SpamPIG: *Also pigs hate spam*

Marco Ramilli *, Stefano Bianchini †

Introduction. Emails are a really useful technology, made thanks Network, are a basic support for world community. With email you can send certificate, personal mail, notification, and much more important module. For this reason is necessary verify the mail correctness and destroy the mass mail abuse. After some study about recent antiSpam technologies based on point counter, we decided to built a new concept of AntiSpam client. Our methodology is based on Intelligence of Prolog Technology through Alice 2Prolog Project (www.alice.unibo.it) because in this mode we can utilize the native pattern recognition of this framework.

Vision. Spam is a really disseminated and serious problem it causes problems in two ways. There are personal problems of checking your email. The more spam you receive the harder it gets to find the valid email amongst the rubbish. In addition, faster you try to delete the spam then greater the risk of deleting a real email by mistake. On another level spam stresses the infrastructure of the internet. It requires large amounts of disk space to store all the spam until it is collected and requires large amounts of data to be transmitted along the internet's many connections. Guess who ends up paying for all of this. Spam has become a problem because of the economics. A spammer can send out millions of emails for a tiny amount of money. The number of people who respond is tiny but even a tiny fraction responding can be enough to make it profitable. It is difficult, if not impossible, to exert any sort of commercial pressure on spammers currently because the internet and email grew up in a trusted academic world. Let loose in the real world there are too many ways for a spammer to hide who they really are. There are several proposals to change this but none of them are here now. In addition as more and more people get clued up as to how to

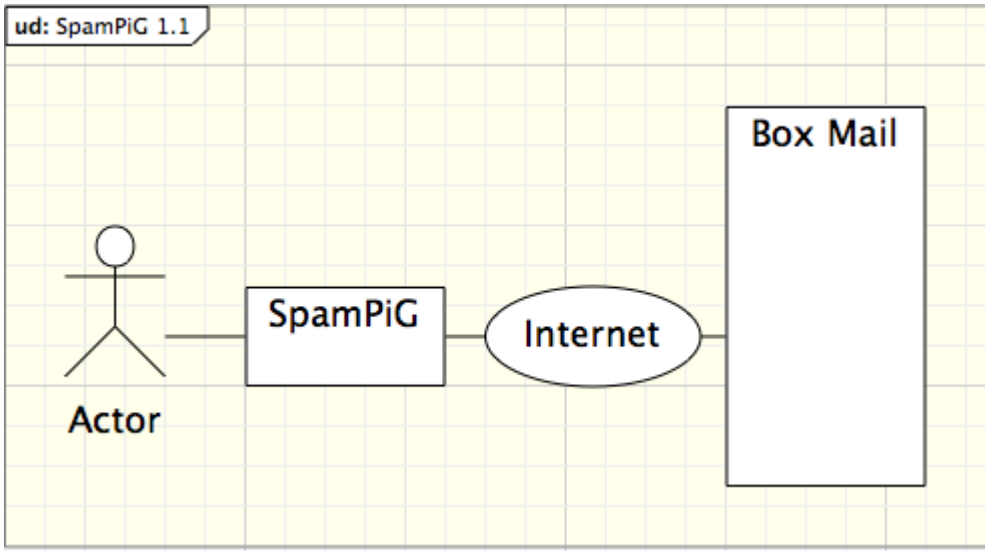
*marco.ramilli@studio.unibo.it
†stefano.bianchini2@studio.unibo.it

avoid spam the spammers respond by pumping out more and more emails to try and catch those few people who don't know how to avoid spam. Novaday we think that is necesary built an intelligence application that understand the right mails from the bad's ones. This program will run as a process and it will do always the same things; eat the spam mails ! This program will be **SpamPiG**: a starved of mails pig !

Analisys. First of all we analize the spam's teminology: **Spamming** is the abuse of electronic messaging systems to send unsolicited, undesired bulk messages. While the most widely recognized form of spam is email spam, the term is applied to similar abuses in other media: instant messaging spam, Usenet newsgroup spam, Web search engine spam, spam in blogs, and mobile phone messaging spam. Spamming is economically viable because advertisers have no operating costs beyond the management of their mailing lists, and it is difficult to hold senders accountable for their mass mailings. Because the barrier to entry is so low, spammers are numerous, and the volume of unsolicited mail has become very high. The costs, such as lost productivity and fraud, are borne by the public and by Internet service providers, which have been forced to add extra capacity to cope with the deluge. Spamming is widely reviled, and has been the subject of legislation in many jurisdictions. **E-Mail spam** is a subset of spam that involves sending nearly identical messages to numerous recipients by e-mail. The main target is " *Terminate the spam* " limiting the false positive and negative mails. SpamPIG may analyze the incoming mails and understand which are good ones and which are bad ones. It can work thanks its own internal prolog theory engine and a simple User Graphic Interface that allows user to select *ham mails* (good ones) against *spam mails* (bad ones). It should have a little black list parser to learn which domains are allowed and which ones are labeled as spammer. It also may has a easy way to communicate with the user, so its Graphic User Interface must has a reader zone (human) and a writer zone (SpamPIG). SpamPIG is a mail client so if connects to deliberate mail box it wants username and password . You should set these information through an easier GUI. Will be useful an easy mail viewer for, eventually, open and modify one or more mail directly on SpamPiG tool. The problem could be classified in classical Model View Control problems, where:

1. Model is the mediator between View and Control.
2. View, better views are the different user interface.
3. Control is the SpamPiGs core, it allows and denies the mails .

In this UML Case Diagram you will find the first and the most intuitive case of use : an uman or artificial actor uses SpamPig to clean its mail.

