SPAM-Buster

Projektarbeit

Marc Fritsche, Daniel Dilitz

HSR Hochschule für Technik Rapperswil

Wissensbasierte Systeme

Prof. Stefan Keller

28. Juni 2004

Änderungsgeschichte

Version	Datum	Beschrieb	Kürzel	Status
1.0	01.06.04	Erste Version	mfritsch	freigegeben
1.1	14.06.04	Hinzufügen von Installation, Konfiguration, Darstellung	mfritsch	freigegeben
1.2	20.06.04	Hinzufügen von Projektinformationen sowie der Lizenzbedingungen	mfritsch	freigegeben
1.3	26.06.04	Ergänzungen sowie beschreib des Robinson Algorithmus	ddilitz	freigegeben
1.4	28.06.04	Hinzufügen der Erläuterungen über Trainingsdaten	mfritsch	freigegeben

Inhaltsverzeichnis

Abbildungsverzeichnis

Projekt

Homepage

SPAM-Buster wird als OpenSource Projekt auf www.sourceforge.net gehostet. Die neusten Versionen können dort bezogen werden. Darüber hinaus können auf der Projekthomepage www.spambuster.ch.vu weitere Informationen bezüglich Bug-Tracking, Feature Request und Dokumentation gefunden werden.

Lizenz

SPAM-Buster unterliegt der GNU General Public License:

```
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. The name of the of SPAM-Buster's contributors should not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Einleitung

Thematik

Im Zeitalter der Informationstechnologie gewinnen elektronische Medien mehr und mehr an Bedeutung. Wichtigstes Medium ist mit Abstand E-Mail. Über dem schnellen, effizienten und kostengünstigem Kommunikationsmedium hängt jedoch ein dunkler Schatten: SPAM.

SPAM ist der elektronische Massenversand von E-Mails. Millionen von E-Mails fluten stündlich die elektronischen Briefkästen und verursachen bereits heute enorme Kosten in Unternehmungen und sorgen für entnervte Privatanwender.

Historisches

SPAM ist ein Frühstücksfleisch der Firma Hormel und wurde durch einen Sketch im "Flying Circus" der britischen Komiker von Monty Python berühmt:

Mr. und Mrs. Bun betreten ein Restaurant und wollen etwas zu Essen bestellen. Doch zum Missfallen von Mrs. Bun wird jedes einzelne Gericht, selbst der Hummer, mit SPAM serviert.

Die Situation eskaliert, in der für Python üblichen Art ohne Pointe, als die Wikinger am Nebentisch alle anderen Darsteller mit dem "SPAM-Song" übertönen: "Spam. Spam. Spam. Lovely Spam! Wonderful Spam!"

Diesem Sketch verdankt das Wort SPAM seine Verwendung im Internet als Begriff für alle unerbetenen Werbe-E-Mails. Wie bei Python gibt es auch im E-Mail-Verkehr kaum eine Rettung vor SPAM, der mittlerweile zwischen 20 und 30 Prozent aller weltweit versendeten E-Mails ausmacht. Die Fach-Bezeichnungen für unerwünschte E-Mails sind UCE ("Unsolicited Commercial Electronic Mail", unerbetene, kommerzielle E-Mails) und UBE ("Unsolicited Bulk E-Mail", unerbetene Massen-E-Mails).

So witzig der Python-Sketch auch sein mag, in der Realität ist SPAM keinesfalls "wunderbar", sondern einfach nur ärgerlich und kann im schlimmsten Fall dazu führen, dass ein Postfach überläuft und nicht mehr erreichbar ist. Das Problem: Während ein regulärer Postbrief Porto kostet, sind E-Mails für die Versender fast umsonst. Deshalb werden so viele Werbenachrichten verschickt, egal an wen.

Dass das Spammen überhaupt so erfolgreich werden konnte, liegt am E-Mail-Boom der letzten Jahre. Die Einführung des World Wide Web und die immer schnelleren und billigeren Online-Zugänge bringen immer mehr Menschen ins Netz. Die Weiterentwicklung machte es möglich, HTML-E-Mails mit Bildern oder Dateianhängen zu versenden. Heute ist die E-Mail sowohl im privaten als auch im geschäftlichen Bereich nicht mehr wegzudenken. Und das bedeutet leider allzu oft nichts als zeitraubenden und nervtötenden Werbemüll.

Warum wird gespamt?

SPAM bietet gegenüber konventioneller Werbung enorm tiefe Kosten. Der Versand von einer Million E-Mails kostet nicht mehr als hundert Dollar. Jedoch ist die so genannte „response rate“ also die Antwortrate verglichen zu herkömmlicher Werbung sehr gering. Sie beträgt lediglich 15 pro Million versendeter E-Mails, verglichen mit rund 3000 pro Million versandter Kataloge. Trotzdem lohnt sich spammen.

Fighting Back!

Wenn es möglich wird die Gewinnspanne, welche sich mit SPAM eröffnet, zu verringern, wird es möglich sein SPAM den gar aus zu machen. Spammer sind Geschäftsleute, und gespammt wird, weil es sich als rentabel erweist.

Spammen ist jedoch nicht ganz gratis! Wenn es gelingt, die Anwortrate auf SPAM Mails zu verringern, wird unweigerlich die Gewinnspanne geringer und spammen wird somit ein unattraktives Geschäft.

Lösung

SPAM zu stoppen ist möglich und Inhalts-basierte Filter sind die Lösung! Die Achilles Ferse von Spammer ist ihre Nachricht. Sie können jede Barriere, die wir aufbauen, umgehen. Aber was sie auch anstellen um Filter zu umgehen, sie müssen ihre Nachricht an den Mann bringen. Existiert Software, die Nachrichten erkennt, besteht für Spammer keinerlei Möglichkeit mehr Drumherum zu kommen.

Für den Empfänger ist SPAM einfach zu erkennen. Warum sollte diese Aufgabe also nicht mit Hilfe von Künstlicher Intelligenz lösbar sein?

Algorithmus von Bayes

Ziel

Ziel des Algorithmus ist es, eine Nachricht als SPAM oder nicht SPAM zu klassifizieren.

Dazu wird für jede E-Mail die SPAM-Wahrscheinlichkeit, wie nachfolgend beschrieben, berechnet.

Bayes

SPAM-Buster verwendet eine leicht modifizierte Version des Bayes Algorithmus, welcher erstmals von Paul Graham umgesetzt wurde. Es handelt sich hierbei um einen statistischen Lösungsansatz. In einem gut trainierten System ist es möglich damit 995 von 1000 SPAM Mails zu erkennen bei 0 sogenannten „false-positives"!

„False-positives" sind legitime E-Mails welche fälschlicherweise als SPAM erkannt wurden. Für die meisten Nutzer ist das Löschen von legitimen Nachrichten erheblich schlimmer als SPAM zu erhalten. Deshalb muss die „false-positive" Rate unter allen Umständen so gering wie möglich gehalten werden!

Ablauf

Der Algorithmus besteht aus zwei unterschiedlichen Phasen:

1. Offline Trainingsphase
   Hierbei wird das System mit Daten trainiert um später E-Mails richtig klassifizieren zu können. Dazu verwendet man einen Block SPAM-Mails und einen weiteren Block von legitimen Mails. Basierend auf diesen Trainingsdaten ist das System später im Stande neu ankommende Mails korrekt zu klassifizieren.
2. Online-Klassifikation
   Für eingehende E-Mail Nachrichten wird deren SPAM-Wahrscheinlichkeit basierend auf den Trainingsdaten berechnet.

Trainingsphase

In der Trainingsphase wird das System mit zwei Blöcken an E-Mails trainiert. Einem Block an SPAM Nachrichten und einem anderen Block korrekter E-Mails. Für optimale Resultate sollten Blöcke von rund 4000+ E-Mails verwendet werden.

Zuerst wird der entsprechende Block an Nachrichten gescannt und in einen einzigen langen String gepackt. Berücksichtigt wird jeweils die komplete Nachricht, Header sowie sämtlich enthaltener HTML Code (Header und HTML Code enthalten meist besonders verräterische Merkmale). Die Nachrichtenkette wird in einzelne Tokens zerlegt. Als Token werden Alphanumerische Zeichen, Bindestriche, Apostrophe und Dollar Zeichen verwendet. Alle anderen Zeichen werden als Token Separatoren angesehen. Zahlen werden ignoriert.

Anschliessend wird die Anzahl jedes Tokens in einer TreeMap festgehalten. Es bestehen zwei verschiedene TreeMaps jeweils für Tokens aus dem SPAM Block und solchen aus dem Non-SPAM Block.

In einem weiteren Schritt wird für jedes Token die jeweilige SPAM-Zugehörigkeit berech-

net. Berücksichtigt dabei werden die Anzahl des Tokens aus dem SPAM-Block, die Anzahl aus dem Non-SPAM Block sowie die Anzahl an entsprechenden Nachrichten. Dabei werden Non-SPAM Token doppelt gewichtet.

Abbildung 1: LISP Algorithmus nach Paul Graham

