

Klassifizierung von SPAM mittels Library Based Ensemble Classifier

Saarbrücken, 22. April 2007

Sebastian Germesin
Betreuung: Michael Feld

Lehrstuhl Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
Deutsches Forschungszentrum für Künstliche Intelligenz
Saarbrücken, Deutschland



Zusammenfassung

In der vorliegenden Arbeit wird auf die Problematik von SPAM-eMails und die Verteidigung mit Hilfe aktueller Machine Learning Verfahren eingegangen. Es werden zunächst die entstehenden Probleme und Kosten, welche durch SPAM-eMails auftreten, aufgeführt und die bereits bestehenden Verteidigungsmaßnahmen erläutert. Anschließend wird ein Machine Learning SPAM-Filter vorgestellt. Dieser wurde auf Basis eines Library Based Ensemble Classifiers implementiert, da dieser - gegenüber den individuellen Klassifizierern - bessere Klassifikationsergebnisse liefert. Zudem wird die Entwicklung eines SPAM-Filters mit Hilfe von Machine Learning Algorithmen aus dem WEKA-Toolkit beschrieben.

Inhaltsverzeichnis

1	Einführung	1
1.1	Gliederung	1
1.2	Zusammenfassung	2
2	SPAM	3
2.1	Auswirkungen	3
2.2	Verteidigungsmöglichkeiten	3
3	Ensemble Classifier	5
3.1	Individual vs. Ensemble	5
3.2	Beteiligte Machine Learning Algorithmen	7
3.3	Ensemble-Erstellung	7
3.4	Anwendung eines LBEC als SPAM-Filter	9
4	Implementierung eines lernfähigen SPAM-Filters	11
4.1	Durchführung	11
4.2	Attribute	12
4.3	eMail-Korpus	12
4.4	Ergebnisse	14
5	Konklusion	16
5.1	Zusammenfassung	16
5.2	Ausblick	16
	Literaturverzeichnis	18

Abbildungsverzeichnis

3.1	statistische Sicht	6
3.2	computationale Sicht	6
3.3	repräsentative Sicht	6
4.1	Screenshot: SPAM-Filter	13

Kapitel 1

Einführung

Als SPAM werden allgemein unerwünschte Nachrichten bezeichnet. Jedoch hat sich dieser Begriff besonders auf dem Gebiet der elektronischen Nachrichten (eMails) etabliert. Die Anzahl dieser unerwünschten elektronischen Nachrichten hat in den letzten Jahren enorm zugenommen und verursacht der Wirtschaft Milliarden-schwere Verluste. Zurzeit besteht ein reger Wettlauf zwischen der SPAM-Verbreitung und der Entwicklung neuer Verteidigungstechniken. Die schnell wechselnden Charakteristika bringen statische - auf regulären Ausdrücken basierende - Filter schnell an ihre Grenzen und machen sie auf Dauer zu pflegeintensiv. Deshalb werden verstärkt Machine Learning Filter entwickelt, welche sich an neue Arten von SPAM automatisch anpassen können und somit keiner bzw. nur geringer Wartung bedürfen. In dieser Arbeit wird besonders auf eine bestimmte Art von Machine Learning Filter eingegangen: die *Library Based Ensemble Classifier* (LBEC). Dabei werden mehrere Klassifizierer zu einem Ensemble zusammengefasst und die Ergebnisklassifikation durch einen Meta-Algorithmus aus den Teilergebnissen der beteiligten Algorithmen berechnet.

1.1 Gliederung

In Kapitel 2 werden die Auswirkungen von SPAM beschrieben und bisherige und weitere Möglichkeiten zur Bekämpfung von SPAM erläutert.

Die Beschreibung und Erstellung eines LBEC wird in Kapitel 3 behandelt. Es wird dabei auf verschiedene Techniken zur Bibliotheks-Erstellung eingegangen und Meta-Algorithmen zur Klassifizierung erläutert.

In Kapitel 4 wird die Implementierung eines SPAM-Filters im Rahmen des dieser Arbeit zu Grunde liegenden Seminars erläutert. Dabei werden Probleme die während der Implementation auftraten, besprochen und die erreichten Ergebnisse dargelegt.

Kapitel 5 bildet mit der Zusammenfassung und einem Ausblick in die Spracherkennung den Abschluss dieser Arbeit.