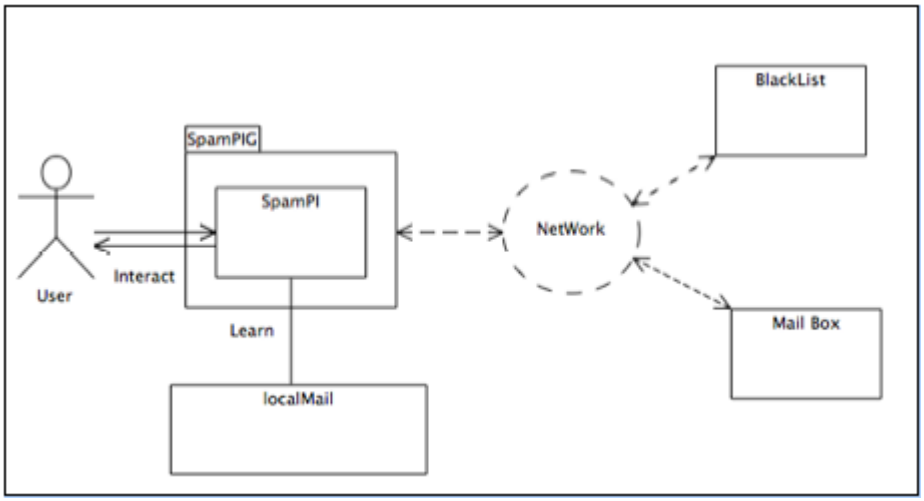


Obviusly SpamPiG uses Internet to connect to Actor' s Mail Box, it cleans spam mail directly on mail box, in this case Actor could download its mail with preferred mail client.

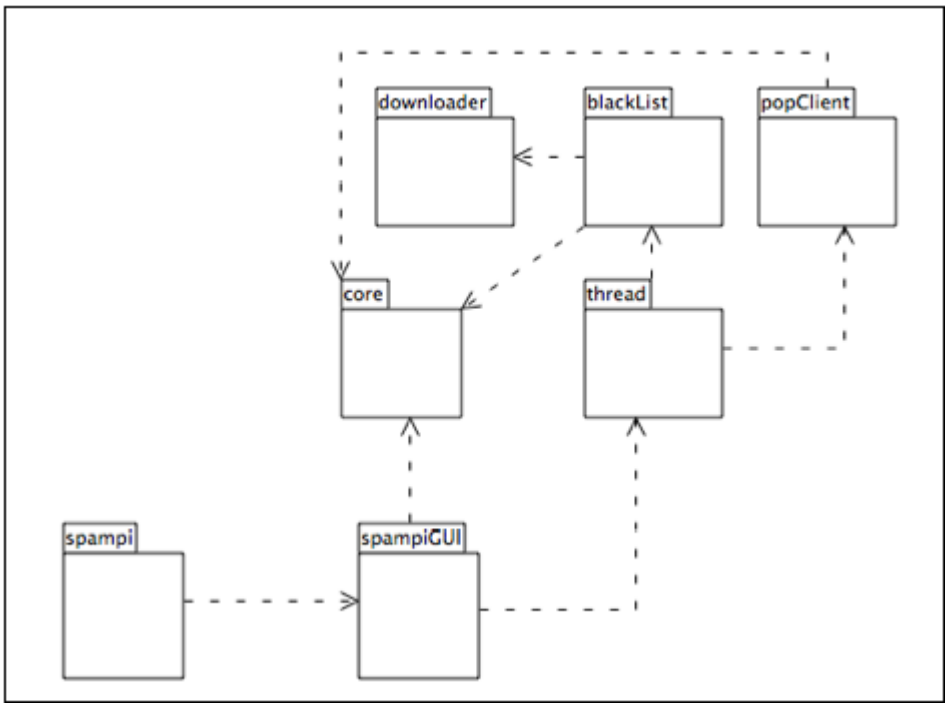
Project. We decided to implement this plain with java Platform independent technologies (java 1.4 or higher) and 2Prolog engine (www.alice.unibo.it). There are two main behaviors:

1. Internal behaviors: each process that compute on local mails are considerate internal .
2. External behaviors: each process that connect with remote machine are considerate External.

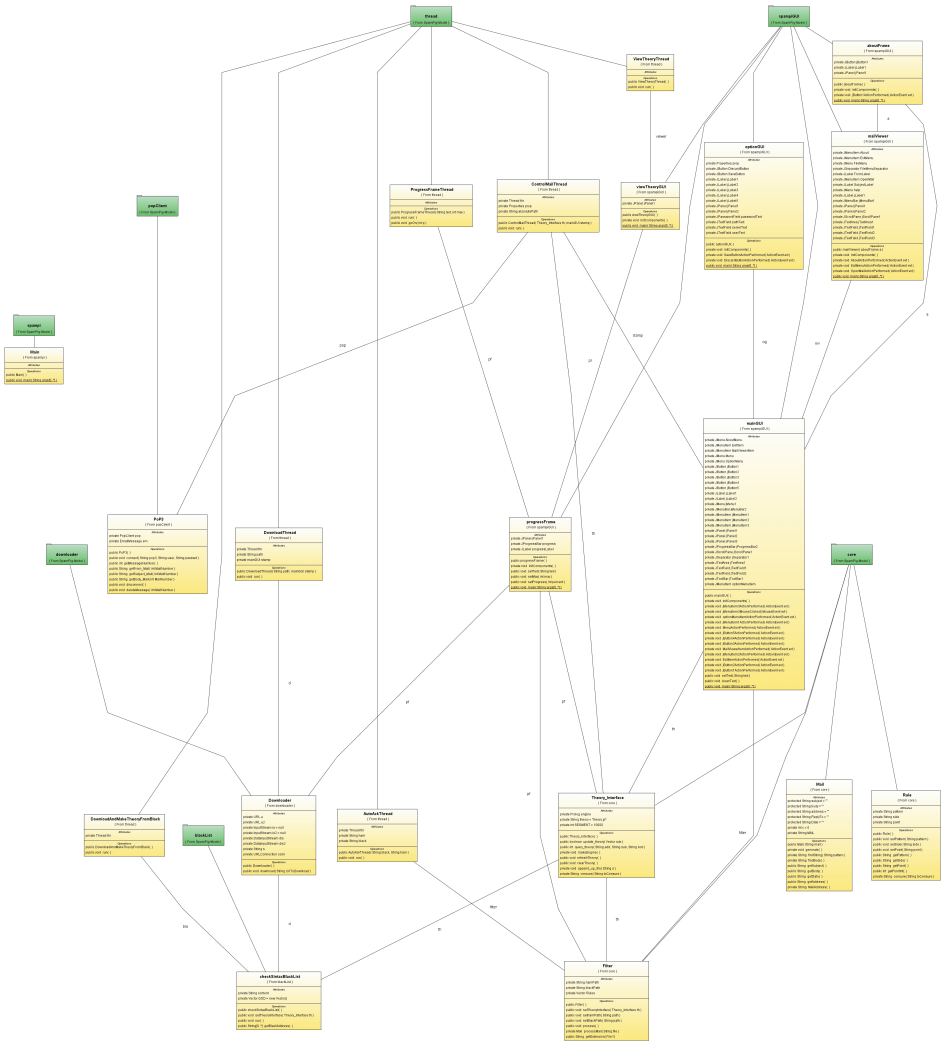
Each external thread must connect to a particular remote server (mail box or black list server), if the remote server is a mail box it must compute the mails and cancelled the spam ones, if the remote server is a black lists it must download inter list and upgrade its prolog theory. The following UML scheme explain the principal behavioural 's case..