```
(let ((g (* 2 (or (gethash word good) 0)))
      (b (or (gethash word bad) 0)))
   (unless (< (+ g b) 5)
     (max .01
          (min .99 (float (/ (min 1 (/ b nbad))
                             (+ (min 1 (/ g ngood))
                                (min 1 (/ b nbad)))))))))
```

Basierend auf der soeben erstellen Liste ist es Möglich neue E-Mails korrekt zu klassifizieren.

Klassifikation

Wenn eine neue Nachricht eintrifft wird diese ebenfalls in Token zerlegt. Anschliessend werden die 15 interessantesten Tokens weiterverwendet. Interessant wird hier an der Grösse der Abweichung von dem neutralen Wert 0.5 gemessen. Diese 15 Tokens mit ihren individuellen SPAM-Wahrscheinlichkeiten werden nun zu einer gemeinsamen Wahrscheinlichkeit kombiniert. Falls ein unbekanntes Token auftritt wird diesem der Wert 0.4 zugewiesen. Dieser Wert scheint vernünftig, da neue Tokens meist in eher ungefährlichen E-Mails auftreten. SPAM hat meist einen sehr eingeschränkten Wortumfang.

Liegt die Gesamtwahrscheinlichkeit über 0.7 handelt es sich bei der untersuchten Nachricht mit grosser Wahrscheinlichkeit um SPAM. (In einem sehr gut trainiertem System kann dieser Wert auch höher angesetzt werden).

DNS Blocklist

Was ist DNSBL?

Eine DNSBL ist eine DNS-basierende Blockliste, welche Hosts sowie IP Adressen enthält, welche für SPAM Zwecke genutzt werden.

SPAM-Buster verwendet eine derartige Blacklist als einen erster Filter. Dieser hat zum Zweck, den grössten Anteil (etwa 50%) an SPAM herauszufiltern.

Funktionsweise

In einem ersten Schritt werden IP-Adressen aus den Header Informationen herausgefiltert. Anschliessend werden diese Adressen bei einem Realtime Blackhole List Anbieter überprüft.

IP Adressen der Form a.b.c.d werden in der Form d.c.b.a.rbl an den jeweiligen Anbieter gesendet. Dabei können folgende Rückgabewerte unterschieden werden:

127.0.0.2	„Verified Open Relay“
127.0.0.3	„Dialup Spam Source“
127.0.0.4	Confirmed Spam Source
127.0.0.5	Smart Host
127.0.0.6	Spamware Developer
127.0.0.7	List Server
127.0.0.8	Insecure Formmail Script
127.0.0.9	Open Proxy Server

Bei der jetzigen Implementation wird falls die IP Adresse einen Rückgabewert liefert das Mail als SPAM gekennzeichnet.

Bayes Algorithmus von Robinson

Über den Author

„I've been in the software business for about 25 years. A few years back I made enough of a profit that I could personally fund Transpose, LLC. So that's what I'm working on now; I'm CEO.“

Weitere Informationen über Garry Robinson finden sich auf seiner Homepage: http://www.garyrobinson.net/

Ziel

Ziel des Algorithmus ist es, eine Nachricht als SPAM oder nicht SPAM zu klassifizieren.

Dazu wird für jede E-Mail die SPAM-Wahrscheinlichkeit, wie nachfolgend beschrieben, berechnet.

Nachteile des vorherigen Algorithmus

Der oben beschriebene Bayes Algorithmus wurde bereits sehr bekannt um Spam zu filtern. Robinson beschreibt jedoch einen Weg um den oben beschriebenen Algorithmus zu verbessern.

Der Algorithmus basiert auf der Annahme dass die Wahrscheinlichkeiten unabhängig sind. Das sind sie jedoch nicht wenn man über Wörter in Emails spricht.