1.2 Zusammenfassung

Die vorliegende Arbeit ist eine Ausarbeitung des gleichnamigen Vortrages im Rahmen des Seminars AI Tools an der Universität des Saarlandes im Wintersemester 2006/07. Dabei wird das Thema SPAM behandelt und Verteidigungsmaßnahmen gegen SPAM erläutert. Eine Art der Verteidigung bilden die sogenannten SPAM-Filter, welche die eMails vor dem Lesen des Empfängers nach erwünschten und unerwünschten eMails trennen. In der vorliegenden Arbeit wird auf Machine Learning Filter eingegangen und deren Vorteile gegenüber statischen Filtern an Hand eines Library Based Ensemble Classifiers erläutert. Anschließend werden Ergebnisse der im Rahmen des Seminars durchgeführten Implementierung eines Machine Learning SPAM-Filters vorgestellt und interpretiert. Diese Implementierung wurde mit Hilfe des JAVA-Toolkits WEKA durchgeführt, welches eine Grundlage für Machine Learning Programmierung bildet.

Kapitel 2

SPAM

Als SPAM werden heutzutage unerwünschte eMails bezeichnet, die dem Empfänger unverlangt zugestellt werden und massenhaft versandt wurden oder werbenden Inhalt haben. Es wird dabei zwischen mehreren Arten von SPAM unterschieden: *Unsolicited Bulk Email*, *Phishing*, *Joe-Job* und *Hoax* (aus: Clayton, 2004, Petković u. a., 2005, Koj u. a., 2006 und zeitform Internet Dienste, 2003). In dieser Arbeit werden diese jedoch alle mit dem Oberbegriff SPAM bezeichnet und nicht auf die einzelnen Untergruppen en-detail eingegangen.

2.1 Auswirkungen

Im Jahre 2005 waren 80% aller eMails weltweit SPAM. Zudem enthielten 15% von diesen eMails gefährliche Viren. Der Wirtschaft entsteht ein jährlicher Verlust von ca. 40 Milliarden Euro. Dieser berechnet sich auf Grund verminderter Produktivität der Mitarbeiter und erhöhten Netzwerkkosten. Ein normaler Bürger hat einen geschätzten Jahresverlust von 10 Tagen alleine durch das Löschen der SPAM-Nachrichten aus dem Posteingang (aus Carpinter, 2005). Dabei machen die schnellwechselnden Charakteristika von SPAM das Entwickeln von zuverlässigen Filtern sehr schwierig.

2.2 Verteidigungsmöglichkeiten

2.2.1 Bisherige Techniken

Die bisher existierenden Techniken, um SPAM zu filtern sind meist auf statischen, regulären Ausdrücken basierend. Ein anderer Ansatz besteht darin die Absender-Adresse in so genannten *Black-Listen* abzufragen. Dies sind Listen von eMail-Adressen, von denen bereits SPAM versendet wurde. Jedoch sind alle diese Techniken weit entfernt von einer perfekten Performance.

2.2.2 Weitere Möglichkeiten

Neben den genannten Methoden SPAM zu filtern, gibt es weitere Überlegungsansätze, ungewünschten eMail-Versand schon im Keim zu ersticken. Dabei könnte man Kosten für den Versand von eMails einführen, oder das SMTP-Protokoll insofern adaptieren, dass der Versand nur noch mit Zertifikaten möglich ist. Auch gesetzliche Strafen gegen den Versand von SPAM könnten eingeführt, bzw. verschärft werden, jedoch sind all diese Möglichkeiten entweder fehlerbehaftet oder unrealistisch, weshalb sie wenig Anerkennung finden.

Eine praktikable und effektive Methode von unerwünschten eMails verschont zu bleiben sind so genannte *Machine Learning* Filter, die sich an die schnell wechselnden Charakteristika von SPAM in angemessener Zeit anpassen können.

