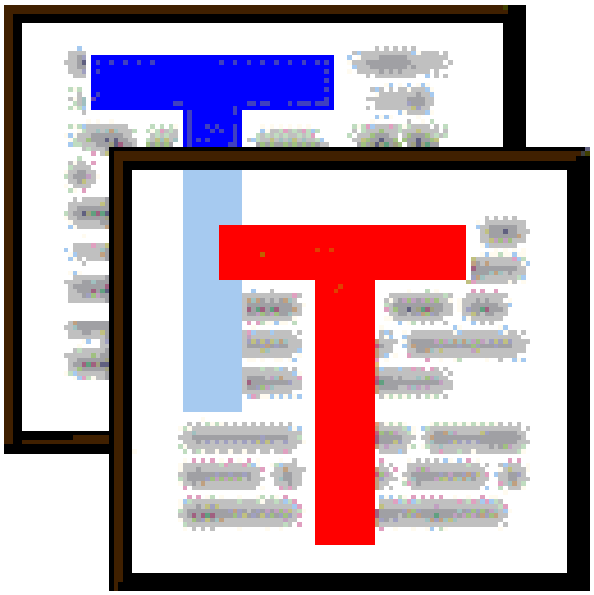
IMP Filter



1 Einleitung

Der IMP-Filter ist Plugin für die Antispam-Software Spamihilator:

http://www.spamihilator.com

KURZBESCHREIBUNG:

Mit dem IMP-Filter werden E-Mails daraufhin untersucht, ob es sich um Spam handelt oder nicht. Die Kriterien dieser Klassifizierung sind nicht vorgegeben, sondern müssen in Form eines TextTransformer-Projekts vom Benutzer selbst formuliert werden. Für die Erstellung solcher Projekte reicht die freie Version des
TextTransformer-Programms aus. Texte können mit solchen Projekten nahezu beliebig genau analysiert werden. Der Aufwand für die Anfertigung solcher Projekte lohnt sich allerdings vielleicht nur, wenn der Spam-Charakter von Mails in bestimmten Fällen sicher klassifiziert werden muss und sich gezeigt hat, dass die statistischen Lernergebnisse zu viele falsche Ergebnisse hervorbringt. Was der IMP-Filter wirklich zu leisten vermag, muss sich noch herausstellen.

WORTBEDEUTUNG

"IMP" ist ein Akronym für Individueller-Mail-Parser und "IMP" ist zugleich ein englisches Wort, für "Teufelchen".

2 De-/Installation

Die **Installation** des IMP-Filters setzt die Installation des Spamihilators voraus und erfolgt über die IMPInstall.exe.

Die **Deinstallation** sollte über die Systemsteuerung/Software des Betriebssystems erfolgen. Zuvor muss der Spamihilator beendet werden, da die geladenen dll's sonst nicht entfernt werden.

Die Pfade werden automatisch an die des Spamihilators angepasst.

Nach der Installation des IMP-Filters tut dieser zunächst nichts. Erst, wenn der Benutzer ein TextTransformer Projekt auswählt, tritt der Filter in Aktion.

Die Position in der Liste der Spamihilator-Plugins sollte an das Projekt angepasst werden..

2.1 Pfade und Dateien

Die Dateipfade werden automatisch an die des Sapmihilators angepasst.

Programm-Verzeichnis

Das eigentliche Plugin, die *impfilter.dll* wird im Unterverzeichnis plugins des Programmverzeichnisses installiert:

```
C:\Programme\Spamihilator\plugins
```

Anwendungsdaten-Verzeichnis

Die Projektbeispiele, diese Hilfe und eine language-Datei werden bei der Installation zunächst in ein Unterverzeichnis des Programm-Verzeichnisses geschrieben.
Je nach Betriebssystem und Art der Spamihilator-Installation werden sie von dort beim ersten Aufruf des Plugins in ein Anwendungsdaten-Verzeichnis des jeweiligen Benutzers kopiert. Der Name solcher Anwendungsdaten-Verzeichnisse wird durch Anhängen eines Versionskürzels an den Namen "impfilter" gebildet. Z.B. für die Version 0.7.0 ergibt sich so der Name:

```
...\impfilter070
```

Diese Verzeichnisse müssen bei der Deinstallation des IMP-Filters von Hand gelöscht werden.

Wenn den Vorschlägen bei der Installation des Spamihilators gefolgt wurde ergeben sich folgende Verzeichnisse:

Wenn bei der Installation von Spamihilator „Getrennte Einstellungen“ gewählt wurde (empfohlen, Standard-Methode):

Windows 2000/XP:

```
C:\Dokumente und Einstellungen\Benutzername\Anwendungsdaten\Spamihilator\plugins\impfilterXXX
```

Windows Vista:

```
C:\Benutzer\Benutzername\AppData\Roaming\Spamihilator\plugins\impfilterXXX
```

Wenn bei der Installation „Gemeinsame Einstellungen für alle Benutzer“ gewählt wurde (alte Methode) werden die Daten nicht in ein gesondertes Anwendungsdaten-Verzeichnis kopiert:

```
C:\Programme\Spamihilator\plugins\impfilter
```

Vor der Installation eines Updates des IMP-Filters müssen in diesem Verzeichnis geänderte Projekte von Hand gesichert werden.

System-Verzeichnis

Im Systemverzeichnis, z.B.:

```
C:\WINNT\system32
```

wird der Parserinterpreter *imp_engine.dl* installiert, sowie einige weitere dll's, die von dem Interpreter benötigt werden. Die letzteren werden mit dem TextTransformer geminsam benutzt.

Für Experten hier die vollständige Liste:

```
imp_engine.dll
imp_cppitp.dll
TTXercesLib.dll
```

```
boost_regex-bcb-mt-1_34_1.dll
stlpmt45.dll
cc3260mt.dll
rtl60.bpl
vcl60.bpl
```

2.2 Priorität

Es wird empfohlen, den IMP-Filter gleich an zweiter Stelle nach dem Newsletter-Plugin zu plazieren, wenn Sie Ihren individuellen Mail-Parser auf die unten beschriebene Weise konstruiert haben.

Hier wird in der Spamihilator Online-Hilfe beschrieben, wie diese Einstellung vorzunehmen ist:

http://wiki.spamihilator.com/doku.php?id=de:configpriorities

Es hängt natürlich von der Art des Projektes ab, wo der IMP-Filter in der Liste der Spamihilator-Filter positioniert werden sollte. Die besondere Stärke des IMP-Filters ist aber, dass Sie damit Ihren individuellen Mail-Parser so konstruieren können, dass damit für bestimmte Mails sehr sicher entschieden werden kann, ob es sich um Spam oder Nicht-Spam handelt. Daher ist es sinnvoll den IMP-Filter relativ weit oben in der Liste zu plazieren. Erwünschte Mails sollten nicht von anderen Filtern eliminiert werden und unerwünschte Mails nicht als Nicht-Spam von der weiteren Analyse ausgeschlossen werden, bevor es überhaupt zu einem Test mit dem IMP-Filter kommt. Die Fälle, bei denen der IMP-Filter nicht sicher entscheiden kann, ob es sich um Spam oder Nicht-Spam handelt, sollten jedoch konsequent an die anderen Filter weitergeleitet werden.

