
[Ljavax.mail.internet.Inter
[Ljavax.mail.internet.Inter
[Ljavax.mail.internet.Inter

Are you sure you want

to delete the message ?

Figure (4.10) Warning Message

The agent will then reply or forward the messages in the positive list according to the extracted features as shown in figure (4.11) for the reply and figure (4.12)for the forward.

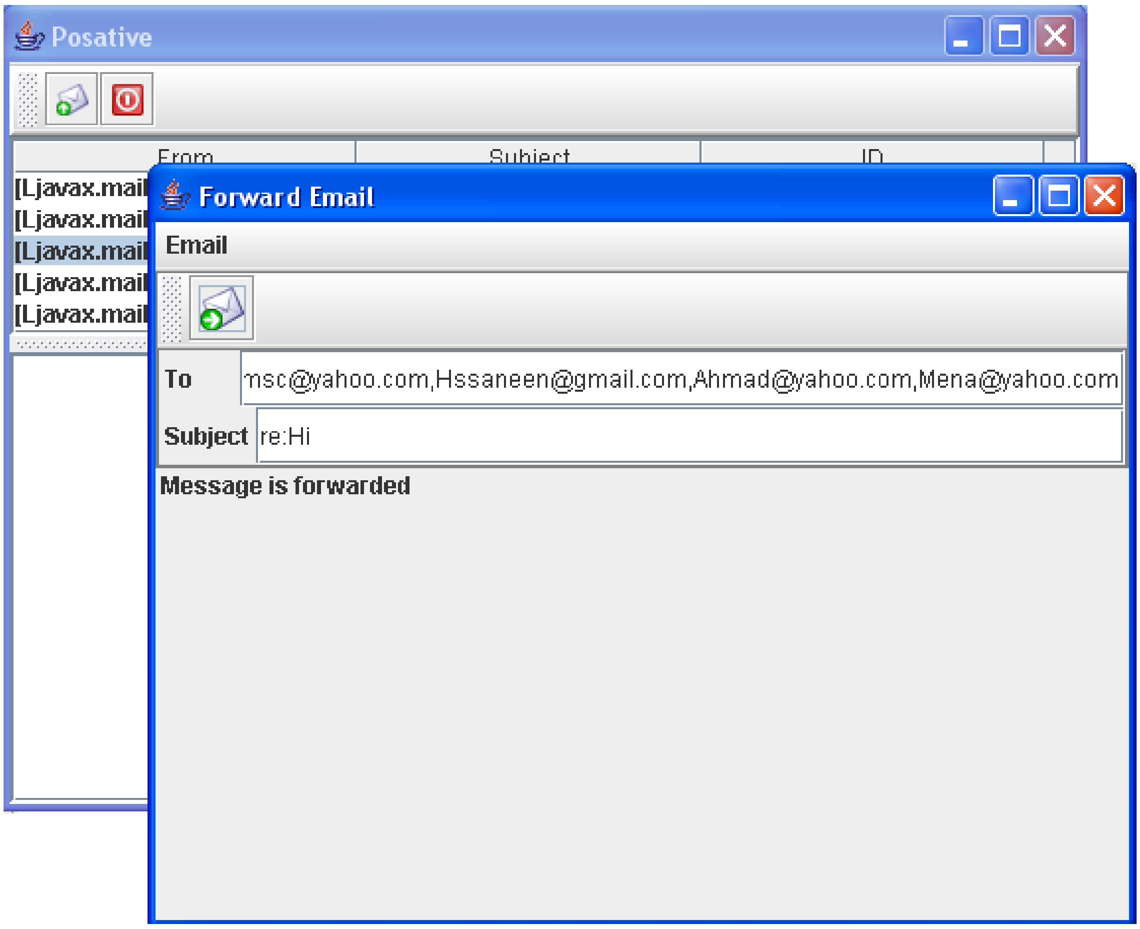


To inessa_p66@yahoo.com

Subject Hi

Hi .... Thanks alot