Zudem ist der Algorithmus asymmetrisch in Umgang mit Wörtern die “spaminess” indizieren im Vergleich zu Wörtern die “non- spamminess” indizieren. Er ist nicht sehr empfindlich in der Unterscheidung von spamminess / non-spamminess. Einige Leute denken dies ist besser um „false positive“ zu reduzieren.

Der obige Algorithmus generiert extreme Zahlen für beide Klassifikationen, sodass kein Feintuning mehr möglich ist. Dies währe aber ein gewolltes Feature.

Implementation des Algorithmus

Die Spamprobability für ein Wort heissen p1, p2, …, pn. Für ein Mail gibt es n Wörter und n assozierte “probabilities”. Jedes indiziert die Wahrscheinlichkeit dass das Mail ein Spammail ist.

Ein Wert nahe 1 indiziert spaminess und ein Wert nahe 0 indiziert non- spaminess.

Bis hier funktioniert alles noch genau so wie im vorherigen Algorithmus.

Jetzt berechnen wir:

P = 1 – ((1-p1)*(1-p2)*…*(1-pn))^(1/n) [spamminess]
Q = 1 – (p1*p2*…*pn)^(1/n) [non-spamminess]

S = (P – Q) / (P + Q) [combined indicator]

(^(1/n) sollte als n-te Wurzel gelesen werden)

S ist die Zahl zwischen -1 und 1. Hohe Zahlen bedeuten Spam, Tiefe Zahlen bedeuten kein Spam. 0 bedeutet, es gibt Beweise für beide Arten von Mail.

Wenn man lieber eine normierte Zahl zwischen 0 und 1 hat, kann man mit der Formel (1 + S) / 2 das Resultat normieren.

Nun muss nur noch der Grenzwert für Spam / non- Spam gewählt werden. Wir haben diesen auf >= 0 gesetzt. Dies bedeutet jedes Mail welches ein grösseres oder gleiches S als 0 hat wird als non- Spam erkannt. Dieser Wert hat sich in Tests als effizientester Wert herausgestellt.

Architektur

Anforderungen

Aufgrund der Aufgabenstellung und dem Einsatzgebiet von SPAM-Buster ergaben sich folgende Anforderungen an die Ziel-Anwendung:

- Plattform Unabhängigkeit
- Verwendung auf Client- sowie Server Systemen
- Unterstützung gängigster Protokolle für E-Mail Bezug: Post Office Protocol (POP) und Internet Message Access Protocol (IMAP)
- Verwendbarkeit bestehenden E-Mail Clients (Microsoft Outlook, Microsoft Outlook Express, Mozilla, Thunderbird, Netscape, etc.)
- Einfache Erweiterbarkeit der Anwendung
- „Plug and Play“ Mechanismus für verschiedene Algorithmen
- Einfache Konfiguration und Verwendbarkeit der Applikation

Als Basisapplikation für die Entwicklung von SPAM-Buster zeigte sich JFetch als sehr robuste und komfortable Lösung. JFetch ist das Java Pendant zu dem bekannten UNIX Programm „fetchmail“, welches es ermöglicht, Emails von verschiedensten Mail Servern abzuholen und lokal zuzustellen.

JFetch

JFetch ist eine Anwendung, welche es erlaubt E-Mail Nachrichten mittels POP oder IMAP Protokoll herunter zu laden. Es besteht sogar die Möglichkeit News-Nachrichten abzuholen mittels NNTP. Darüber hinaus verfügt JFetch über ein ausgeklügeltes Filter-Konzept. Die Filterarchitektur von JFetch ist eventbasiert aufgebaut. Ein Filter hat die Aufgabe zu Entscheiden, ob eine Nachricht herunter geladen wird oder nicht. Verschiedene Filter werden sequentiell durchlaufen. Dabei kann zwischen global Filtern oder Filtern per Maildrop unterschieden werden. Eine Maildrop repräsentiert ein E-Mail Konto, von welchem E-Mail Nachrichten bezogen werden.

Durch Implementation bestimmter Schnittstellen ermöglicht es dem Entwickler auf einfache Art und Weise neue Filter ins System zu integrieren. Filter können verschiedenste Aufgaben übernehmen. Beispielsweise sind dies:

- Mailgrössen Filterung
- Whitelist oder Blacklist Filterung
- SPAM Filterung
- etc

JFetch lädt Nachrichten, die den angegeben Kriterien entsprechen herunter und stellt diese einem Message Distribution Agent zu. Unterstützt werden mailbox und SMTP Ser-

ver.

Die Übersichtliche Konfiguration mittels XML und die saubere Implementierung sprechen für den Einsatz von JFetch als Basis-Applikation.

SPAM-Buster

SPAM-Buster erweitert JFetch-Filtersystem mit einem SPAM Algorithmus von Bayes, welcher erstmals von Paul Graham in dieser Weise umgesetzt wurde.

Darüber hinaus verfügt SPAM-Buster über ein leicht verwendbares Graphical User Interface (GUI). Dies erleichtert den Umgang mit der Applikation für grafische Systeme. Selbstverständlich kann SPAM-Buster auch auf nicht-grafischen Systemen eingesetzt werden. Dies ist insbesondere bei Server-Systemen von Nutzen.

Der vielfältige Einsatzbereich, die besonders einfache Handhabung und die leistungsstarke und stabile Architektur sprechen für den Einsatz von SPAM-Buster im End-User sowie Server Bereich.

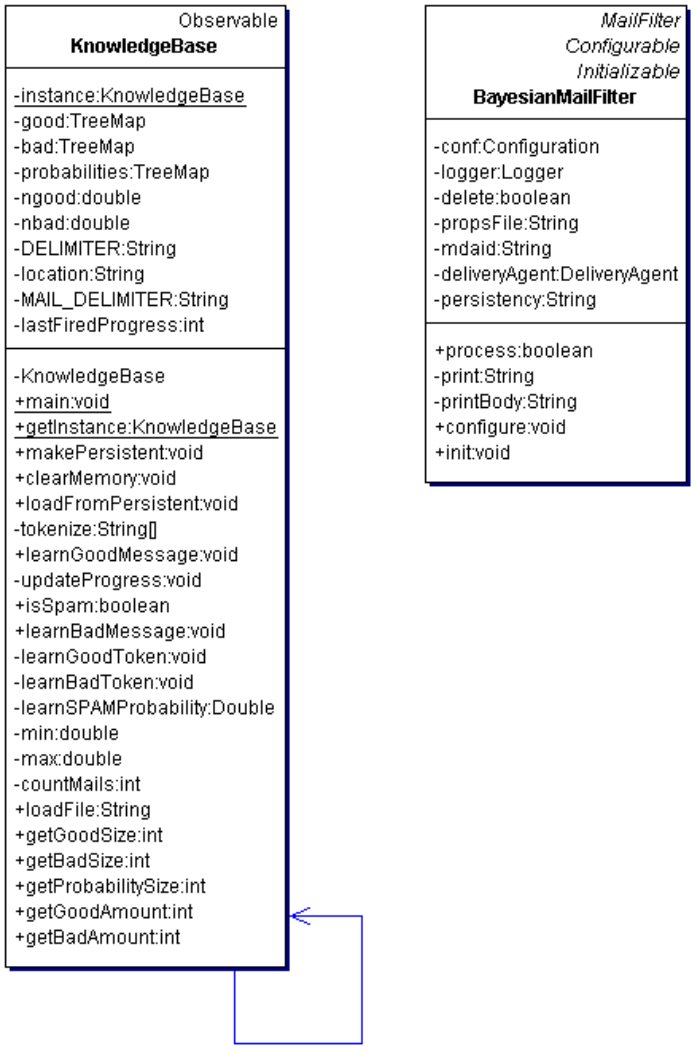
Design

Filter

Der BayesianMailFilter, welcher die Schnittstelle zur JFetch-Architektur darstellt, besteht aus drei wesentlichen Teilbereichen:

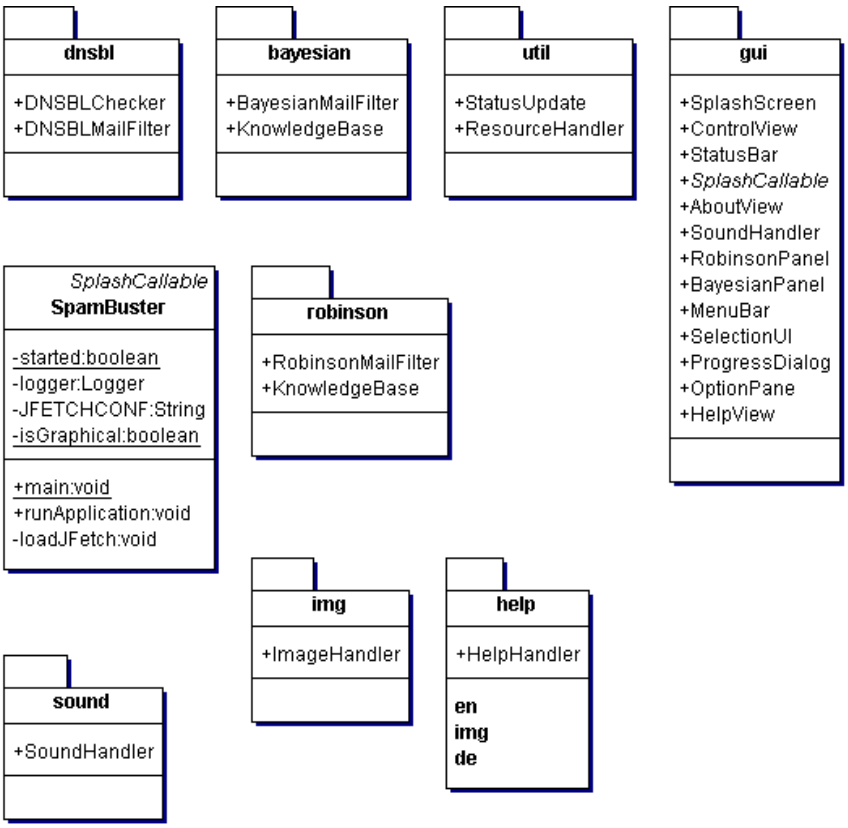
- Initialisierung
- Message Processing
- Konfiguration

Diese Bereiche widerspiegeln sich auch im Konzept:

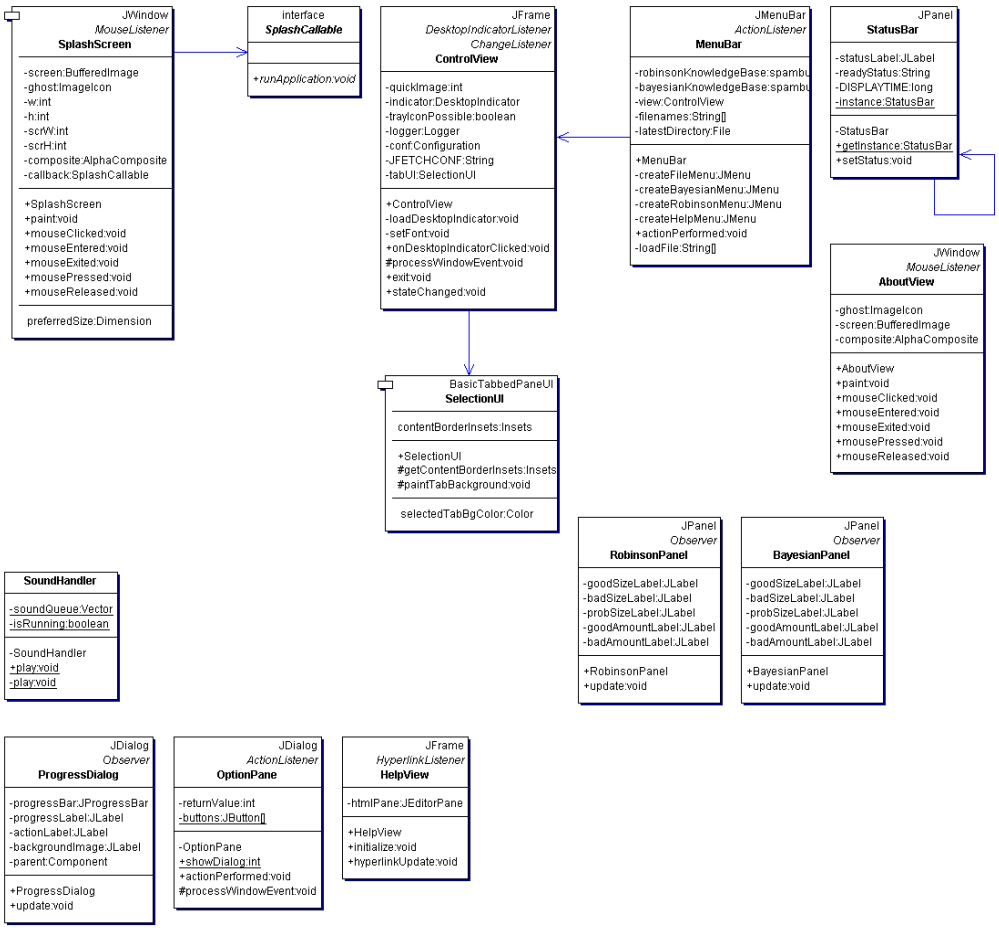


Im Teilbereich Initialisierung werden die grundlegenden Eigenschaften für die Klasse festgelegt. Dies sind insbesondere das lokalisieren des SPAM Delivery Agent, initialisieren des Logging Kit und dem laden der Wissensdatenbank.

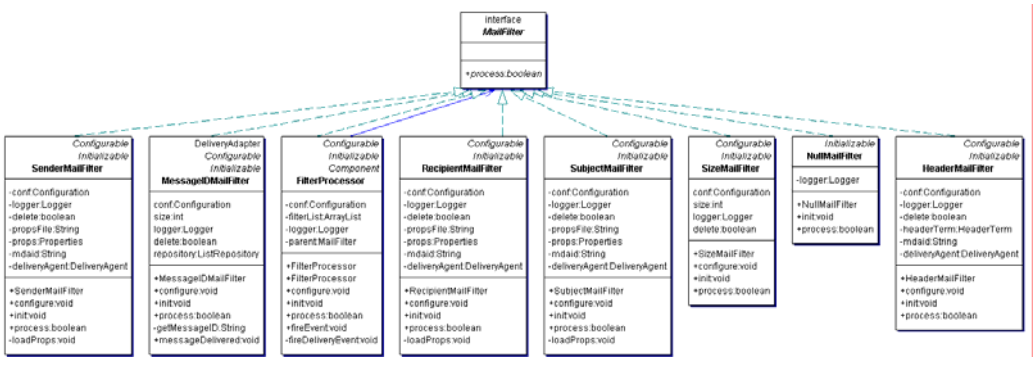
Packages


dnsbl
+DNSBLChecker
+DNSBLMailFilter
bayesian
+BayesianMailFilter
+KnowledgeBase
util
+StatusUpdate
+ResourceHandler
gui
+SplashScreen
+ControlView
+StatusBar
+SplashCallable
+AboutView
+SoundHandler
+RobinsonPanel
+BayesianPanel
+MenuBar
+SelectionUI
+ProgressDialog
+OptionPane
+HelpView
SplashCallable
SpamBuster
-started:boolean
-logger:Logger
-JFETCHCONF:String
-isGraphical:boolean
+main:void
+runApplication:void
-loadJFetch:void
robinson
+RobinsonMailFilter
+KnowledgeBase
img
+ImageHandler
help
+HelpHandler
en
img
de
sound
+SoundHandler

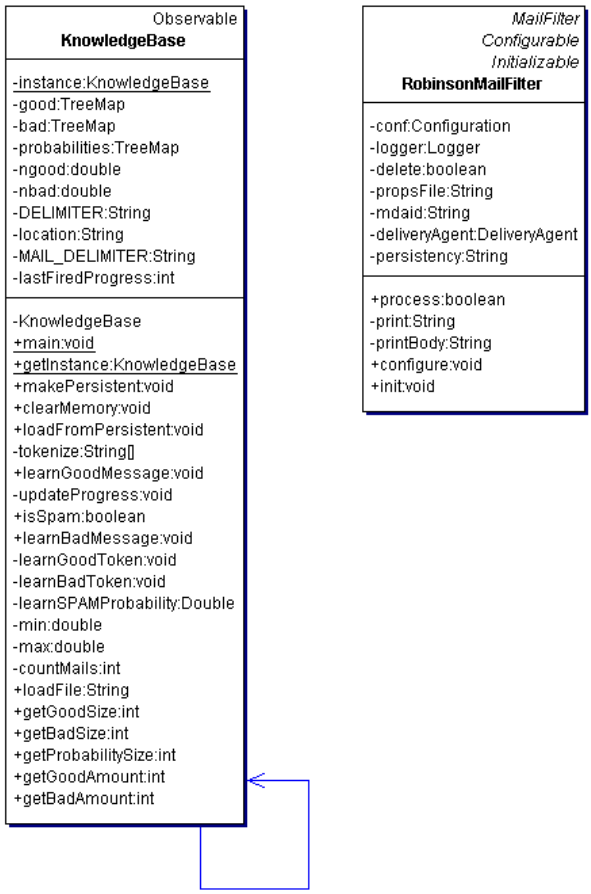
Subpackages *spambuster.gui*



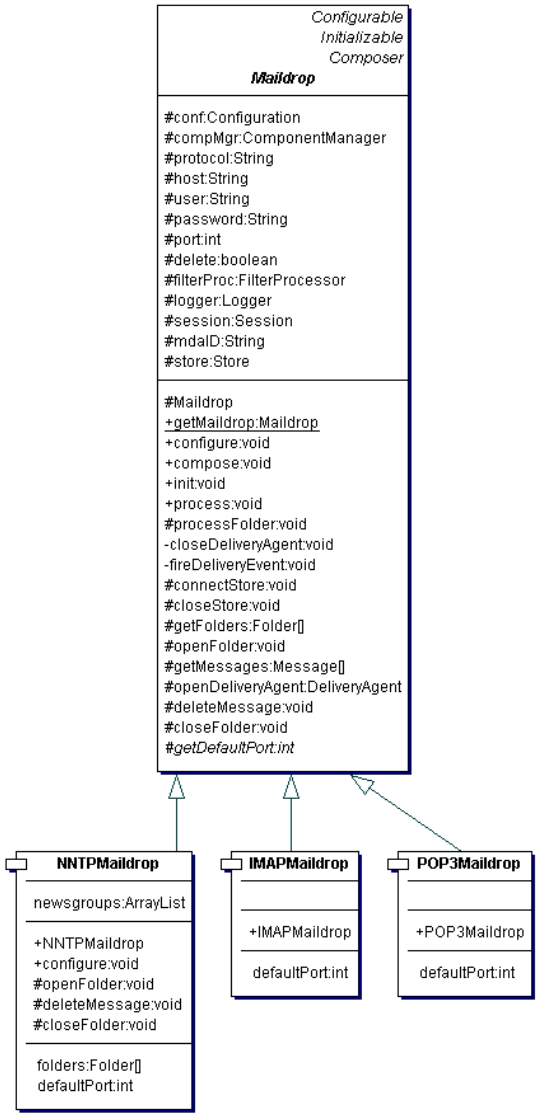
spambuster.filters



spambuster.robinson



spambuster.core

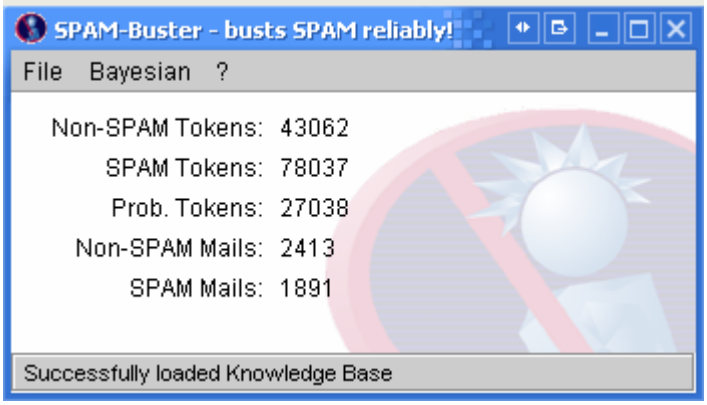


Darstellung

UserInterface

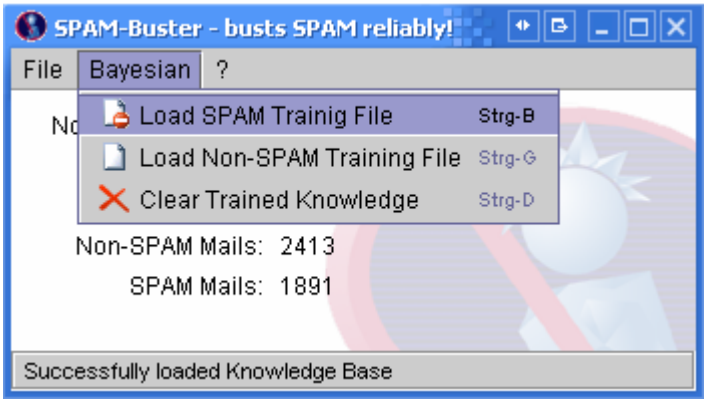
SPAM-Buster verfügt über ein User Interface, welches ermöglich auf einfache und bequeme Weise Einstellungen vorzunehmen. Bei der Gestaltung wurde bewusst auf Simplizität und Komfort der Applikation geachtet. Da die grafische Darstellung vorwiegend im Client bereich zum Einsatz kommt, ist diese eher weniger versierten Benutzern vorbehalten.

Abbildung 2: Übersicht der Benutzerschnittstelle



Es stehen dem Benutzer einige wichtige Funktionen für das Trainieren des Systems zur Verfügung. Dies sind insbesondere das Trainieren des Systems mit SPAM bzw. Nicht-SPAM sowie das Löschen der Wissensdatenbank. Darüber hinaus steht dem Benutzer eine Online-Hilfe zur Verfügung.