3 Bekannte Fehler

Der IMP-Filter verträgt sich nicht gut mit dem **Blacklist-Filter**. Wenn in den Spamihilator Einstellungen unter Plugins auf *Neuladen* und dann auf *OK* geklickt wird stürzt der Spamihilator ab. Sonst gibt es keine bekannten Probleme.
Der Blacklist-Filter gilt nun ohnehin als obsolet.

Wenn versucht wird mit einer 0.7'er Version des IMP-Filters ein Projekt zu laden, das mit einer neueren Version des TextTransformers als 1.6.1 erstellt wurde, stürzt der Spamihilator ab.

4 TextTransformer

Der IMP-Filter benötigt ein vom Benutzer individuell an seine Bedürfnisse angepasstes TextTransformer-Projekt. Im Anwendungsdaten-Verzeichnis gibt es einige Beispiele, die als Basis für eigene Entwicklungen verwendet werden können. Zur Modifikation dieses Projektes ist die Installation des TextTransformer-Programms erforderlich. Es kann unter folgendem Link heruntergeladen werden:

http://www.texttransformer.de/Downloads_ge.html

Die freie Version des TextTransformers reicht für den Einsatz mit dem Spamihilator normalerweise aus.

5 Texte parsen

Mit dem IMP-Filter werden die Texte von E-Mails analysiert, man sagt auch, die Texte werden geparst. Das soll am Beispiel einer typischen Mailstruktur veranschaulicht werden. Je nach Struktur werden die Texte in Spam oder Nicht-Spam klassifiziert. Wenn die Analyse scheitert, muss der Parserfehler behandelt werden.

5.1 Zur Begrifflichkeit

Struktur einer E-Mail

Es gibt verschiedene Arten, wie E-Mails aufgebaut sein können, aber allen Arten gemeinsam ist, dass die vollständige Mail mit einem **Kopf** beginnt, der z.B. den Absender, das Datum und das Betreff der Mail enthält. Die eigentlichen **Daten** der E-Mail folgen auf den Kopf. Die Daten können verschiedener Art sein:

- reine Texte
- HTML-formatierte Texte
- Bilder, Audio-, Videodaten etc. in binärer Form.

Der Kopf kann auch Informationen enthalten, die den weiteren Aufbau der aktuellen Mail und die Art der Daten beschreiben.

- Im einfachsten Fall fehlen diese Informationen und auf den Kopf der Mail folgt dann eine Leerzeile und schließlich der Text mit der eigentlichen Nachricht.
- Komplexe E-Mails sind **MIME**-kodiert (Multipurpose Internet Mail Extensions). Solche Mails können aus einer Menge von Bereichen bestehen, die selbst wieder aus Kopfzeilen und anderen Daten bestehen.

Parsen

Mit dem IMP-Filter werden die Texte von E-Mails analysiert, d.h. die Texte werden gemäß ihrer Struktur oder Syntax zerlegt. Der Fachterminus hierfür ist ***Parsen***: die Texte werden geparst. Ein Programm, das Texte parst, ist ein Parser. Die **Parser** für die Texte können vom Benutzer selbst erstellt werden. Dazu dient der TextTransformer. Dieses Programm ist ein **Parser-Generator-IDE** Eine **IDE** (integrated development environment) ist eine integrierte Entwicklungsoberfläche. TextTransformer-Projekte enthalten die Spezifikationen für Parser.

5.2 Eine typische Mailtextstruktur

Mit dem IMP-Filter werden Text geparst, das heißt, dass Texte auf ihre Struktur und ihre Bestandteile hin analysiert werden. An einem einfachen Beispiel soll das Prinzip kurz demonstriert werden. Von dem Kopf der E-Mail und eventuellen Unterstrukturen mit binären Daten soll auf dieser einführenden Seite abgesehen werden. Hier soll der reine Text analysiert werden, wie er in einem E-Mail-Programm angezeigt wird.

Man kann diesen Text auf verschiedene Weise analysieren. Im einfachsten Fall als simple Wortliste. (Auch eine Liste ist eine Struktur.) Bei der folgenden typischen Mail drängt sich aber eine etwas komplexere Struktur auf.

```
    Lieber  Heinz,

    blah  blah  blah

    Herzliche  Grüße

    Dein  Fettie
```

Auf die Anrede folgt ein Text und darauf folgt ein Gruß. Und die Anrede ist wiederum in sich strukturiert und hat u.a. einen Namensbestandteil. Wenn es gelingt, den letzteren zu ermitteln, so kann er gut als Kriterium für eine Nicht-Spam-Mail verwendet werden. Wenn der Adressat der obigen Mail tatsächlich *Heinz* heißt, dann handelt es sich bei der Mail höchstwahrscheinlich nicht um Spam, wohl aber, wenn da z.B. stünde "Lieber Fettie". Das wäre vermutlich Reklame für ein Schlankheitsmittel.
Ein Wortfilter reicht in diesem Fall für die Klassifikation als Spam oder nicht Spam nicht aus. Heinz könnte ja einen Freund haben, dessen Spitzname *Fettie* ist. Für die Unterscheidung ist es daher nötig zu wissen, an welchen Stellen die Namen auftauchen: *Fettie* in der Anrede wäre Spam und *Fettie* als Unterzeichner nicht.

Mit dem TextTransformer kann man so einen Mail-Text analysieren. Dazu wird zunächst die Struktur obiger Mail etwas abstrakter beschrieben:

```
    Mailtext  ::=  Anrede  Text  Gruß
```

Anrede, Text und Gruß sind Textbestandteile, die in sich auf jeweils eigene Art strukturiert sind. Eine Definition für die Anrede könnte sein:

```
    Anrede  ::=  "Lieber"  "Heinz"
```

D.h. in einer Anrede folgen die Worte "Lieber" und "Heinz" aufeinander.
Das wäre aber zu kurz gegriffen. "Hallo Heinz" wäre eine ebenso korrekte Anrede. Deshalb könnte die obige Regel (Fachterminus: Produktion) so verallgemeinert werden:

```
    Anrede  ::=  ("Lieber"  |  "Hallo")  "Heinz"
```

D.h. in einer Anrede folgt auf eines der Worte "Lieber" oder "Hallo" das Wort "Heinz". Das ist sicher immer noch zu kurz gegriffen, veranschaulicht aber das Prinzip. Es funktioniert sehr ähnlich wie die regulären Ausdrücke, die vielen Benutzern des Spamihilators bekannt sein dürften. In einem

TextTransformer-Projekt können Regeln, die den Aufbau eines Textes beschreiben ganz ähnlich formuliert werden, wie reguläre Ausdrücke, die den Aufbau eines Wort-Ausdruckes beschreiben. Für die Textregel können z.B. die von den regulären Ausdrücken bekannten Wiederholungsoperatoren verwendet werden:

```
Text  ::=  (WORT+  SATZZEICHEN)*
```

Die Token *WORT* und SATZZEICHEN seinerseits können als "echte" reguläre Ausdrücke beschrieben werden.