An user may interact with SpamPIG, SpamPIG may interact with *Mail Box*, if has to work, else may interact with *BlackList server*, if it has to upgrade its prolog theory. Finally SpamPIG has to understand distinguish *ham* mails from *spam* mails through bayesian filter. Here you will find a simple UML diagram to explain the *component management*.



As you can read there are some thread that make interaction with *popClient* package and *baclkList* one. These interaction are considered the External behavior because they take from remote server something (mail or blackList). All the components except downloader need to interaction with the core package for parsing email in prolog theory and for storing the information. The spampi package contain the main class for start the GUI thread. Following you will find a complete class diagram to understand completely the interaction. We know, it is very small and difficult to print, but if you want know every class and package used in SpamPiG you should zoom-in with your pdf Reader, the image is hight quality so you can do that in easy way.



The above class diagram explicit the interaction described before. Another important aspect to SpamPIG is its intelligence. In fact it could understand if a particular mail is spam thanks a useful prolog theory. This theory , obviusly, must auto upgrade any time SPamPIG reach a spam mail. The main concept of baysian filter presented in this technology is a *deadLine point*. Each mail acquire some content points, the poits are delivered by a spam policy that uses a prolog theory to matching mail body with some black list and bayesian filter words. The *deadLine point* is configurable in front of each necessity . Prolog theory is so structured :

1. static zone to iterate the pattern matching
2. dinamic zone able to auto upgrade itself at runtime during SPamPIG history.

Following you will find the theory 's description.

1. **First zone:** fondamental theory.

Prolog Theory

```
%ADDRESS
address([A|T],Punteggio) :-
        address(T,Punteggio).

address([],0).

%SUBJECT
subject([A|T],Punteggio) :-
        subject(T,Punteggio).
subject([],0).

%BODY
body([A|T],Punteggio) :-
        body(T,Punteggio).
body([],0).

%CHECKING
check(Address,Subject,Body,Punteggio) :-
        subject(Subject,X1),
        (address(Address,X2),
        (body(Body,X3),
        (Punteggio is X1 + X2 + X3))).
```

As you can read in this theory zone there is *the point concept*. For istance: if a mail came from marco.ramilli@studio.unibo.it, probably is a ham mail,

so this theory decrement its point. We must remember that a mail is considered spam only if its points are above 10; so if you put -100 points to marco.ramilli's mail, surely it will be considerate a ham one. Moreover the totally points are assigned using some policy:

subject : if there is a well know pattern on subject (XXX, penis, buy it, ecc..) the mail isn't ham.

mail : all domains from blackList are blocked (+ 100 points) .

body : SpamPIG try to find some internal body pattern as before.

You can upgrade this kind of theory just press on "Refresh Auto Knowledge"[1] after insert the correct ham mails and spam mails path.

2. **Second zone:** autoUpgrade theory.

```
address(['08273732856852@'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['082viiofvl@yahoo.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['083bkhdod@bigfoot.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['083pzcc@earthlink.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['083rlp@juno.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['083rpjl@yahoo.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['083tig@aceweb.net'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['08432@earthlink.net'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['085xznvn@msn.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['086akjdxtp@yahoo.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['086mblkq@hn.vnn.vn'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['086niusagm@fcc.net'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['086zps@msn.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['087cdtx@msn.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['087kpm@concentric.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['088setlzpq@juno.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['089rlu@seductive.com'|T],Punteggio):- address(T,P), Punteggio is P + 11.
address(['08bljdtzp@msn.co'|T],Punteggio):- address(T,P), Punteggio is P + 11.
```

This theory was build with automatic process. SpamPiG could upgrade its theory using a BlackList server and bayesian filter built in for capture spam mail. As you can read the mails' points are always over eleven because these mails are downloaded from blackList server, for this reason are considerate spam forever. The second zone theory based on black list data, is upgrade every time we decide to upgrade it pressing the relative button. The following image represent the black list structure, parsed to be translated into a second zone prolog theory.

[1] It is an implementation Button, you will see later what it is, at this abstraction's level imagine Refresh Auto Knowledge as a process that refresh the prolog theory

```
zqxtyn.com  JUNK
zrafty.com  JUNK
zsupper.com JUNK
ztenterprises.com JUNK
ztenterprises.comtremendousinternetnews.com     JUNK
ztoall.com  JUNK
ztoday.com  JUNK
ztqlrni.ph  JUNK
zumpo.com   JUNK
zunoz.com   JUNK
zvv261.com  JUNK
zxc0923.com JUNK
zynic.com   JUNK
zzim4u.co.kr      JUNK
zzizzi.tv   JUNK
$random$@   SPAMMER
$randomizec@netscape.com      SPAMMER
$strippeduserft@netscape.com  SPAMMER
%counter%2hgh@    SPAMMER
'lloxhe@email.com SPAMMER
'rehwiemobys@yahoo.com  SPAMMER
'rekassna@email.com     SPAMMER
'reoflb@yahoo.com SPAMMER
'res93678@yahoo.com     SPAMMER
'rex976393@yahoo.com    SPAMMER
```

As you can see, the black list is divided in two kind of information:

1. **JUNK**: referred to spammer domains.
2. **SPAMMER**: referred to spammer mail addresses.