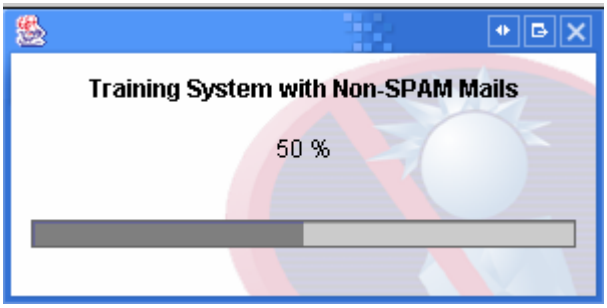
Abbildung 3: Menu Funktionen von SPAM-Buster



Feedback

Um den Benutzer angemessen über den Status der Applikation zu informieren wurde speziellen Wert auf System-Feedback gelegt. Das System gibt über allfällige Aktionen ständig Rückmeldung. Dies ist insbesondere bei langwierigen Operationen wie dem Trainieren des Systems von Wichtigkeit.

Abbildung 4: Benutzer Feedback



Diskretion

SPAM-Buster wurde so konzipiert, dass die Anwendung grösstenteils selbstständig im Hintergrund arbeitet. Aufgrund dessen wäre ein ständig geöffnetes Fenster für den Benutzer und seinen Arbeitsfluss störend und somit unangebracht. Deshalb wird das Fenster von SPAM-Buster beim Minimieren geschlossen und die Anwendung als sogenanntes „Tray Icon“ angezeigt. Aufgrund des Einsatzes einer Native Schnittstelle ist dieses Feature lediglich WIN32 Systemen vorbehalten.

Abbildung 5: SPAM-Buster als Tray Icon

Installation

Mail Server

Für den Einsatz von SPAM-Buster muss vorgängig ein Mail Server installiert werden. Dies kann sowohl lokal als auch serverseitig geschehen.

Im folgenden wird der Installationsvorgang für ein Client System beschrieben:

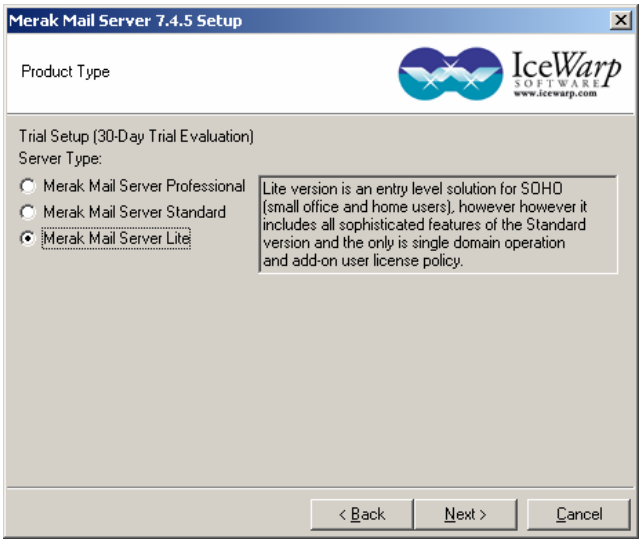
Als Mail Server wird der Merak Mail Server verwendet. Dieser kann unter http://www.merakmailserver.com bezogen werden. Dieser ist kostenlos als Trial Version für 30 Tage nutzbar. Natürlich kann jeder beliebige Mail Server verwendet werden.

Füllen sie die Eingabefelder aus:

Abbildung 6: Ausfüllen der Benutzerdaten

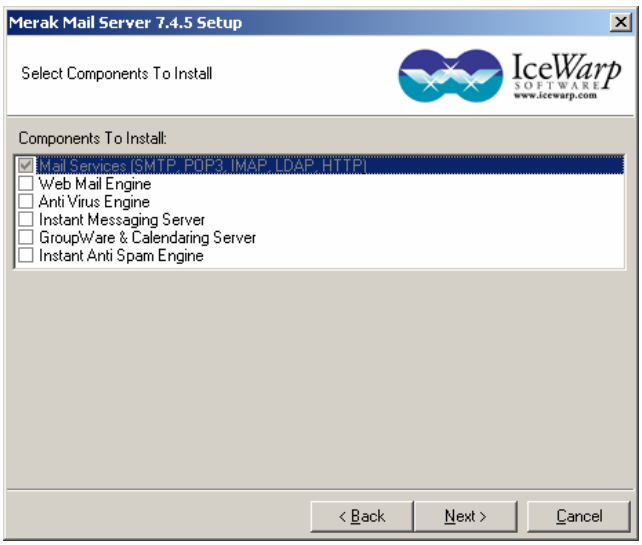
Wählen sie Merak Mail Server Lite:

Abbildung 7: Mail Server Produkt Typen



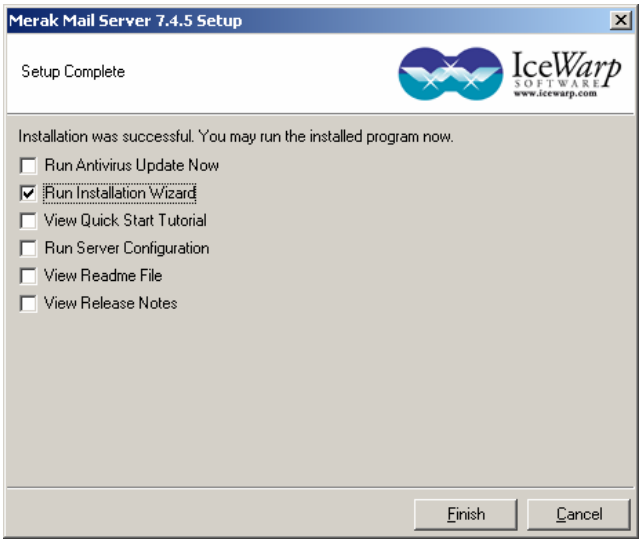
Deaktivieren sie sämtliche Optionen und starten sie anschliessend die Installation:

Abbildung 8: Merak Komponenten Auswahl



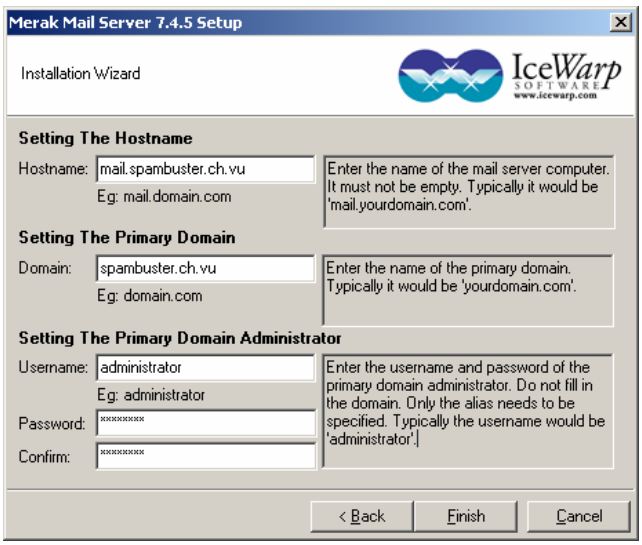
Wählen sie ledliglich die Option „Run Installation Wizard“ für abschliessende Einstellungen:

Abbildung 9: Merak Setup Abschliessende Optionen



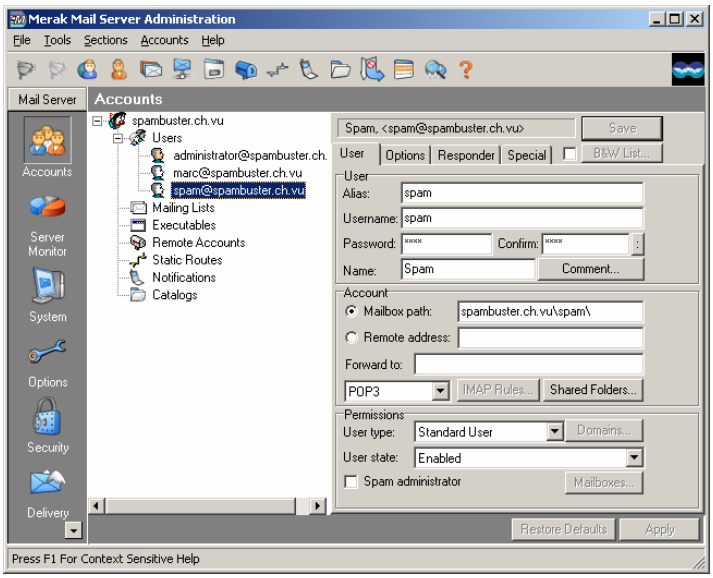
Füllen sie die Eingabefelder gemäss folgender Grafik aus:

Abbildung 10: Mail Server Einstellungen



Starten sie anschliessend die Merak Mail Server Administration und fügen sie die gewünschten Benutzer hinzu. Sie sollten mindestens 2 verschiedene Benutzer anlegen. Jeweils einen für nicht-SPAM Mails und einen, welcher später die SPAM Mails erhalten wird:

Abbildung 11: Mail Server Administration



POP-Server: localhost
Mail-Server: <mailserver>

Mail Client

Um auf die neu erstellten E-Mail Konten zugreifen zu können muss der lokale Mail Client (Bsp: Microsoft Outlook, Netscape, Thunderbird, …) konfiguriert werden. Es müssen lediglich zwei neue Anschliessend müssen sie lediglich im persönlichen Mail Client zwei neue E-Mail Konten mit folgenden Einstellungen erstellt werden:

Username:	<user>	(Benutzer der für nicht-SPAM Mails angegeben wurde)
Passwort:	<passwort>	(Passwort des Benutzers)
POP-Server:	localhost	

Mail-Server: <mailserver> (Ihr ISP Mail Server)

Username: spam
Passwort: <passwort>
POP-Server: localhost