5.2.1 Anmerkungen zur Syntax

Die regulären Ausdrücke im TextTransformer unterscheiden sich geringfügig von denen im Spamihilator. Im TextTransformer wird immer nach der längstmöglichen Übereinstimmung im Text gesucht, im Spamihilator nach der erstmöglichen. Auch ist die Analogie der Syntax der TextTransformer-Produktionen mir der der regulären Ausdrücke begrenzt. Dazu muss hier auf die Hilfe zum Texttransformer verwiesen werden. Hier nur die Andeutung: die Produktionen des TextTransformers müssen zunächst dem sogenannten LL(1)-Prinzip gehorchen, auch wenn dieses Prinzip auf einer höheren Ebene durch Vorauschau-Produktionen erweitert wird.

5.3 Spam oder Nicht-Spam

Ideal wäre, wenn man die Textstruktur erwünschter E-Mails so vollständig beschreiben könnte, dass die Menge der Mails, die nicht geparst werden können identisch wäre mit der Menge der Spam-Mails. Dieser Idealfall dürfte selten sein, ist aber nicht ausgeschlossen. Z.B. könnte Firmenpost so organisiert werden, dass sie eine exakt definierte Struktur aufweisen muss. Normalerweise wird aber auch der IMP-Filter mit Näherungen arbeiten und nur eine bestimmte Teilmenge der Mails wird sicher als Spam oder Nicht-Spam erkannt. Die anderen Filter des Spamihilators werden also weiterhin benötigt.

TextTransformer-Projekte sind normalerweise dafür da, um aus Quelltexten Zieltexte zu erzeugen. Für den Spamihilator ist der Zieltext einfach "1" oder "0" oder "-1", für Non-Spam, indifferenten Text und Spam.

Non-Spam	"1"
Spam	"-1"
indifferent	"0"

Dieser Rückgabetext wird in den sogenannten semantischen Aktionen erzeugt. Obige Definition der Anrede wird deshalb nochmals erweitert zu:

```
Anrede  ::=
("Lieber"  |  "Hallo")
(
     "Heinz" {{iResult = 1; }}
   | WORT     {{iResult = -1; }}
)
```

"iResult" ist eine zuvor deklarierte Integer-Variable. Zum genaueren Verständnis dieser Variable sei auf

zunächst die Beispielsprojekte hingewiesen und natürlich letztlich auf die Hilfe zum Texttransformer.

5.4 Parserfehler

Leider ist es nicht so einfach einen "allgemeinen Mail-Parser" zu konstruieren. Bei dem Versuch kann man jede Menge Überraschungen erleben. Z.B. eine Mail der folgenden Art:

```
---------

    (English version in the 2nd part of this mail.)
    ================================================


    Hallo  My_Name,

    blah blah blah

    Herzliche  Grüße

    Dein  Fettie


    ...

----------
```

Mit der obigen Produktion:

```
    Mailtext ::= Anrede Text Gruß
```

kann diese Mail nicht geparst werden, weil vor der Anrede noch ein anderer Text steht. Was der Filter in einem solchen Fall tut, lässt sich einstellen.

5.5 Parameter

Der IMP-Filter übergibt zwei Parameter an das geladene TextTransformer-Projekt.

1. auf die Logdatei kann im Projekt mit der Funktion *ConfigParam* zugegriffen werden.
2. ein Name für die aktuell Mail kann im Projekt mit der Funktion *ExtraParam* gelesen werden. Der Name wird aus dem Wort "Message" gebildet, dem das Startdatum des Spamihilators und eine Nummer für die Mail angehängt werden. Z.B.: Message_09-01-15-14-51-05_6

Achtung: Die Parameter werden eventuell in zukünftigen Updates des IMP-Filters komplexer. Das kann bedeuten, dass Sie Ihr Projekt dann ändern müssten, wenn sie die Parameter verwenden.

Mittels dieser Parameter ist es z.B. möglich zusätzliche Informationen in die Logdatei zu schreiben

```
{{
if(!ConfigParam().empty())
{
   RedirectOutput(ConfigParam(), true);   // an die Lofdatei anfügen
  m_bLog = true;
  out << ExtraParam() << endl;  //  Mail-Name
}
else
  m_bLog = false;
}}

... // Text parsen

{{
if(m_bLog)
{
  if(m_iResult == 0)
      out << indent << " indifferent" << endl;
   ResetOutput();
}
}}
```

Ein weiteres Beispiel für die Verwendung von *ExtraParam* ist die Speicherung der Mailtexte.

6 Einstellungen

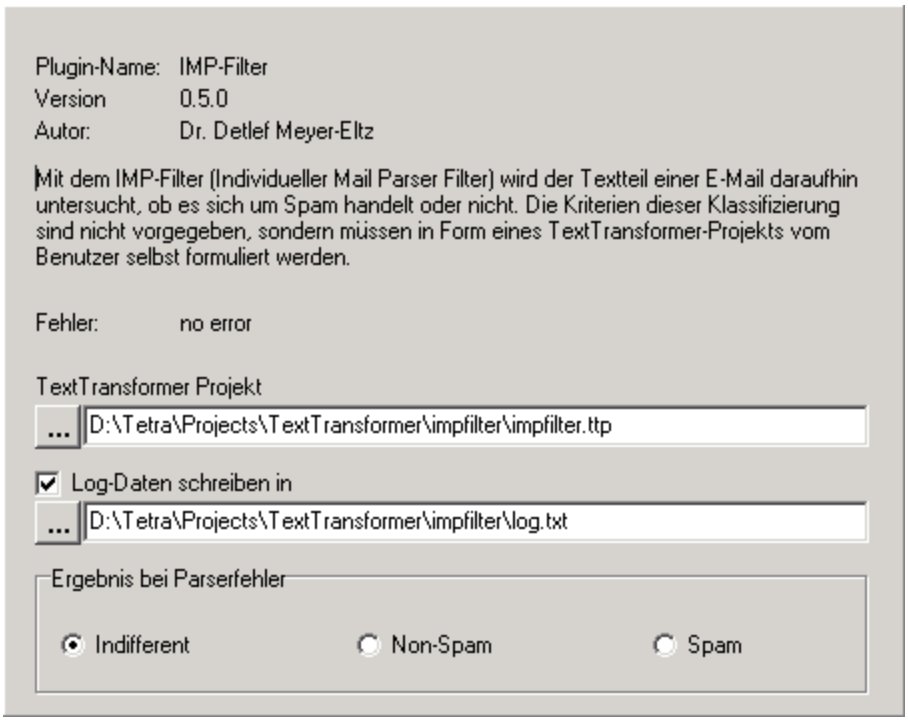
Wenn mit der rechten Maustaste auf das Programmsymbol des Spamihilators im Taskleisteninfobereich geklickt wird und dann *Einstellungen* gewählt wird, wid das Einstellungsfenster des Spamihilators angezeigt. Siehe:

http://wiki.spamihilator.com/doku.php?id=de:mainmenu

Dort kann zu den Plugins-Einstellungen navigiert werden:

http://wiki.spamihilator.com/doku.php?id=de:configplugins

Wird dort der IMP Filter markiert, kann dann über den *Einstellen*-Schalter ein Dialog zu dem Filter aufgerufen werden.