Figure (4.11) Reply window



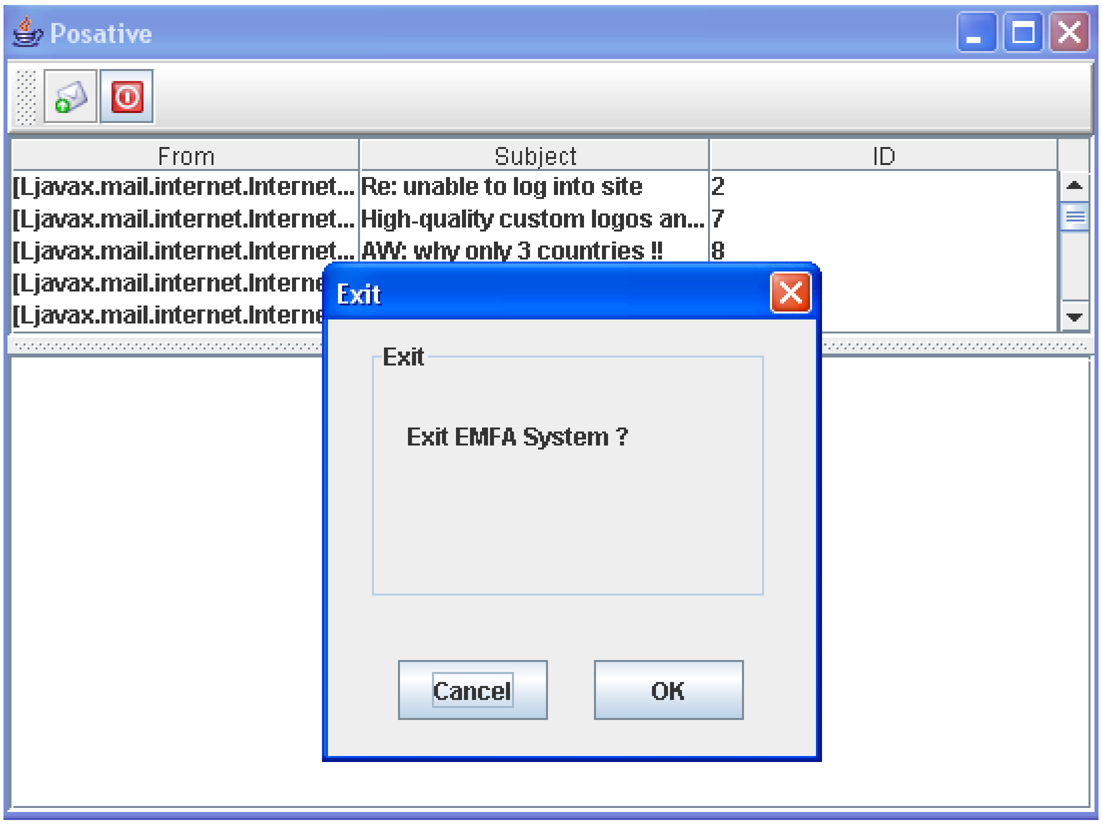
To nsc@yahoo.com,Hssaneen@gmail.com,Ahmad@yahoo.com,Mena@yahoo.com

Subject re:Hi

Message is forwarded

Figure (4.12) Forward window

When the agent finishes doing all of its operations on the inbox, it will ask the user to exit the system, an exit window will appear asking the user to confirm exiting the EMFA system with two buttons one for OK and the other for cancel.