2.2.3 Machine Learning Filter

Machine Learning Filter werden auf einer möglichst homogenen Menge von HAM und SPAM trainiert (erwünschte und unerwünschte eMails). Aus diesen eMails werden dann Attribute extrahiert, die für den Erkennungsprozess benutzt werden. Die unterschiedlichen Attribute werden in Kapitel 4.2 eingehender besprochen. Der Vorteil besteht darin, dass sobald eine neue Art von SPAM-eMails auftritt, ein bestehender SPAM-Filter mit einer Trainingsmenge der neuen eMails auf den neuesten Stand gebracht werden kann. Dabei hängt es von dem verwendeten Machine Learning Algorithmus ab, ob dieses Training inkrementell oder auf allen bisherigen eMails durchgeführt werden muss.

Seit 1998 gilt der Naive Bayes-Filter als etablierter Standard, an dem alle neuen Filter gemessen werden. In dieser Arbeit wird ein neuer Ansatz verfolgt, der einen **Ensemble Classifier** als Basis-Algorithmus verwendet.

Kapitel 3

Ensemble Classifier

Ein interessanter Aspekt auf dem Gebiet der künstlichen Intelligenz und speziell auf dem Gebiet der Machine Learning Algorithmen ist, dass zwei verschiedene Algorithmen, welche auf derselben Menge von Trainingsdaten erstellt wurden, unterschiedliche Ergebnisse liefern. Dies liegt zum größten Teil an den Unterschieden in der Verarbeitung der Daten und der Berechnungsweise der verschiedenen Algorithmen. Um dies effizient zu nutzen, werden einzelne Algorithmen zu einem so genannten *Ensemble* zusammengeführt. Dabei werden Klassifizierungsanfragen immer von allen beteiligten Algorithmen bearbeitet und deren Ergebnisse anschließend auf Basis eines vorher festgelegten Berechnungsalgorithmus gemittelt. Dieser Abschnitt beschäftigt sich mit den Vorteilen eines Ensemble Klassifizierers gegenüber den individuellen Klassifizieren und stellt einige Verfahren zur Berechnung des Gesamtergebnisses vor.

3.1 Individual vs. Ensemble

Ein individueller Klassifizierer besitzt nur ein Modell zur Klassifikation und kann somit nur in den Grenzen dieses Modells diese durchführen. Deshalb bietet ein individueller Klassifizierer nicht auf allen Bereichen der Suchdomäne die gleiche Effektivität.

Ein Ensemble-Klassifizierer hingegen besitzt mehrere individuelle Modelle zur Klassifikation, wobei ein Meta-Algorithmus entscheidet, welcher Algorithmus auf welchem Bereich der Suchdomäne bei der Gesamtklassifikation benutzt wird und kann somit die Effektivität der Gesamtklassifikation verbessern. Die Vorteile lassen sich in drei Kategorien einordnen.

- Statistische Sicht

Durch das Zusammenführen der beteiligten Modelle werden deren Ergebnisse gemittelt. Somit werden falsche Teilergebnisse aus dem Endergebnis gestrichen (vgl. Abb. 3.1).

- Computationale Sicht

Viele Machine Learning-Algorithmen geben keine Garantie, das optimale Ergebnis zu finden. Sie finden meist lokale statt globale Maxima. Wird jedoch

die Suche in verschiedenen Bereichen der Suchdomäne gestartet, erhöht sich die Chance, die optimale Lösung - das globale Maximum - zu finden (vgl. Abb. 3.2).

- Repräsentative Sicht

Es ist möglich, dass das gesuchte (optimale) Ergebnis nicht im Ergebnisbereich eines einzelnen Algorithmus liegt. Eine (gewichtete) Summe innerhalb des Ensembles kann den Suchbereich erweitern und somit eine größere Domäne abdecken, als alle beteiligten individuellen Algorithmen (vgl. Abb. 3.3).