Der Dialog bietet Einstellungsmöglichkeiten für

die Projektdatei
die Reaktion auf Fehler des Parsers
das Schreiben einer Log-Datei

6.1 Projektdatei

Nach der Installation des IMP-Filters tut dieser zunächst nichts. Erst, wenn der Benutzer ein TextTransformer Projekt auswählt, tritt der Filter in Aktion. Beispielsprojekte gibt es im Anwendungsdaten-Verzeichnis.

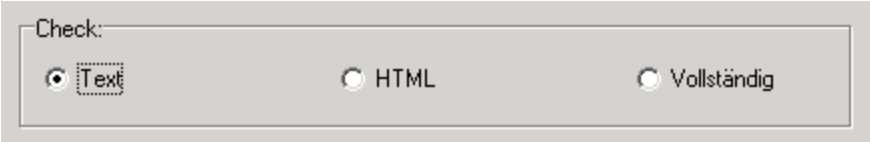
Änderungen an dem geladenen Projekt werden erst nach Neustart des Spamihilators oder nach dem erneuten laden der impfilter.dll wirksam.

6.2 Check

Der Spamihilator bietet seinen Plugins nicht nur die Möglichkeit den gesamten Text einer E-Mail inklusive der Kopfdaten zu analysieren, sondern er bietet auch die Möglichkeit die Analyse auf die

bereits extrahierten lesbaren Texte zu beschränken. Diese werden vom Spamihilator wiederum in zweifacher Form bereitgestellt: mit eventuell vorhandenen HTML-Formatierungen oder den reinen Text, aus dem alle HTML-tags entfernt sind.

Der IMP-Filter bietet seinen Anwendern die gleichen Möglichkeiten:



1. Text

Diese Option ist standardmäßig aktiviert und bedeutet, dass der reine lesbare Nachrichtentext, so wie er im E-Mail-Programm angezeigt wird, mit dem Projekt analysiert wird. Die in Spammails meist vorhandenen Links auf externe Internetseiten, sind in diesen reinen Texten häufig nicht sichtbar.

2. HTML

Wenn diese Option aktiviert ist, muss das verwendete Projekt in der Lage sein, mit HTML-tags zurechtzukommen. Leider ist gerade bei Spammails nicht garantiert, dass der HTML-Text wohlgeformt ist. Außerdem besteht der vom Spamihilator bereitgestellte Text oft aus reinen Textanteilen neben HTML-formatiertem Text.

3. Vollständig

Bei dieser Option stehen dem Parser sämtliche Information der E-Mail zur Verfügung, weil er die vollständige Mail mit Kopfdaten, Unterstrukturen und Binärdaten zu verarbeiten hat.

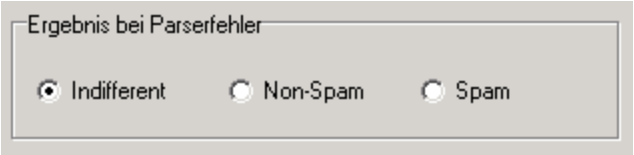
Nach einem Aufruf der Einstellungen wird ein Wert für den zu analysierenden Text die Spamihilator.ini eingetragen.

Text	Text
HTML	HTML
Vollständig	Complete

[impfilter.dll]
…
Checkr=Text

6.3 Ergebnis bei Parserfehler

Es stehen drei Optionen zur Auswahl, wie der Filter verfahren soll, wenn eine Mail nicht geparst werden kann.



1. Indifferent

Standardmäßig werden Mails, die nicht geparst werden können als indifferent behandelt und somit an die anderen Filter weitergegeben. Das ist die Einstellung für den IMP-Filter mit hoher Priorität.

2. Non-Spam

Die Klassifizierung der Mails als Non-Spam stellt sicher, dass sie nicht verloren geht. Das kann insbesondere dann sinnvoll sein, wenn man mit einem neuen TextTransformer-Projekt experimentiert und Spam-Mails als Beispiele zum Testen des Projekts benötigt werden.

3. Spam

Die Mail wird als Spam klassifiziert. Das ist riskant, weil Mails, die einmal als Spam markiert sind leicht verloren gehen können. Selbst erfahrene Benutzer des TextTransformers übersehen leicht Alternativen, die zu Parserfehlern führen. Diese Einstellung wird daher im Normalfall nicht empfohlen.

Nach einem Aufruf des Dialogs wird ein Wert für die Behandlungsmethode in die Spamihilator.ini eingetragen.

Non-Spam	1
Spam	-1
indifferent	0

```
[impfilter.dll]
…
OnParseError=0
```

6.4 Log-Datei schreiben in

Über die Checkbox kann eingestellt werden, ob eine Logdatei geschrieben werden soll oder nicht. Wenn das Häkchen gesetzt ist werden die neuen Log-Informationen an eventuell bereits vorhandene Daten angefügt.
Die Loginformationen, die standardmäßig vom IMP-Filter geschrieben werden, können mit Informationen

aus dem TextTransformer-Projekt angereichert werden.

(Dieser Code-Schnipsel ist aus *NonFree.ttp*.)
Nach einem Aufruf des Dialogs wird ein Wert für die Logoption und den Logpfad in die Spamihilator.ini eingetragen.

```
[impfilter.dll]
Log=1
LogPath=C=3A=5CProgramme=5CSpamihilator=5Cplugins
```

6.5 Letzte Fehlermeldung

Falls es einen Fehler gab, kann man sich die letzte Fehlermeldung in dem Einstellungsdialog nochmals ansehen.

7 Beispielsprojekte

Im Anwendungsdaten-Verzeichnis gibt es einige Beispielsprojekte. Es wird versucht, die Beispiele so weit zu erklären, dass sie auch ohne genauere Kenntnisse der TextTransformer Syntax verständlich werden. Das kann aber nur gelingen, wenn der Leser sie sorgfältig in der Reihenfolge studiert, wie sie angeführt sind.

Die Beispielsparser sind zu Gruppen zusammengefasst, die den Check-Optionen entsprechen.

Reine Text-Parser

- Emptymail
- NonSpam-Worte
- SpamAndNonSpam
- DirtyFriends
- Formmail
- Impfilter

HTML- und Text-Parser
Vollständige Mail-Parser

Die Projekte lassen sich teilweise durch einfache Wortersetzungen und analoge Ergänzungen intuitiv modifizieren. Für anspruchsvollere Projekte muss auf die Hilfe zum TextTransformer verwiesen werden.

7.1 Reine Text-Parser

Wenn einer der Parser für reine Texte im IMP-Filter verwendet werden soll, so sollte entsprechend die Option zum Check von Texten eingestellt werden. Die folgenden Beispiele sind allerdings so konstruiert, dass sie auch mit den anderen Check-Optionen funktionieren sollten.

Die Parser für reine Texte sind am leichtesten an die eigenen Bedürfnisse anzupassen, weil ihr Verständnis keine Kenntnis des Aufbaus einer vollständigen E-Mail oder von HTML erforderlich ist.