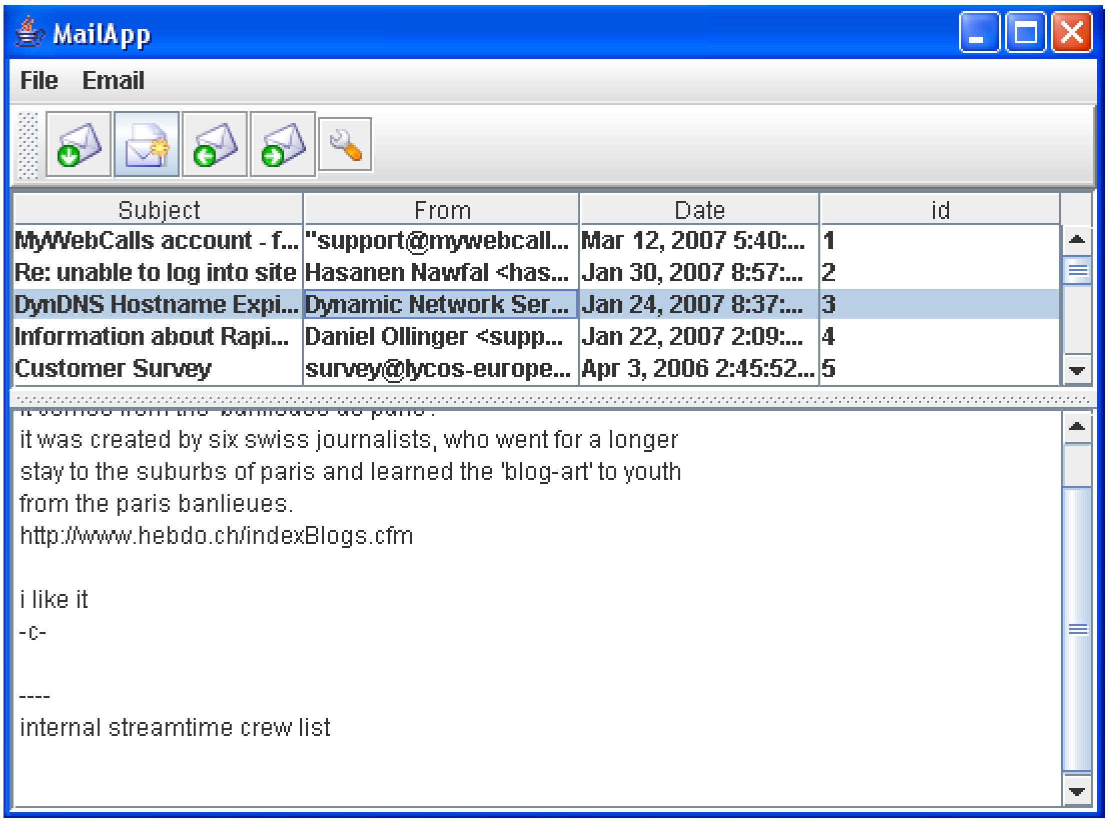
From	Subject	ID
[Ljavax.mail.internet.Internet...	Re: unable to log into site	2
[Ljavax.mail.internet.Internet...	High-quality custom logos an...	7
[Ljavax.mail.internet.Internet...	AW: why only 3 countries !!	8
[Ljavax.mail.internet.Interne		
[Ljavax.mail.internet.Interne		

Exit

Exit EMFA System ?

4.5 Example

In this example, three messages will be taken to show how to apply EMFA on these messages. This example takes first, third and fifth message in order in the Inbox as shown in figure (4.13)



Subject	From	Date	id
MyWebCalls account - f...	"support@mywebcall...	Mar 12, 2007 5:40:...	1
Re: unable to log into site	Hasanen Nawfal <has...	Jan 30, 2007 8:57:...	2
DynDNS Hostname Expi...	Dynamic Network Ser...	Jan 24, 2007 8:37:...	3
Information about Rapi...	Daniel Ollinger <supp...	Jan 22, 2007 2:09:...	4
Customer Survey	survey@lycos-europe...	Apr 3, 2006 2:45:52...	5

```
it was created by six swiss journalists, who went for a longer
stay to the suburbs of paris and learned the 'blog-art' to youth
from the paris banlieues.
http://www.hebdo.ch/indexBlogs.cfm

i like it
-c-

----
internal streamtime crew list
```