Mail-Server: <mailserver> (Ihr ISP Mail Server)

SPAM-Buster

Für die Konfiguration von SPAM-Buster sei auf das nachfolgende Kapitel verwiesen.

Trainingsdaten vorbereiten

Um Spambuster effizient zu nutzen müssen möglichst viele Trainingsdaten vorhanden sein. Hier wird eine effiziente Methode beschrieben wie man eigene Trainingsdaten aus dem Outlook exportieren und in Spambuster verwenden kann.

Folgenden Programme werden benötigt:

- MessageSave (http://www.techhit.com/messagesave/)
- TXT Collector (http://bluefive.pair.com/index.php)

Als erstes müssen die Mails von Outlook in Textfiles gespeichert werden. Dabei unterstützt uns MessageSave. Das Programm findet sich nach dem installieren im Outlook als kleines Symbol:

Abbildung 12: MessageSave Icon um Nachrichten zu exportieren

Um die Mails zu speichern, einfach die entsprecheneden Mails in Outlook markieren und dann auf das MessageSave Symbol klicken und den Save- Pfad auswählen.

Die Mails werden von MessageSave als einzelne .txt- Files gespeichert. Diese müssen jetzt noch zu einem .txt File zusammengefügt werden. Hierbei unterstützt uns TXT Collector:

Abbildung 13: TXTcollector um Nachrichten zusammenzufügen

Die Einstellungen wie im obigen Screen vornehmen (Pfad anpassen) und auf "Combine all files" klicken. Nun sind alle Mails in einem File und können von Spambuster eingelesen werden.

Trainingsdaten einlesen

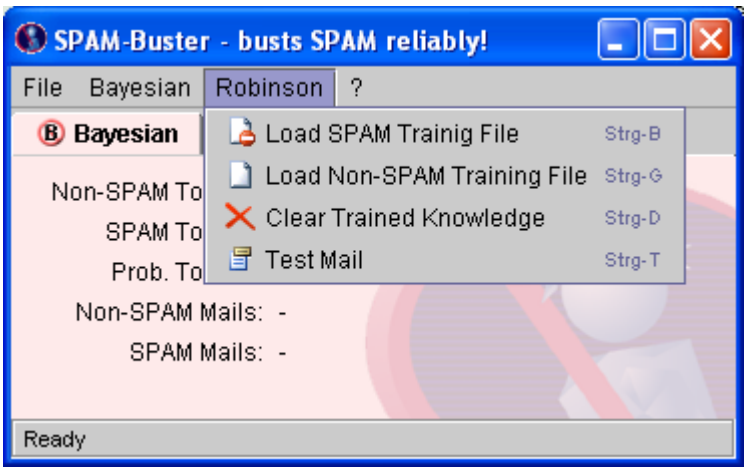
Die jeweiligen Antispam- Algorithmen können einzeln (unabhängig) trainiert werden.

Es werden zwei typen von Trainingsdaten unterschieden. Spam- Training, und non-Spam Training.

Beim Spam- Training werden dem Algorithmus Spam- Mails eingegeben um ihn auf das erkennen von Spam- Mails zu trainieren. Umgekehrt werden beim non- Spam Training Mails eingegeben welche kein Spam sind, um den Algorithmus zu Trainieren wie er non-Spam Mails erkennen kann.

Um einen Trainingsvorgang zu starten, einfach im entsprechenden Menu (welches dem Algorithmus entspricht), Spam, oder non- Spam Training auswählen:

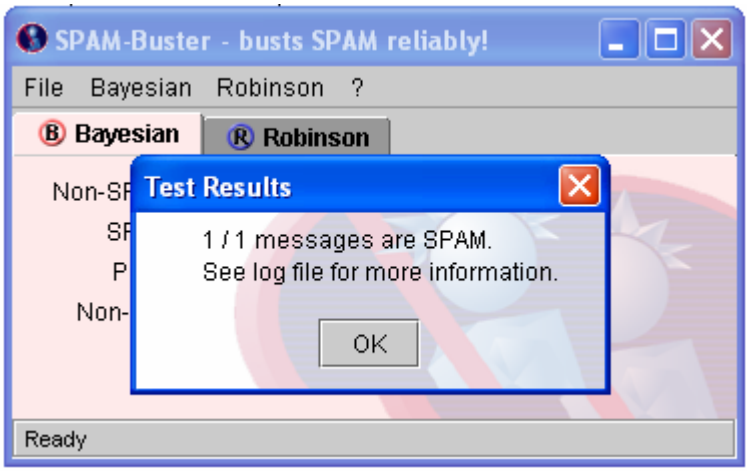
Abbildung 14: Trainieren des Systems



Nachrichten testen

Um die Anti- Spam Algorithmen zu testen bietet Spambuster die Möglichkeit ein oder mehrere Testmails von Textfiles zu laden und direkt auf Spam zu testen:

Abbildung 15: Anti-SPAM Algorithmen testen



Um einen Algorithmus zu testen, einfach im entsprechenden Menu "Test Mail" wählen. Somit wird nur der gewählte Algorithmus getestet.

Konfiguration

Einleitung

Für den Einsatz von SPAM-Buster müssen vorgängig die Einstellungen für Filter und JFetch korrekt festgelegt werden. Die Einstellungen befinden sich im conf-Verzeichnis von SPAM-Buster in der Datei „jfetch.xml“.

Maildrop

Eine Maildrop ist ein E-Mail Konto, von welchem E-Mail Nachrichten bezogen werden. Charakteristische Protokolle dafür sind POP, IMAP und NNTP. Für den Zugang zu solchen Konten sind typischerweise Benutzername, Passwort, Hostname und Port-Nummer anzugeben. Nachrichten, die von einer Maildrop bezogen werden, durchlaufen sequentiell die aufgelisteten lokalen Filter und werden anschliessend dem dazugehörigen Deliver Agent zugestellt.

Beispiel einer Maildrop Konfiguration:

```
<maildrop protocol="pop3" mda="smtp">
        <host>mail.popserver.ch</host>
        <port>110</port>
        <user>benutzername</user>
        <password>passwort</password>
        <delete>true</delete>

<!-- filters specific to this maildrop -->
        <filters>
        </filters>
</maildrop>
```

Es können mehrere Maildrops spezifiziert werden. Diese werden nacheinander abgearbeitet.

Filter

Filter sind der Kern der Anwendung. Ein Filter entscheidet per E-Mail, ob diese Nachricht herunter geladen wird oder nicht. Jede Nachricht kann eine Pipeline an Filtern durchlaufen. An jeder Stelle diese Pipeline kann ein Filter veranlassen, dass die Nachricht nicht weiter verarbeitet wird.

Es werden zwei verschiedene Filtertypen unterschieden: Globale und lokale. Lokale Filter werden per Maildrop assoziiert, globale werden auf sämtliche Maildrops angewendet.

Beispiel eines globalen Filters:

```
<filters>
        <!-- size filter -->
         <filter class="jfetch.filters.SizeMailFilter" max-
                size="1548576" delete="false">
        </filter>
</filters>
```

Lokale Filter werden innerhalb einer Maildrop festgelegt:

```
<maildrop protocol="pop3" mda="smtp">
        <!-- .... standard maildrop config goes here .... -->
```