Emptymail
NonSpam-Worte
SpamAndNonSpam
DirtyFriends
Formmail
Impfilter

7.1.1 Emptymail

Das Beispielsprojekt Emptymail.ttp im Anwendungsdaten-Verzeichnis funktioniert ähnlich wie der Spamihilator Empty Mail Filter. Es soll aber den flexiblen Empty Mail Filter keineswegs ersetzen. Das Emptymail-Projekt ist nur aus dem didaktischen Grund gemacht, dass es sehr einfach ist. Das gesamte Projekt besteht aus nur zwei Zeilen:

```
  SKIP {{ out << 0; }}
| {{ out << -1; }}
```

SKIP ist ein besonderes Token im TextTransformer. Damit wird der gesamte Text erkannt bis zu der Position an der etwas auf *SKIP* folgt. Hier folgt allerdings nichts auf *SKIP* als das Textende. Deshalb wird hier mit SKIP der gesamte Text erkannt, wenn es einen gibt. Wenn die Mail keinen Text enthält greift die die Alternative hinter dem oder-Zeichen '|', also "{{ out << -1; }}"
Die Klammern "{{...}}" kennzeichnen Aktionen. "out << -1;" bedeutet, dass der Wert '-1' ausgegeben wird und der Wert '-1' steht für Spam. Der Wert '0' hingegen bedeutet, dass durch den Filter nicht entschieden werden kann, ob der Text Spam ist oder nicht.

7.1.2 NonSpam-Worte

Das im folgenden beschriebene Projekt *Nonspamwords.ttp* eignet sich gut dazu einen ersten Test der Funktion des IMP-Filters nach der erstmaligen Installation durchzuführen. Er sollte dazu mit hoher Priorität aufgerufen werden.

Es gibt eine Menge Spamihilator-Filter, die dazu dienen, Spam aus den empfangenen E-Mails herauszufiltern. Demgegenüber verfolgt das Projekt *Nonspamwords.ttp* eine positive Strategie. E-Mails, die bestimmte Schlüsselworte enthalten, sollen als Nicht-Spam vor der folgenden negativen Filterung geschützt werden. Solche Schlüsselworte können der eigene Name oder Firmenname sein, oder auch

eigene Produktbezeichnungen oder Begriffe zu eignen besonderen Interessengebieten.

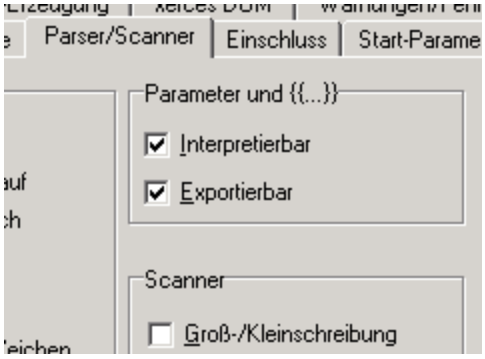
```
{{
int iResult = 0;
}}
SKIP?
(
   (
         "Spamihilator"
      |  "TextTransformer"
      |  "tetra"
   )  {{ iResult = 1; }}
   SKIP?
)?
{{
out << iResult;
}}
```

Hier wird das Resultat nicht direkt in die Ausgabe geschrieben, sondern in der Variablen *iResultat* zwischengespeichert. Erst zum Schluss wird der Wert mit der Anweisung "out << iResultat" ausgegeben. Bei der Deklaration (Erzeugung) den Variablen mit "int iResultat = 0" wird ihr gleich der für Indifferenz stehende Wert 0 zugewiesen. Dieser Wert wird nur geändert, wenn die äußere Klammer (...)? durchlaufen wird. Das Fragezeichen '?' bedeutet - wie bei den regulären Ausdrücken - dass der Ausdruck vor dem Fragezeichen optional ist. Optional ist das Vorkommen eines der Worte, die durch das oder-Zeichen '|' voneinander getrennt sind.

```
   (
         "Spamihilator"
      |  "TextTransformer"
      |  "tetra"
   )  {{ iResult = 1; }}
   SKIP?
```

Wenn eines der Worte im Text gefunden wird, wird der für Nicht-Spam stehende Wert 1 der Variablen *iResult* zugewiesen. Dann wird der wahrscheinlich nachfolgende Textabschnitt mit dem zweiten *SKIP* bis zum Ende übersprungen. Wenn keines der Nonspam-Worte im Text vorkommt, wird mit dem ersten *SKIP* der gesamte Text erkannt.

Um mehr mögliche Schreibweisen der NonSpamwörter abzudecken, wurde im TextTransformer in den Projektoptionen die Groß-/Kleinschreibung abgeschaltet.



Durch zwei kleine Modifikationen, könnte dieses Projekt mit dem Emptymail-Projekt kombiniert werden:

```
{{
int iResult = 0;
}}
(
    SKIP
    (
       (
             "Spamihilator"
           |  "TextTransformer"
           |  "tetra"
       )  {{ iResult = 1; }}
    )?
  | {{ iResult = -1; }}
)
{{
out << iResult;
}}
```

In der "leere" Alternative "| {{ iResult = -1; }}" wird nur eine Aktion ausgeführt, aber es wird kein Text konsumiert. Sie macht den umgebenden Klammerausdruck auch ohne '?' optional.

7.1.3 SpamAndNonSpam

Das vorherige Projekt soll nun in dem Projekt *SpamAndNonSpam.ttp* so erweitert werden, dass auch Spam-Worte erkannt werden. Wegen der Priorität des IMP-Filters, sollten dies aber nur wenige ausgezeichnete Worte sein, die mit sehr hoher Wahrscheinlichkeit auf Spam schließen lassen.

Um das Projekt übersichtlicher zu machen, werden zunächst zwei zusätzliche Regeln (Produktionen) erzeugt. Die erste *NonSpam* umfasst die Nicht-Spam-Worte.

```
   "Spamihilator"
|  "TextTransformer"
|  "tetra"
```

Die zweite zusätzliche Regel möge *Spam* heißen und umfasst einige Spamworte:

```
  "Viagra"
| "Casino"
| "Rolex"
| "Watch"
| "Watches"
```

In der Startregel von *SpamAndNonSpam* wird eine Schleife solange ausgeführt, bis der gesamte Text verarbeitet ist. Der Stern '*' hinter dem Klammerausdruck symbolisiert die Schleife.

```
{{
int iResult = 0;
}}
(
    SKIP
  | NonSpam  {{ iResult = 1; }}
  | Spam      {{ if(iResult == 0) iResult = -1; }}
)*
{{
```

```
out << iResult;
}}
```

Die Formulierung mit der Schleife ist ein kleiner Trick. Die Alternativen zu **SKIP** können ihm sowohl vorausgehen als auch nachfolgen. In der Startregel wird demnach mit *SKIP* entweder zum nächsten Spamwort oder Nicht-Spam-Wort gesprungen oder bis zum Ende des Textes. Wenn ein Wort gefunden wurde, wird dem Result ein Wert zugewiesen.
Die Mail wird aber nur dann als Spam klassifiziert, wenn sie kein Nicht-Spam-Wort enthält. Dafür sorgt die Anweisung:

```
if(iResult == 0) iResult = -1;
```

D.h. das Resultat wird nur dann gleich -1 gesetzt, wenn es noch indifferent ist. Wenn dann noch ein Nicht-Spam-Wort folgt, wird das Resultat auf 1 gesetzt, die Mail also insgesamt als Nicht-Spam klassifiziert. So kann der IMP-Filter weiterhin mit hoher Priorität verwendet werden.