We have to parse it in two different way, the component that do it, is checkSintaxBlackList, that divides and make a rule vector.

```
public void run(){
    try{
     d.download("http://basic.wirehub.nl/spamlist.txt");
     File inFile = new File("blacklist.temp");
     FileInputStream fis  = new FileInputStream(inFile);
     BufferedReader in = new BufferedReader(new InputStreamReader(fis));

     String thisLine = "";

        while ((thisLine = in.readLine()) != null) {
                if( thisLine.endsWith("SPAMMER") == true ){
                    Rule r = new Rule();
                    r.setPattern(thisLine.split("SPAMMER")[0].trim());
                    r.setSide("address");
                    r.setPoint("ll");
                    this.GOD.addElement(r);
                }else{
                    Rule r = new Rule();
                    r.setPattern(thisLine.split("JUNK")[0].trim());
                    r.setSide("address");
                    r.setPoint("ll");
                    this.GOD.addElement(r);
                }
```

Rule vector is a kind of vector built in this way:

```
public class Rule {
    private String pattern;
    private String side;
    private String point;
    /** Creates a new instance of Rule */
    public Rule() {
    }


    public void setPattern(String pattern){
        this.pattern = censure(pattern);
    }
    public void setSide(String side){
        this.side = side;
    }
    public void setPoint(String point){
     this.point = point;
    }
```

Where, **pattern** is matchs element, **side** could be address, body or subject and **point** represent the current weight into spamPIGs system . In blacklist upgrade case, we use only rule with side equals to address . The other fields will be used during auto knowledge phase. The component called TheoryInterface translate these rules in prolog theorys rows.

```
while(counter < rule.size()){
// to initialize the signal var.
pf.setText("Process Progress: "+counter+" of:"+rule.size());
pf.setProgress(counter+1);
pf.repaint();
segno = " ";
if(((Rule)(rule.elementAt(counter))).getPointInt() > 0 )
 segno = "+ ";
 if( ((Rule)(rule.elementAt(counter))).getSide().equalsIgnoreCase("address") == true ){
 // address
 query_for_update = query_for_update
 +" address([’"+((Rule)(rule.elementAt(counter))).getPattern()+"’|T],Punteggio):- "
 + " address(T,P), Punteggio is P "+segno+((Rule)(rule.elementAt(counter))).getPoint()"
 +". \n";
  }
  if( ((Rule)(rule.elementAt(counter))).getSide().equalsIgnoreCase("subject") == true ){
  // subject
 query_for_update = query_for_update
  + " subject([’"+((Rule)(rule.elementAt(counter))).getPattern()+"’|T],Punteggio):- "
```

```
 + "subject(T,P), Punteggio is P "+segno+((Rule)(rule.elementAt(counter))).getPoint()"
 +". \n";
  }
  if( ((Rule)(rule.elementAt(counter))).getSide().equalsIgnoreCase("body") == true ){
  // body
  query_for_update = query_for_update
  + " body(['"+((Rule)(rule.elementAt(counter))).getPatter()+"'|T],Punteggio):- "
  +"body(T,P), Punteggio is P "+segno+((Rule)(rule.elementAt(counter))).getPoint()"
  +". \n";
   }
counter++;
stop++;
if (stop==SEGMENT){
   this.append_up_file(query_for_update);
   stop = 0;
    query_for_update= "";
 }
}//while
```

For each rule inside the rule vector , we make a prolog theory string as followed code: side([pattern—T], Punteggio):- side(T,P), Punteggio is P + Point. Where:

- Side is the side of the rule.
- Pattern is the pattern of the rule.
- Point is the point value of the rule

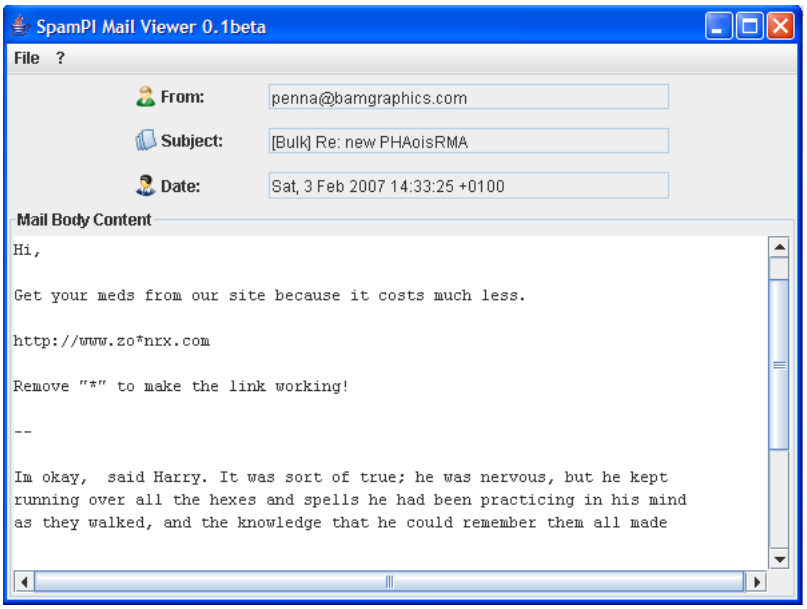
It is really important guarantee the first zone of prolog theory placed on bottom side of the file. To do that we built the append method, called append up file that upgrade the new theory at top of file. You can see it in the next picture.

```
//append the String at firs line of a file !
private void append_up_file(String s){

    try{

        System.out.println("I'm appending !!!!");

        // temp file
        File outFile = new File("temp.tmp");
        // input
        File inFile = new File(this.theory);
        FileInputStream fis  = new FileInputStream(inFile);
        BufferedReader in = new BufferedReader(new InputStreamReader(fis));

        // output
        FileOutputStream fos = new FileOutputStream(outFile);
        PrintWriter out = new PrintWriter(fos);

        String thisLine = "";
        int i =1;
        while ((thisLine = in.readLine()) != null) {
            if(i == 1) out.println(s +" \n");
                out.println(thisLine);
                System.out.println(thisLine);
            i++;
        }
        out.flush();
        out.close();
        in.close();
```