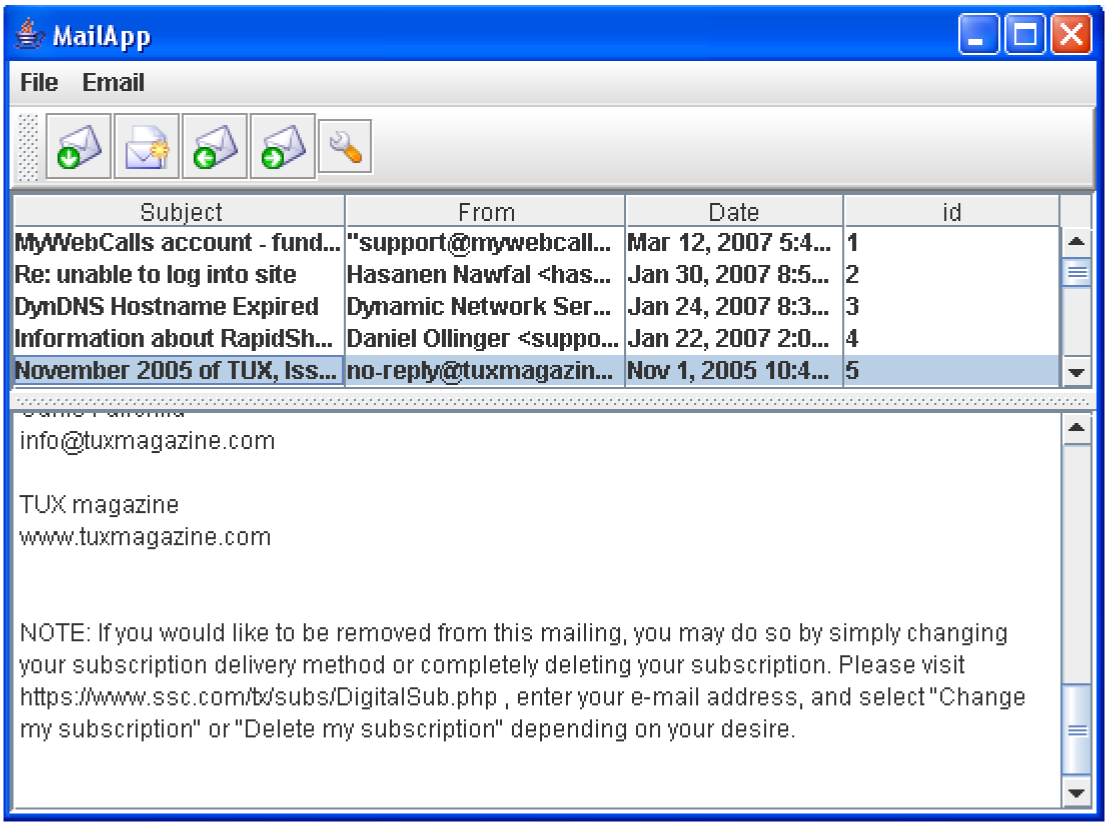
Figure (4.13) messages in example

When EMFA starts working, the features will be extracted from the messages' header. These features are **Subject**, **From** and **Date**, as shown in figure (4.13). Each message has different Subject and From field, while the first message and the third message have the same year date. The fifth message will be negative example because the condition is

< IF the date is before 2007 then it is negative>,

When clustering algorithm is applied, the obtained result show the **negative** list contains the fifth message and the **positive** list contains the first and the third message.

The features will be extracted again from the messages in the positive list. These features are only **Subject** and **From**. The Subject of the first message shows the information of specific account and the From field is (**myweb**) that shows specific message. The Subject of the third message shows hostname and public information, and From field shows Network server. The first message is considered as private message but the third one is not. When the filtering algorithm is applied, the messages in the negative list will be deleted. Figure (4.14) shows the Inbox after delete fifth message and how the next one is replaced instead of it.