```
        <!-- filters specific to this maildrop -->
        <filters>
        <!-- sender mail filter -->
                <filter class="jfetch.filters.SenderMailFilter"
                 delete="true" block-
                 list="/home/gautam/JFetch/spool/linuxlist"
                 mda="linux">
                 </filter>
        </filters>
</maildrop>
```

Delivery Agent

Ein Deliver Agent dient dem Zustellen von E-Mail Nachrichten. Zur Zeit unterstützt werden das SMTP Protokoll und mbox. Deliver Agents haben eine eindeutige id. Maildrops haben einen Standard Delivery Agent welchem die Nachricht zugestellt wird, falls diese die Filter erfolgreich durchläuft.

Einige Filter benötigen für ihre Konfiguration ebenfalls die Angabe eines Deliver Agents, so auch der BayesianMailFilter von SPAM-Buster. Falls eine Nachricht auf das Filterkriterium anspricht wird diese dem angegebenen Deliver Agent zugestellt. Auf diese Weise können SPAM-Mails an einen speziellen Delivery Agent zugestellt werden.

Beispiel eines SMTP Delivery Agent:

```
<mda class="jfetch.delivery.SMTPDeliveryAgent" id="smtp">
        <host>localhost.localdomain</host>
        <port>25</port>
        <localuser>gautam</localuser>
        <domain>localhost</domain>
        <user></user>
        <password></password>
</mda>
```

Beispiel eines mbox Delivery Agent

```
<mda class="jfetch.delivery.MailboxDeliveryAgent" id="junk">
        <destination>/home/gautam/Mail/junkmail</destination>
</mda>
```

Weitere Informationen

Diese Konfigurationshilfe soll lediglich einen groben Überblick, welcher für die Basis-Installation benötigt wird, vermitteln. Für weitere spezifische Informationen zur Konfiguration von JFetch sei hier auf die JFetch Konfiguration Dokumentation verwiesen. Diese befindet sich im doc Verzeichnis des Projektes.

Testresultate

Umgebung

SPAM-Buster wurde in folgender Testumgebung geprüft:

- Microsoft Windows XP SP1
- Microsoft Outlook 2003
- Merak Mailserver 7.4.5

Es wurden Mails von den folgenden E-Mail Konten ausgewertet:

- GMX.ch (2x)
- Bluewin.ch
- HSR.ch

Die unterschiedlich klassierten Nachrichten wurden entsprechend an zwei verschiedene Konten weitergeleitet und anschliessend mit dem Client E-Mail Programm gelesen.

Die folgenden Filter wurden jeweils mit 100 Nachrichten getestet wobei es sich um 50 SPAM sowie um weitere 50 Non-SPAM Mails handelte.

Resultate Bayes

Der Algorithmus von Bayes wurde mit folgender Trainingsstatistik getestet:

- SPAM-Mails: 53
- Non-SPAM-Mails: 2413

Die daraus resultierte Knowledgebase hatte folgende Ausmasse:

- Anzahl SPAM Tokens: 4249
- Anzahl Non-SPAM Tokens: 43062
- Anzahl Probabilities: 12198

Die Testnachrichten wurde folgendermassen klassifiziert:

- Getestete Nachrichten: 100
- Als SPAM klassifiziert: 47
- Als Nicht-SPAM klassifiziert: 53
- False Positives: 0

- False Negatives: 3

Resultate Robinson

Der Algorithmus von Gary Robinson wurde mit folgender Trainingsstatistik getestet:

- SPAM-Mails: 53
- Non-SPAM-Mails: 2413

Die daraus resultierte Knowledgebase hatte folgende Ausmasse:

- Anzahl SPAM Tokens: 4249
- Anzahl Non-SPAM Tokens: 43062
- Anzahl Probabilities: 12198

Die Testnachrichten wurde folgendermassen klassifiziert:

- Getestete Nachrichten: 100
- Als SPAM klassifiziert: 48
- Als Nicht-SPAM klassifiziert: 52
- False Positives:1
- False Negatives:3