7.1.4 DirtyFriends

Das Projekt DirtyFriends.ttp erlaubt es speziellen Freunden ihre schmutzigen Phantasien auszudrücken, wenn sie sich namentlich dazu bekennen, d.h. die Mail mit ihrem Namen unterzeichnen.
Im Resultat wirkt dies Projekt sehr ähnlich wie das *SpamAndNotSpam* Projekt. Statt der NonSpam-Produktion gibt es hier eine *Friends* Produktion. Die Hauptregel ist so aufgebaut, dass sofort zum Ende des Textes gesprungen wird, wenn das Ergebnis feststeht. Der Nachteil ist allerdings, dass die Regel unübersichtlich ist.

```
{{
int iResult = 0;
}}
(
  SKIP?
  (
      Friends[iResult] SKIP?
    | Spam
      (
        SKIP?
        (
            Friends[iResult] SKIP?
          | {{ iResult = -1; }}
        )
      )
  )?
)
{{
out << iResult;
}}
```

Die Produktion *Friends* umfasst eine Liste der Freunde:.

```
(
    "Peter"
  | "Paul"
  | "Mary"
)
{{
xiResult = 1;
}}
```

Neu ist, dass die *Friends*-Produktion mit *iResult* als Parameter aufgerufen wird:

```
Friends[iResult]
```

.
Der Parameter muss dabei im TextTransformer in der Friends-Produktion so deklariert sein:

Mit dem Parameter wird der Wert gewissermaßen aus der Friends-Produktion geholt, mit "`xiResult = 1;`"

Man beachte die vielen '?' in der Startregel. Sie sind nötig, um sicherzustellen, dass jeder Text geparst werden kann. Parserfehler können so nicht auftreten, auch, wenn z.B. die ganze Mail nur aus einem Spamwort besteht.

7.1.5 Formmail

Das Beispielsprojekt *Formmail.ttp* demonstriert, wie Mails eindeutig als Nicht-Spam klassifiziert werden können, wenn sie eine bestimmte Struktur aufweisen. Da das Projekt keine Mail als Spam abweist, sollte es wiederum weit oben in der Liste der Filter platziert werden.

Dieses Projekt soll nur eine imaginäre Firmenpost durchlassen. Z.B. folgende Mail.

```
Sender:  Fettie
Subject:   Schlankheitskur
Message:  will  nicht!
```

Das Projekt besteht aus den Produktionen:

```
Sender  ::= SKIP EOL
Subject:  "Subject:" SKIP EOL
Message  ::=  "Message:" SKIP?

Formmail  ::= Sender  Subject  Message   {{ out << 1; }}
```

EOL ist ein regulärer Ausdruck, der für das Zeilenende steht. Es ist im TextTransformer auf der Token-

Seite definiert als: \r?\n.
Durch die Projekteinstellungen des TextTransformers werden Leerzeichen und Zeilenumbrüche zwar standardmäßig ignoriert, als Nachfolger von *SKIP* wird *EOL* aber dennoch erkannt.

Es fällt auf, dass dieses Projekt immer den Wert 1 (für Nicht-Spam) zurückgibt. Dennoch besteht keine Gefahr, dass Spam akzeptiert wird, weil sicher ist , dass eine Mail, die nicht die angeführte Struktur hat, gar nicht bis zu ihrem Ende geparst werden kann. Stattdessen wird im Spamihilator der Wert für den Fall von Parserfehlern verwendet. Dieser Wert sollte für dieses Projekt daher unbedingt auf "indifferent" eingestellt sein.

Übungsaufgabe:
Auch Mails ohne Absender und Betreff werden nicht geparst. Wie kann man das ändern?

7.1.6 NonFree

NonFree.ttp ist das reale Projekt, an dem ich vor der ersten Veröffentlichung des IMP-Filters einigen Wochen gefeilt habe, um es an meine speziellen Bedürfnisse anzupassen. Nur mein Name ist darin geändert. Bei den bisherigen Projekten wurde darauf acht gegeben, dass sie alle mit der freien Version des TextTransformers verwendbar sind. Das gilt für *NonFree.ttp* nicht. Trotzdem soll es hier als Anregung und "Steinbruch" kurz vorgestellt werden.

Die zentrale Regel darin ist die text-Produktion. Sie besteht im wesentlichen aus einer Schleife mit alternativen regulären Ausdrücken für Worte, Satzzeichen und den übrigen druckbaren Zeichen

```
(
    WORD
  | PUNCTUATION
  | SPECIAL
  | NBSP      {{ SetIsSpam(" : NBSP"); }} //  geschütztes  Leerzeichen
)*

WORD  ::=  [[:alpha:]]+
PUNKTUATION  ::=  [[:punct:]]+
SPECIAL   ::=   [^[:alnum:][:punct:][:space:]Æ-Ïæ-ï€ÀÁÂÃÄÒÓÔÖÙÚÛÜßàáâäòóôöùúûü]+
NBSP  ::=  \xA0
```

Mit dieser Schleife können alle Texte vollständig erkannt werden, weil die Ausdrücke alle ANSI-Zeichen umfassen. (Leerzeichen werden gemäß den Projekteinstellungen automatisch übersprungen.)

Diese Schleife ist nun durch zusätzliche Alternativen erweitert, die die Anfänge möglicher Spam sind. Z. B.:

```
(
    WORD
  | PUNCTUATION
  | SPECIAL
  | NBSP      {{ SetIsSpam("NBSP"); }} //  geschütztes  Leerzeichen
  | pricelist_if
)*
```

Pricelist_if sieht so aus:

```
IF( pricelist() )
  pricelist           {{ SetIsSpam("price list");  }}
ELSE
  price
END
```

Pricelist_if wurde hier ausgesucht, weil darin ein interessantes Feature des TextTransformers verwendet wird, das allerdings in der freien Version nicht verfügbar ist. Mit den Zeilen:

```
IF( pricelist() )
  pricelist
```

wird der Parser angewiesen im Text vorauszuschauen. Nur, wenn tatsächlich eine ganze Liste von Artikel und Preisen folgt, wird der Text als Preisliste geparst und das Spamattribut gesetzt. Andernfalls wird an der aktuellen Stelle mit der obigen Schleife fortgefahren. Die Vorauschau hat den Vorteil, dass nicht nach jedem erkannten Token alle möglichen Alternativen bedacht werden müssen, um einen vorzeitigen Abbruch des Parsers zu verhindern.

7.2 HTML- und Text-Parser

Wenn einer der Parser für HTML und Texte im IMP-Filter verwendet werden soll, so sollte entsprechend die Option zum Check von HTML eingestellt werden.

Auch, wenn die HTML-Option gesetzt ist, heißt das nicht unbedingt, dass der Spamihilator einen Text liefert, der HTML-tags enthält. Parser für diese Option müssen daher in der Lage sein, sowohl mit HTML-Code als auch mit reinem Text oder einer Mischung von beidem zurechtzukommen.