Another method to upgrade prolog theory is to use the auto knowledge, that is teaching to SpamPIG what is good and what is evil. To do this we also use the other role fields: subject and body. SpamPIG takes these fields from ham and spam mails and set a different value to them. For instance SpamPIG assigns a negative points to ham mails and a positive point to spam ones. To obtain these information SpamPIG transform the input mail (ham and spam) through mail filter called filter, in a big string. Then the string will be passed to the mail component that divide it in five fiels:

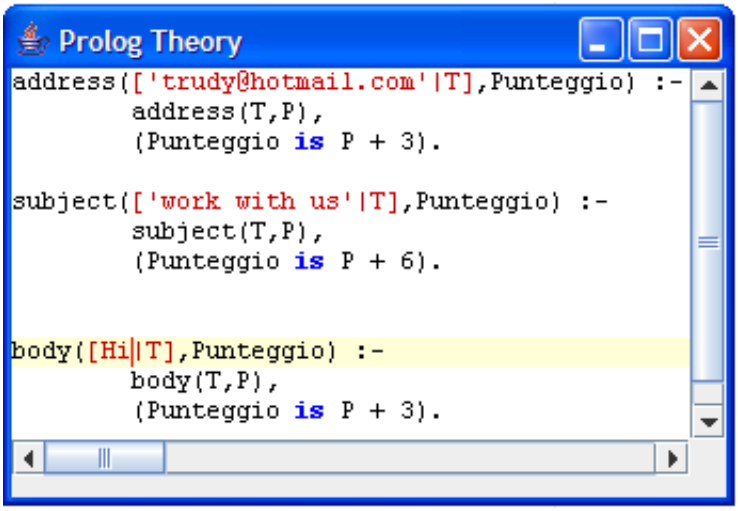
- Address: it contains the sender mail address.
- Subject: it contains the subject of the mail.
- Body: it contains the mail body.
- ReplyTo: it contains the reply to address.
- Date: it contains the mail date.

The mail object is also used from mail viewer, you can see it in the next picture.

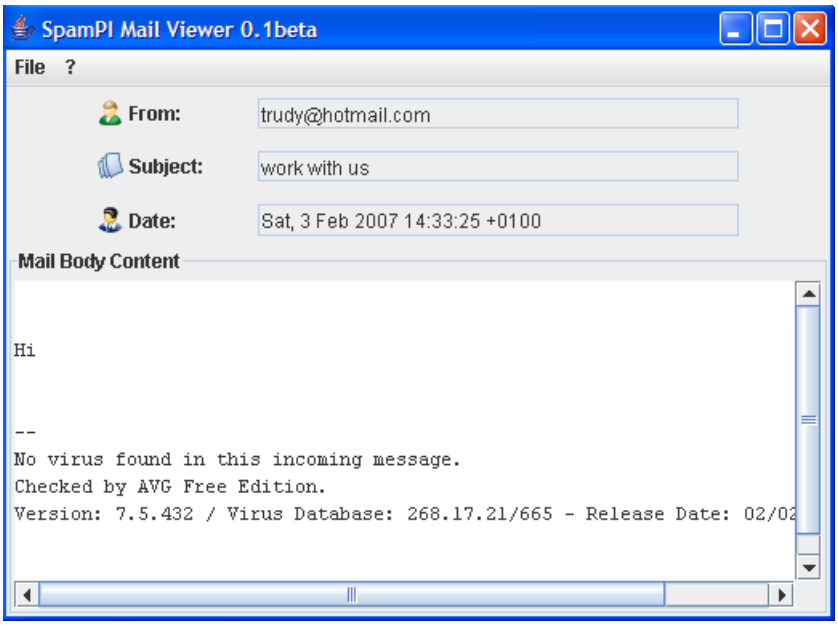


So, we know how SpamPiG works, we know how it is structured, we know how did its theory we have only to know how it is implemented.

Using Prolog Theory. In this section we look a simple example to understand how SpamPiG uses Prolog theory to distinguish spam mails from ham ones. SpamPIG has a theory, result of auto-knowledge and blacklist learning method (we show only the second part, not the first core part of theory), the following image represent this concept:



Now we want focalize our attention to the following mail to analize:



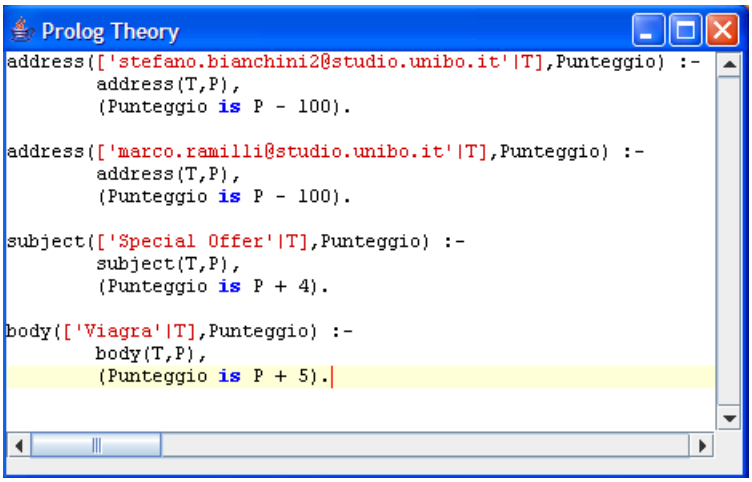
According to us this mail is, of course spam. But it is a spam mai for SpamPIG? We can analyze his behaviour through its theory, step by step:

1. Prolog Engine recognize the From address field trudy@hotmail.com and assign to this mail a point *+10* , because the From address field match with the first prolog theory row .