Subject	From	Date	id
MyWebCalls account - fund...	"support@mywebcall...	Mar 12, 2007 5:4...	1
Re: unable to log into site	Hasanen Nawfal <has...	Jan 30, 2007 8:5...	2
DynDNS Hostname Expired	Dynamic Network Ser...	Jan 24, 2007 8:3...	3
Information about RapidSh...	Daniel Ollinger <suppo...	Jan 22, 2007 2:0...	4
November 2005 of TUX, Iss...	no-reply@tuxmagazin...	Nov 1, 2005 10:4...	5

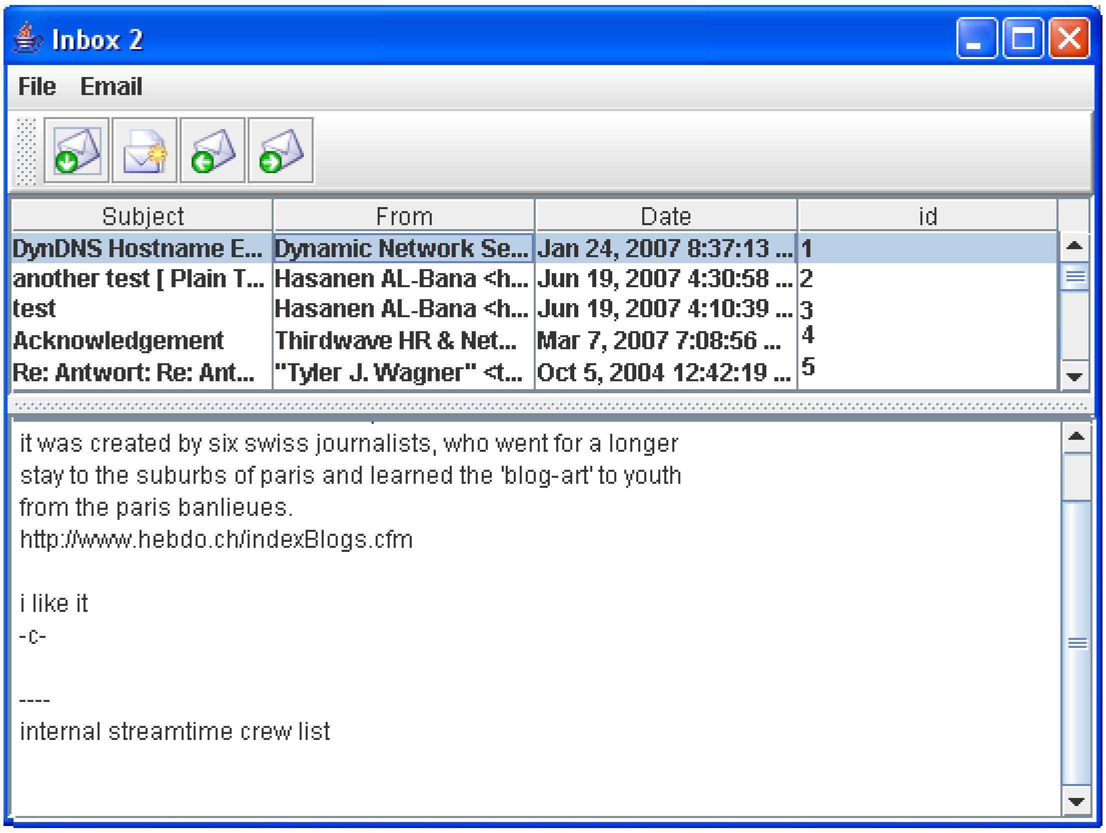
info@tuxmagazine.com

TUX magazine
www.tuxmagazine.com

NOTE: If you would like to be removed from this mailing, you may do so by simply changing your subscription delivery method or completely deleting your subscription. Please visit https://www.ssc.com/tx/subs/DigitalSub.php , enter your e-mail address, and select "Change my subscription" or "Delete my subscription" depending on your desire.

Figure (4.14) deleted message

The first message will be replied to the same sender using a predefined form for the body and the third message will be forwarded to all addresses in the address book. Figure (4.15) shows one of the email addresses inbox that the third message has been forwarded to.