HTMLText
AllLinksAreSpam

7.2.1 HTMLText

Der HTMLText-Parser kommt sowohl mit HTML-Code als auch mit reinem Text oder einer Mischung von beidem zurecht. Auch setzt er mit einer Ausnahme nicht voraus, das HTML-Code wohlgeformt ist. Bei einer solchen Annahme würde der Parser häufig scheitern. Aber, wenn das Token "<!DOCTYPE" gefunden wird, wird angenommen, dass diese der Beginn eines wohlgeformten HTML-Codeabschnitts ist.
Solange nicht von wohlgeformtem HTML ausgegangen werden kann, ist es mit der freien Version des TextTransformers leider nicht möglich zu unterscheiden, ob es sich bei '<' oder '>' um Anfang bzw. Ende eines tags handelt, oder um das Kleiner- bzw, Größer-Zeichen. Damit ist auch nicht bekannt, ob der Parser sich innerhalb oder außerhalb eines tags befindet. (Mit der Standard-Version des TextTransformers könnte eine Vorausschau zur Entscheidung herangezogen werden.)

Um dieses Projekt zu einem Spamfilter zu machen, muss es durch eigene Testfunktionen erweitert werden. Für den Aufruf dieser Funktionen ist die *TextToCheck*-Produktion vorhanden:

```
   WORD
|  STRING
|  SPECIAL
|  Link
```

Hier sind die wichtigen Textbestandteile versammelt: Worte, Anführungen, spezielle Zeichen, E-Mailadressen und Links.

7.2.2 AllLinksAreSpam

AllLinksAreSpam beruht auf dem *HTMLText*-Projekt. Syntaktisch sind sie identisch. Aber es wurde eine simple semantische Aktion eingebaut, die die Mail als Spam klassifiziert, sobald ein Link in ihr gefunden wird. Das macht insofern Sinn, als dass nahezu jede Spammail als Link-Transporteur dient. Wenn man also Links nur in E-Mails zulassen möchte, deren Adressen in der Freundesliste stehen, hat man mit *AllLinksAreSpam* einen effizienten Spamfilter. Außerdem demonstriert dieses Projekt den Vorteil der HTML-Option gegenüber der Text-Option: in reinen Texten sind oft nicht alle Links enthalten.

Die Aktion erfolgt in der Link-Produktion:

```
  NORMAL_LINK                {{m_iResult = -1; }}
|   "http://www.mydomain.com"
|  "mailto:"?
  (
      EMAIL                  {{m_iResult = -1; }}
    | "myname@mydomain.com"
  )
```

NORMAL_LINK ist ein regulärer Ausdruck, der das Muster beschreibt, nach dem die meisten Links aufgebaut sind.

```
NORMAL_LINK  ::=
(http://|ftp://)?[^\r\n\t   <>"@]+(\.[^\r\n\t   <>"@]+)+
```

EMAIL ist ein regulärer Ausdruck, der das Muster beschreibt, nach dem die meisten E-Mailadressen aufgebaut sind.

```
EMAIL  ::=
[\w\.-]+  \//  local  part
@ \
([\w-]+\.)+  \  //  sub  domains
[a-zA-Z]{2,4}  //  top  level  domain
```

Ein Sonderfall bilden die eigenen Adressen. Bisweilen werden sie von Spammern mit in die Mail kopiert. Aber es kann auch sein, dass die eigenen Adressen darauf hindeuten, dass die Mail eine Antwort von einem Absender ist, den Sie noch nicht in Ihre Freundesliste aufgenommen haben.
Es empfiehlt sich für die eigenen Adressen einen regulären Ausdruck zu entwickeln, der nur dann auf die Adresse passt, wenn sie exakt auf die Weise zitiert ist, wie Sie sie in Ihre Mails schreiben. Z.B.: würde der regulärer Ausdruck

```
MY_EMAIL   ::=   -+\r?\nmailto:myname@mydomain.com
```

auf folgende Schreibweise der Adresse passen:

```
----------------
mailto:myname@mydomain.com
```

Dir Link-Produktion könnte dann ergänzt werden zu:

```
    NORMAL_LINK                 {{if(m_iResult != 1) m_iResult = -1; }}
  | "http://www.mydomain.com"
  |  "mailto:"?
    (
        EMAIL                   {{if(m_iResult != 1) m_iResult = -1; }}
      | MY_EMAIL                    {{m_iResult = 1; }}
    )
```

7.3 Vollständige Mail-Parser

Wenn einer der vollständigen Mail-Parser im IMP-Filter verwendet werden soll, so sollte entsprechend die Option zum Check der vollständigen E-Mail eingestellt werden.

Die Standard-Spezifikation für E-Mails heißt MIME (Multipurpose Internet Mail Extensions). Einen wirklich vollständigen MIME-Parser, der auch die Unterstrukturen aller Felder detailliert parst gibt es hier:

http://www.text-konverter.homepage.t-online.de/MIME_ge.htm

Oft sind Spam-Mails schludrig gemacht und passieren diesen Parser nicht. So könnte er mit wenig Aufwand als Spamfilter verwendet werden. Allerdings wär das vielleicht ein Spatzenschießen mit Kanonen.

Der Simple_MIME-Parser ist eine generalisierte Form des obigen MIME-Parsers, die ziemlich tolerant gegenüber Syntaxverstößen ist. Es wurden nur die Teile aus obigem MIME-Parser übernommen, die nötig sind, um die MIME-Struktur aufzuschlüsseln und so die Textabschnitte zugänglich zu machen, getrennt von den Kopfdaten und den binären Teilen.

Simple_MIME

7.3.1 Simple_MIME

Der Name "Simple_MIME" täuscht. Einfach ist dieser Parser nur im Vergleich zu dem detailliertem MIME-Parser, aus dem *Simple_MIME* extrahiert wurde. Die Änderungen an der ausgeklügelten syntaktischen Struktur des Parsers ist nur versierten Kennern des TextTransformers zu empfehlen.

Was hingegen mehr Nutzern des IMP-Filters möglich sein sollte, ist eine semantische Erweiterung des Projekts derart, dass es eine Klassifikation in Spam- oder Nicht-Spam gemäß individuellen Kriterien vornimmt. Der Simple_MIME Parser tut in dieser Hinsicht zunächst nichts. Er ist ein bloßes Gerüst, das der Erweiterung bedarf.

Um solche Spam-Tests zu programmieren, sind Grundkenntnisse des Texttransformers nötig. Dann aber sind den Klassifizierungskriterien keine Grenzen mehr gesetzt, weil im Prinzip **auf alle Informationen**

der E-Mail zugegriffen werden kann.

Die wichtigste Stelle, an der ein Spam-Test eingebaut werden kann ist die *text_element*-Produktion:

```
    WORD
  | DIGITS
  | STRING
  |  PUNCTUATION
  | SPECIAL
  | NBSP
```

WORD ist hier das gleiche Token zur Erkennung von Worten, das schon in einem der vorherigen Projekte verwendet wurde:

```
WORD  ::=  [[:alpha:]]+
```

Der erkannte Inhalt kann z.B. auf folgende Weise mit einem Spamwort verglichen werden:

```
WORD
{{
if(xState.str()  ==  "Spamwort")
   m_iResult = -1;
}}
```

Die *text_element*-Produktion kann leicht um weitere Alternativen ergänzt werden, um z.B. Links oder E-Mail-Adressen zu identifizieren und zu bewerten.