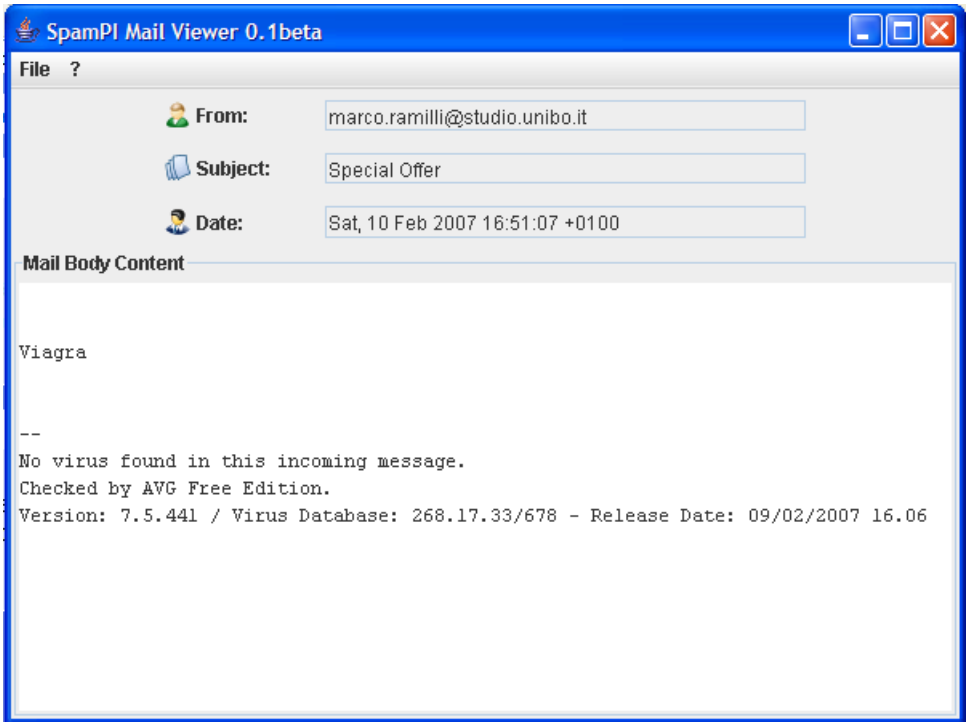
2. The subject is work with us, so the point assigned to mail is *+6*, because subject match with the second row of prolog theory .

3. The final result is the sum of all points: *10+6+3=19*. This mark is called **SpamPiG points**.

We consider *10* points as a correct threshold to separate good mails from spam ones. The SpamPiG points of this mail are higher than threshold so also for SpamPiG this particular mail is spam. Now we consider the opposite case: SpamPIG has also a part of theory about good mail:



So, to analyze the otherside of SpamPiG "Intelligence" we focalize our attention to the following mail:



We can see SpamPigs behaviour step by step, as in the spam example:

1. Prolog Engine recognize From address field *marco.ramilli@studio.unibo.it*and assign to this mail a point *-100* (match with the first row of SpamPIG theory)

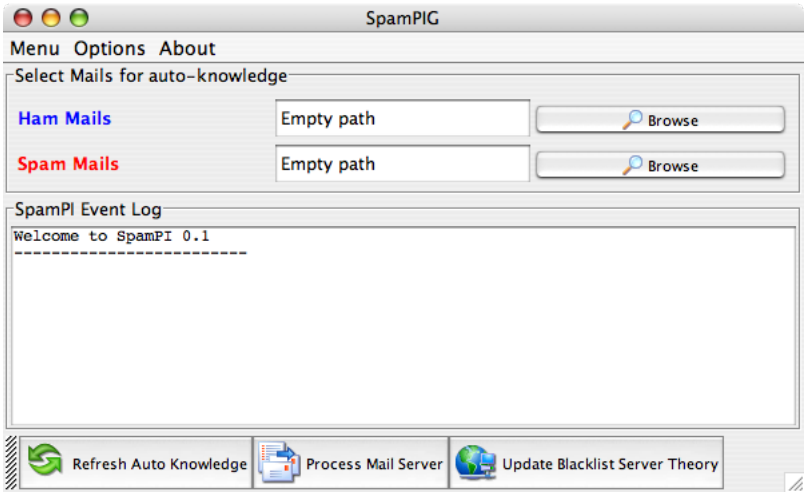
2. The subject is *Special Offer*, so the point assigned to mail is *+4* (match with the second row of SpamPIG theory)

3. The body contains *Viagra*, so we can update point with *+5* (match with the third row of SpamPIG theory)

4. The final result is the sum of all points: *-100+4+5=-91* spampig points.

In this mail we tried to trick SpamPIG: subject and body pretend to be part of spam mail, but the address field is a good address: at the end, -91 spampig points are enough to consider this mail as good!

Implementation. In this paragraph you will find the technical notes about SpamPiG implementation. SpamPiG is implemented in Java[2] Platform Independent Technology and TuProlog[3] for the "intelligent" section. After installation (we will say later about installation procedure) you can open SpamPiG just clicking on .jar file.

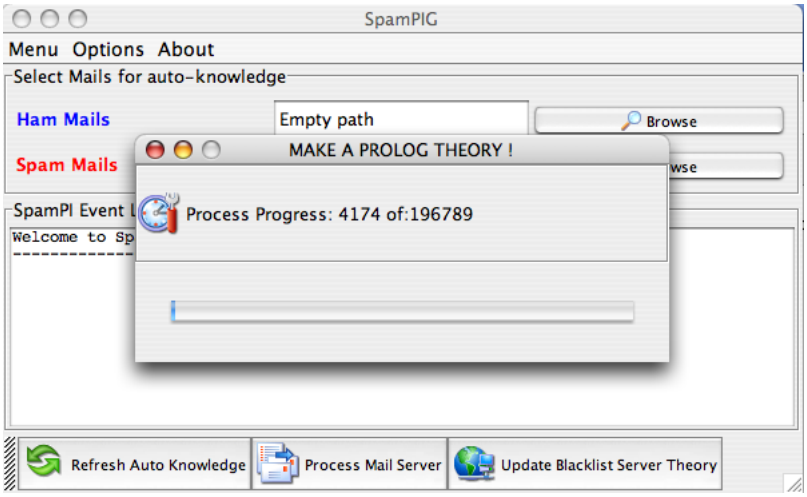


This picture represents the Main window that you will see. In this window there are some push-button to press for do something. First of all take a look to the bottom right push-button called *Update BlackList Server Theory*, if you press this one SpamPiG will connect to a remote server for downloading the last version of BlackList then it will convert the entire file just downloaded in a prolog theory following a subset rules. This operation will take a lot of time, in