Subject	From	Date	id
DynDNS Hostname E...	Dynamic Network Se...	Jan 24, 2007 8:37:13 ...	1
another test [Plain T...	Hasanen AL-Bana <h...	Jun 19, 2007 4:30:58 ...	2
test	Hasanen AL-Bana <h...	Jun 19, 2007 4:10:39 ...	3
Acknowledgement	Thirdwave HR & Net...	Mar 7, 2007 7:08:56 ...	4
Re: Antwort: Re: Ant...	"Tyler J. Wagner" <t...	Oct 5, 2004 12:42:19 ...	5

```
it was created by six swiss journalists, who went for a longer
stay to the suburbs of paris and learned the 'blog-art' to youth
from the paris banlieues.
http://www.hebdo.ch/indexBlogs.cfm

i like it
-c-

----
internal streamtime crew list
```

Figure (4.15) Inbox2

Chapter Five
Discussion, Conclusions and Future Work

5.1 Discussion and Conclusions:

Many things were noticed and concluded while working on the EMFA system. The following are the most important ones:

- The EMFA system makes it easier for the user to use the E-mail rather than the normal system. The agent will assist the user in opening the inbox, reading the messages, deleting the unwanted messages and replying or forwarding the other messages.

- The features that extracted from the messages will facilitate the comparison between messages' header and the predefined conditions that created by the user.

- In on-line connection the performance of the agent is adapted automatically while the data in offline connection are static.

- The classification algorithm was obtained to get more improvement for the filtering performance because it classified the messages into two lists. That process is easier to apply the filtering on the list than Inbox.

- Java is platform independent, so it can be run on Windows, Linux, Unix and many other operating systems. Due of it is simplicity programs can be written in less time less cost and with less bugs.

During EMFA System implementation, two problems aroused, those problems were:

1. The security of the server that the system was supposed to be connected to was an obstacle; the yahoo server has a very high security level, which it is impossible to connect to. A less secured server was used to establish the connection, which was the **mail.s-ictc.com** server.
2. The delete function of the EMFA system requires an offline connection to work properly. Unlike the reply and the forward functions that work properly online. This requires going offline during the deletion process and then reconnecting to perform the other tasks of the system.

5.2 Future Work

After developing EMFA system, several ideas may improve the system. These ideas have been left as recommendations for future work. These recommendations are:-

1- Giving the user a last chance to check the E-mails before permanently deleting them should be considered, so the deleted messages might be moved into a trash folder.

2- In cases of dealing with inbox with a large number of messages it is recommended to use fuzzy to make an efficient classification for the messages so the agent will work in a better way in such cases.

3- The agent may perform extra tasks in sorting the incoming messages by distributing them into folders with different categories such as folders for messages containing pictures or attachment document…etc. This requires extracting the features from the body of the message instead of the subject and the from fields.

4- Including the possibility of attaching files to the E-mail is one of the ideas that may add extra facilities to the system.

5- The user predefined lists (contact lists) can be modified to be updatable. This can be done when the agent prompts the user to add a contact that happened to receive emails from it repeatedly.

Appendix A

Environment Properties

This section lists some of the environment properties that are used by the JavaMail APIs. The JavaMail javadocs contain additional information on properties supported by JavaMail.

Property	Description	Default Value
mail.store.protocol	Specifies the default Message Access Protocol. The Session.getStore() method returns a Store object that implements this protocol. The client can override this property and explicitly specify the protocol with the Session.getStore(String protocol*)* method.	The first appropriate protocol in the config files
mail.transport.protocol	Specifies the default Transport Protocol.The Session.getTransport() method returns a Transport object that	The first appropriate protocol in the

	implements this protocol. The client can override this property and explicitly specify the protocol by using Session.getTransport(String protocol) method.	config files
mail.host	Specifies the default Mail server. The Store and Transport object's connect methods use this property, if the protocol-specific host property is absent, to locate the target host.	The local machine
mail.user	Specifies the username to provide when connecting to a Mail server. The Store and Transport object's connect methods use this property, if the protocol-specific username property is absent, to obtain the username.	user.name
mail.*protocol*.host	Specifies the protocol-specific default Mail server. This overrides the mail.host property.	mail.host

mail.*protocol*.user	Specifies the protocol-specific default username for connecting to the Mail server. This overrides the mail.user property.	mail.user
mail.from	Specifies the return address of the current user. Used by the InternetAddress.getLocalAddress method to specify the current user's email address.	username@host

Hint: Note that Applets can not determine some defaults listed in this Appendix. When writing an applet, you must specify the properties you require.