Die anderen Token der *text*-Produktion sind nötig, um für alle möglichen Zeichen eine Erkennungsalternative zu haben.

8 Weitere Möglichkeiten

Neben der reinen Syntaxanalyse der Quelltexte können in TextTransformer-Projekten Befehle der Programmiersprache C++ ausgeführt werden. So können die aus der Analyse gewonnenen Daten weiter aufbereitet und bewertet werden. Die Projekte sind damit nicht auf eine Spamanalyse beschränkt, sondern eingegangene Post kann auch auf beliebige andere Weise automatisch verarbeitet werden. Hier nur zwei einfache Vorschläge:

Testmails speichern
Mails sortiert abspeichern

8.1 Testmails speichern

Während der Entwicklungsphase eines neuen Projektes ist es hilfreich die Klassifikation im TextTransformer-Debugger schrittweise testen zu können, falls sie nicht korrekt war.

Das Projekt *SaveTests.ttp* enthält Code, der dafür sorgt, die Texte der E-Mails in verschiedene Verzeichnisse geschrieben werden, je nachdem, ob es indifferente Mails waren oder Spam oder Nicht-

Spam.

Das Projekt setzt voraus, dass es auf der Festplatte einen Ordner mit folgender Struktur gibt:

```
..\impfilter\Spam
..\impfilter\NonSpam
..\impfilter\Indifferent
..\impfilter\OnError   // für CopyTextToDisk in NonFree
```

Das *impfilter*-Verzeichnis ist entweder, wie im Projekt erwartet direkt auf den Laufwerk C: zu erstellen oder der Pfad muss im Projekt angepasst werden.

```
{{
str sTestDir = "C:\\impfilter";    <- anpassen
str sTestFile;
int iResult = 0;
}}
SpamAndNonSpam[iResult]
{{
out << 0;  // test only

switch(iResult)
{
  case 1:
   sTestDir += "\\NonSpam";
  break;
  case 0:
   sTestDir += "\\Indifferent";
  break;
  case -1:
   sTestDir += "\\Spam";
  break;
}

sTestFile = append_path(sTestDir, ExtraParam() + ".txt");

if(exists(sTestDir))
{
     RedirectOutput(sTestFile);
   out << xState.lp_str();    // text, der von der letzten Produjtion erkannt wurde
    //out << xState.text(0);  // in der freien Version des TextTransformer nicht erlaubt
    //ResetOutput();
}
}}
```

Das Projekt beeinträchtigt die Spamihilator Filter nicht, da immer der Wert 0 für indifferente Mails zurückgegeben wird. Aber die Mail werden so in die Verzeichnisse sortiert, wie sie klassifiziert würden, wenn *iResult* statt 0 zurückgegeben würde.
iResult wird hier berechnet wie im Projekt SpamAndNonSpam. Die Startregel von *SpamAndNonSpam* wird hier als Unterregel aufgerufen. Das ist ein Trick, um den gesamten Mailtext mit xState.lp_str() (-> siehe Hilfe zum TextTransformer) ausgeben zu können. Diese Funktion darf in der freien Version des TextTransformers verwendet werden. im Unterschied zur text-Funktion die sich sonst anböte.

Im Projekt *NonFree* ist ähnlicher Code als Funktion ***CopyTextToDisk*** eingebaut. Dort werden auch Mails gespeichert, bei denen ein Parserfehler auftrat.

8.2 Mails sortiert abspeichern

Nachdem es schon eine Möglichkeit gibt die Mails nach ihrem Spam-Charakter sortiert abzuspeichern, liegt die Möglichkeit auf der Hand, den Spamihilator nicht nur zur Mailfilterung, sondern auch zur Mailverarbeitung zu verwenden. So könnten in einer Firmenpost z.B. Aufträge, Beschwerden, Angebote etc. getrennt abgespeichert werden, nachdem ihr Typ durch die Textanalyse ermittelt wurde.

9 Lizenzen

Der IMP-Filter unterliegt der gleichen Lizenz wie der Spamihilator.

http://wiki.spamihilator.com

Der IMP-Filter verwendet Teile der boost-Bibliotheken:

http://www.boost.org

Der IMP-Filter bindet indirekt das xerces Paket ein.

http://xerces.apache.org/xerces-c/

9.1 Spamihilator-Lizenz

SPAMIHILATOR LICENSE

Version 1.0
Copyright (c) 2002-2003 Michel Krämer

Everyone is permitted to copy and distibute verbatim copies of this document,
but changing is not allowed.

1. DEFINITIONS

1.1 "License" means this document.

1.2 "You" means the licensee.

1.3 "Source code" means the preferred form of the code, that is covered by this license, including all modules it contains, any associated interface definition files and scripts used to control compilation or installation of an executable or source code.

1.4 "Compilation" means a binary or executable file or binary or executable files created by a compiler, assembler or linker or any other software that is able to process source code.

1.5 "Author" means the original copyright holder of this software.

2. COPYING AND DISTRIBUTION

This software is FREEWARE. You may distribute copies of this software but you may NOT charge for the software or the service of distrubution.
If you make copies of this program you must give the recipients all the rights you have. You must also include this license in the distribution.

Although some parts of the program's source code may be published by the Author, you are not allowed to distribute it in any way. You may read the source code to get an idea how to write new code for yourself but you may not copy any portion of the original source code to create a new compilation or verbatim compositions.

3. MODIFYING

Modifying your copy or copies of the program or any portion of it in any way is prohibited.

Although some parts of the program's source code may be published by the Author, you are not allowed to modify these parts to create a new compilation or verbatim compositions.

4. DECOMPILING

Decompiling, disassembling or any reverse engineering of the binary and executable files that are associated with this program is prohibited.

5. ACCEPTING THIS LICENSE

Distributing or copying the program is prohibited by law unless you accept this license. You are not required to do this, because you have not signed the license, but by copying or distributing the program, you declare your acceptance of this License.

6. DISCLAIMER OF WARRANTY

THIS SOFTWARE IS PROVIDED "AS IT IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THIS SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THIS PROGRAM IS WITH YOU. IF THE SOFTWARE SHOULD PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE AUTHOR OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

7. TERMINATION

This license is valid as long as you use the product. It will be terminated as soon as you violate any of its terms and conditions. In this case you have to immediately destroy, respectively delete, all copies of this product you made.

8. MISCELLANEOUS

This license represents the complete agreement between the Author and you. It overrides all other agreements, either spoken or written. It can only be changed by a written and signed contract.

If, as a consequence of a court judgment or allegation of patent violation or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this license, they do not excuse you from the conditions of this license. If you cannot distribute so as to satisfy simultaneously your obligations under this license and any other pertinent obligations, then as a consequence you may not distribute the program at all.

END OF TERMS AND CONDITIONS.

9.2 boost-Lizenz

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer,

must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

9.3 xerces-Lizenz

Xerces-C++ unterliegt der Apache Software License, Version 2.0.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Index

- N -

- O -

- P -

- R -

- S -

- T -

- V -

Endnotes 2... (after index)

Back Cover