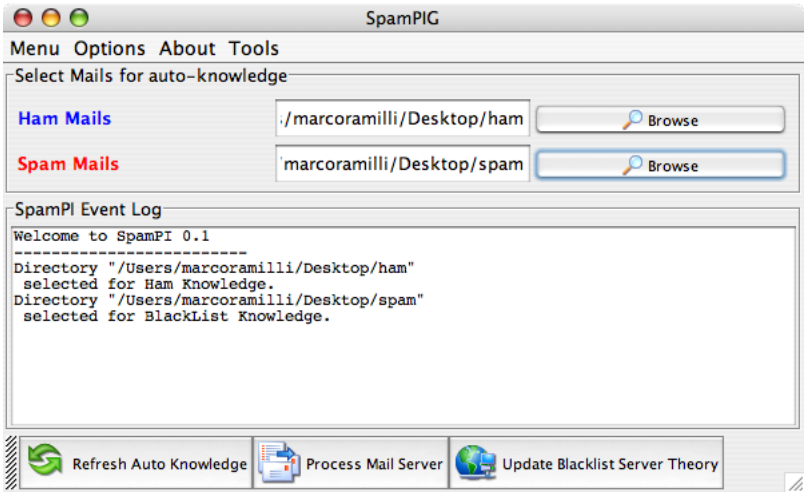
[2] Java is an object-oriented programming language developed by Sun Microsystems in the early 1990s. Java applications are often compiled to bytecode, which may be compiled to native machine code at runtime.

[3] tuProlog (also called 2P) is a Java-based light-weight Prolog engine developed at the Alma Mater Studiorum - Universit di Bologna, and maintained by the aliCE Research Group based in Cesena, at the Second Faculty of Engineering, with some members working at the Faculty of Engineering located in Bologna.

particular the blackList-to-theory conversion can thake some hours !! So make attention to use it . Following SpamPiG is working to take the new BlackList (5MB).

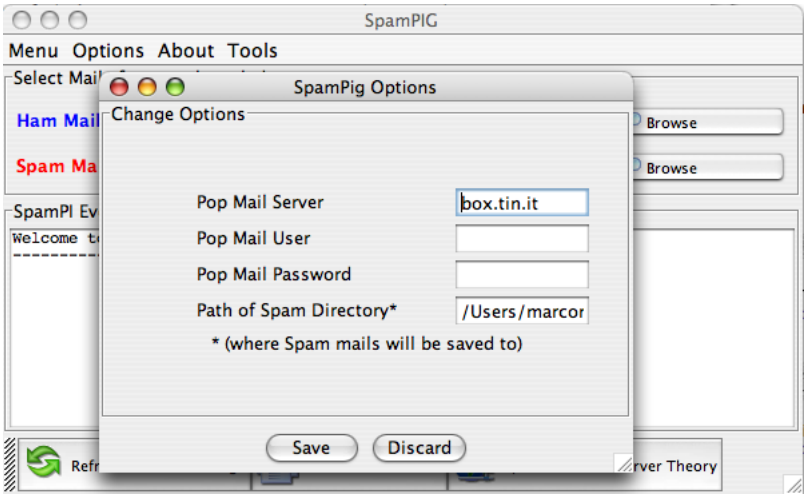


Another interesting SpamPiG' s goal is the bayesian's tests. If you want teach to SpamPiG what is ham and what is spam you may select, using the appropriate push-button, the two folders (one for spam mail and one for ham mail) contain the mails that you consider as spam and as ham. After selected the correct folders you can press on *Refresh Auto Knowledge*, it will show some windows of working in progress and in the end you have a new ad-hoc theory built on your personal mail.



Well, we have spoken only about the prolog's theory, it will be useful understand how SpamPiG works on user's Mail Box. So, first of all you have to

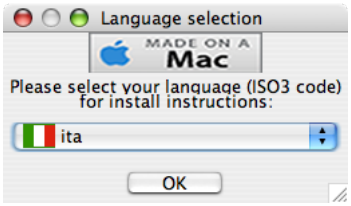
configure the correct mail box parameters; to do this you have to go under *Option*, then *change option* and insert the right parameters inside the right field as you can see in the next picture.



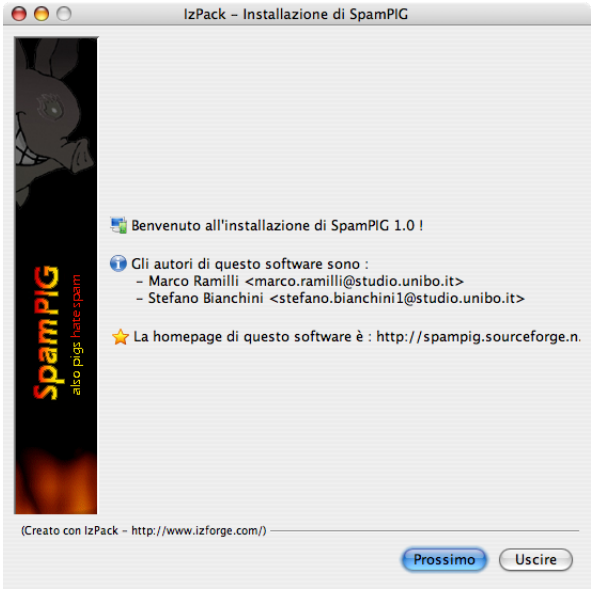
Finally you can click on *Process Mail Server* and wait that SpamPiG finish its work. In the center window you can see the errors or the right action that SpamPiG do. It could be useful to understand why SpamPiG crash, if it happen.

```
Welcome to SpamPI 0.1
-------------------------
Try to Connect to: box.tin.it
Connected to:box.tin.it
Message number --> 0
Disconnected
Try to Connect to: box.tin.it
Connected to:box.tin.it
Message number --> 0
Disconnected
```