Refrence

[Ame07] Amershi, S. and Conati, C.,"Unsupervised and Supervised Machine Learning in User Modeling for intelligent Learning Environment" ,Proceedings of the 12th international conference on Intelligent user interfaces, 2007.

[Big01] Bigus, J., "Constructing Intelligent Agents with JAVA", Professional Developer's Guide, 2001.

[Cam96] Campbell, M. and Aspray, W.," History of the Information Machine", 1996.

[Cha99] Chang, Y., "Email Filtering: Machine Learning Techniques and an Implementation of the UNIX Pin Mail System",
Master thesis,1999.

[Che06] Shen, D., Zhang, B. and Chen, Z., "Adding Semantics to Email Clustering", Sixth International Conference on Data Mining, 2006.

[Cro78] Croker, D., "Email History", Application of Information Netwark, IEEE, Vol. 66, No.11,1978.

[Cro97] Crocker, D., H., "Standard for the format of ARPA

Netwark Text Message (1), RFC 733, 1997.

[Dav02] David, R. and Reilly, M., “JAVA Netwark Programing and Distributed Computing”, 2002. "http://www.davidreilly.com/jnpbook"

[Den98] Deng, Y., “A personalized Email Agent”, Master Thesis, Department of Computer Science and Information Engineering, National Taiwan University, 1998.

[Her97] Hermans, B., “Intelligent Software Agents on the Internet”, First Monday Vol.2, No. 3, 1997.

[Hui00] Hui, Y., “Keyphrase-Based Information Sharing In Multi-Agent System”, Master thesis, 2000.

[Jen98] Jenning, N., R. and Wooldridge, M., “Application of Intelligent Agent”, 1998.

[Kur05] Kuridi, A., H., “Design and Implementation of Intelligent Information System for Junk Filtering”, Master thesis, 2005.

[Las94] Lashkari,Y and Metral,M, “Collaborative Interface Agent”, Conference on Interface agent, 1994.

[Ozg04] Ozgur, A., “Supervised and Unsupervised Machine Learning techniques for text document categorization”, Master thesis, 1994.

[Paz00] Pazzani, M., J., “ Representation of Electronic Mail Filtering Profiles: A user study”, research in part by the National Science Foundation Grant, 2000.

[Pay94] Payne, T.,” Learning Email Filtering Rules with Magi”, A mail agent Interface, Master Thesis 1994.

[Pay97] Payne, T., “Experience with Rule Induction and K-nearest Neighbor Methods for Interface Agents that learn”, IEEE, Transformations on Knowledge and Date Engineering, Vol.9, No.2, 1997.

[Pfe01] Pfeier, R., “Software Agent”, Seminar in new Artificial Intelligence, 2001.

[Ren00] Rennie, J., D., “ifile: An Application of Machine Learning to E-MailFiltering”, Text Mining Workshop, 2000.

[Res01] Resnick, P.,”Internet Message Format”, RFC2822, Network working group, 2001.

[Ric99] Richard B. S. and Jeffrey O. K.,” MailCat: An Intelligent Assistant for Organizing E-Mail”, Proceedings of the Third International Conference on Autonomous Agents, 1999.

[Sah03] Saha, M. and Wadhera, G., “Learning a Distance Metric to cluster E-Mails”, 2003.

[Sun98] Sun Microsoft.Inc, Java Mail Guide for Service Providers”

[Tur03] Turenne, N., “Learning Semantic Classes for Improving Email Classification”, Proceedings of Text Mining and Link analysis Workshop, 2003.

[Wik07] Wikipedia, “Email”, The free encyclopedia, last modified 20-May-2007.

[Woo98] Wooldridge, M. and Jennings, N., R.,” Intelligent Agents: Theory and Practice”, Research supported by UK Science and engineering Research ,1998.