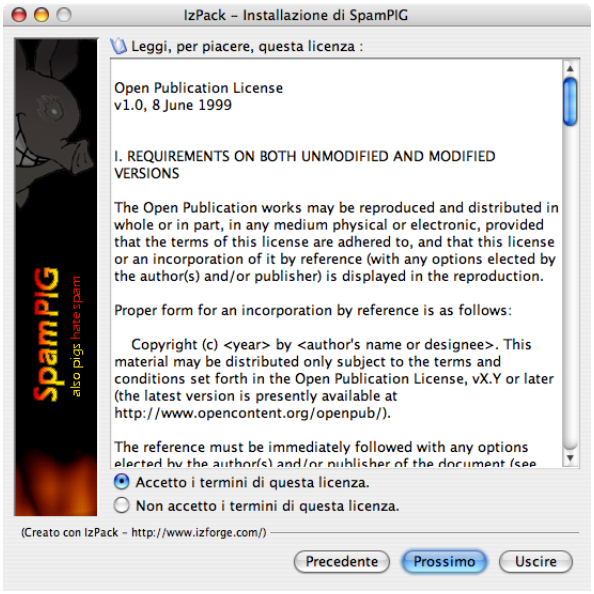
Installation Pack. In this section we describe the graphic install pack built to help end-user, here we describe in very easy way how you have to do for install SpamPiG tool. To take start just double-click on SpamPiG.jar and select your language. This panel will condition the language of your installation. If you select Italian the installation language will be Italian Language, on the other hand if you select English the future instalation language will be English. So select your language and click on Ok push-button.



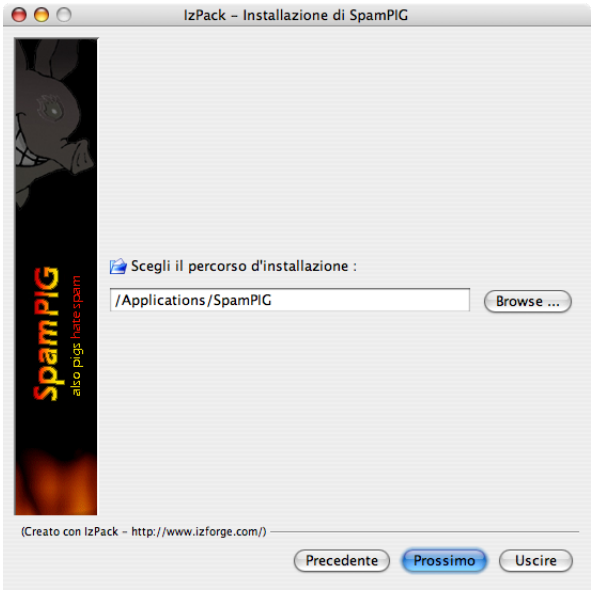
Follow the guide procedure to install it. In this section you have to read the global information and then click next.



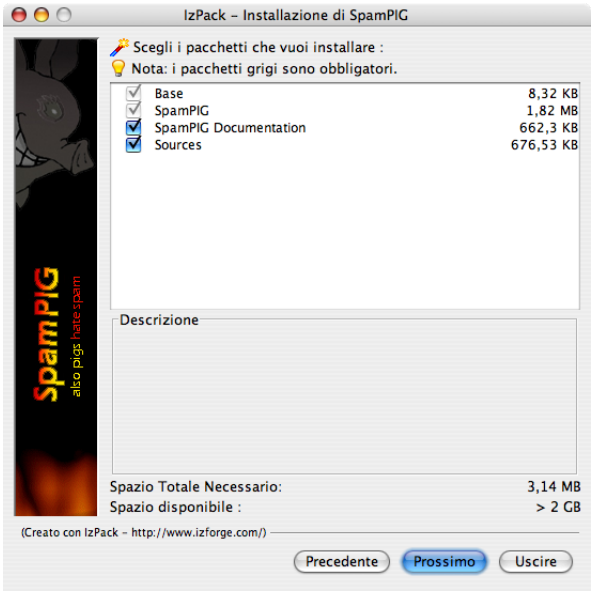
Read the license contract (Open Source) and then click on *Next*.



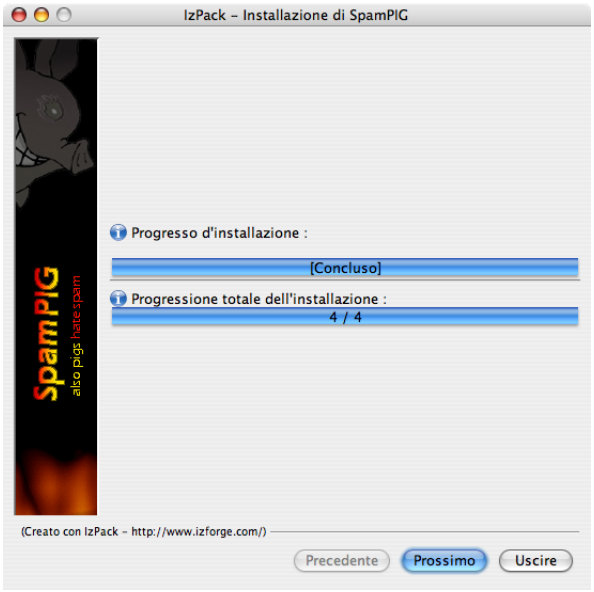
Select your installation's Folder and then click on *Next*.



Now select the installation detail, if you want install the source code, the documentation and so on.



Then it will show and short detail 's reassumed of your installation, click on *Next*.



Well, now you have just finished your installation, if you are using Macintosh or Linux Operating System, go under your installation folder and run the correct script :

1. SpamPIGLinux: if you are using Linux OS

2. SpamPIGMAC: if you are using Mac OS

3. SpamPiG.bat: if you are using Windows[4]

If you need to remove SpamPiG inside your installation folder you can find a Uninstall' s folder that contains an easy tool *Uninstall.jar* allowed to remove SpamPIG.

Conclusion. SpamPiG is only a simple case of study of antiSpam engine with prolog techlonogy. We understand that prolog could be very useful in informatic sience for this reason we try to fight spam with it. SpamPiG is started as a simple project but we know that it could be really interesting for a lot of people that want fight spam in a different approach. If someone want contribute with this project please contact marco.ramilli@studio.unibo.it . Thanks a lot

Marco & Stefano.

[4]if you use windows you can also make your selection in Programs-¿SpamPiG-¿SpamPiG, so under Windows you can adopt the normal procedure as ather programs.