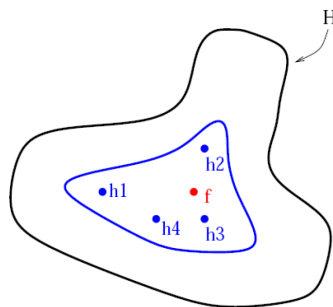


Abbildung 3.1: statistische Sicht

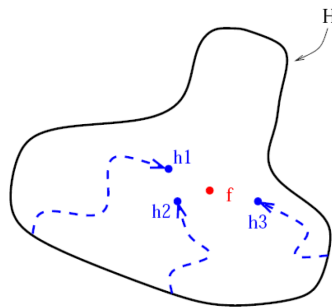


Abbildung 3.2: computationale Sicht

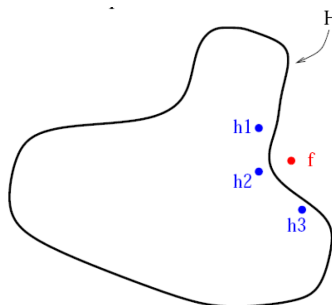


Abbildung 3.3: repräsentative Sicht

3.2 Beteiligte Machine Learning Algorithmen

In diesem Abschnitt werden die gebräuchlichsten Machine Learning Algorithmen aufgezählt, welche bei der Erstellung des LBEC eine Rolle spielen.

- k-Nearest-Neighbour (kNN)
- Artificial Neuronal Networks (ANN)
- Decision Trees (DT)
- Support Vector Machines (SVM)
- Naive Bayes (NB)

3.3 Ensemble-Erstellung

Die Erstellung eines Ensembles ist ein zweistufiger Prozess. Im ersten Schritt wird eine Bibliothek aus Machine Learning Modellen erstellt, welche aus unterschiedlichen Algorithmen bestehen, die jeweils mit verschiedenen Parametern initialisiert wurden. Dies ist ein sehr rechenintensiver Prozess, da das Training der verschiedenen Modelle sehr lange dauert. Der zweite Schritt besteht aus der Erstellung des Ensemble-Klassifizierers. Dabei werden - meist durch eine *forward stepwise selection* - Modelle aus der Bibliothek ausgewählt und dem Ensemble hinzugefügt (vgl. PseudoCode). Bei diesem Prozess kann jedoch das so genannte *Overfitting* auftreten. Overfitting bedeutet, dass das Ensemble zu sehr auf die Trainingsinstanzen spezialisiert wird.

- (1) Initialisierung eines leeren Ensembles.
- (2) Hinzufügen des Modells aus der Bibliothek, welches die Leistung des Ensembles auf der Validierungsmenge maximiert.
- (3) Wiederholung von (2) bis kein Modell mehr hinzugefügt werden kann, welches die Leistung des Ensembles erhöht.

Ist das Ensemble erstellt, kann es anschließend wie ein individueller Klassifizierer zur Klassifikation benutzt werden.

3.3.1 Methoden zur Bibliothekserstellung

Es gibt verschiedene Ansätze, die Bibliothek eines Ensemble-Klassifizierers zu erstellen. Bei dem so genannten *Bagging* (**bootstrap aggregation**) werden die Trainingsdaten der Modelle verändert. Dabei werden in jedem Iterationsschritt dem zu trainierenden Modell eine zufällige Teilmenge der Trainingsdaten als Basis zur Verfügung gestellt. Dies eignet sich sehr gut für instabile Machine Learning Algorithmen, wie z.B. DT, ANN, kNN.

Das *Boosting* verändert ebenfalls die Trainingsdaten der einzelnen Modelle. Aber anstatt die Größe der Trainingsmenge zu verändern, werden Gewichtungen in jedem Trainingsschritt vergeben um somit die Fehlerrate zu minimieren. Falsch-klassifizierte Daten werden “geboostet”, d.h. die Gewichtung wird Erhöht und die Gewichtung der richtig-klassifizierten Daten wird verringert.

Eine weitere Art der Bibliotheksberechnung besteht darin, die Attributmenge der Trainingsdaten während des Trainingsprozesses abzuändern. Dabei werden Modelle nur auf Teilmengen der Attribute (Eingabe-Attribute) trainiert. Dies ist jedoch nur dann sinnvoll, wenn die Attributmengen redundant sind.

Es ist auch möglich, die Output Vektoren (Ausgabe-Attribute) so umzuformen, dass Multi-Klassen-Daten in Binärklassen umgeformt werden und die Modelle nur auf diesen Daten trainiert werden.

3.3.2 Methoden zur Ensemble-Erstellung

Bei der Erstellung eines Ensembles wird unterschieden zwischen deterministischen und indeterministischen Verfahren. Dies ist davon abhängig, ob der verwendete Algorithmus zur Auswahl der einzelnen Modelle mit Zufallswerten initialisiert wurde oder nicht.

Deterministische Auswahl

Die *SELECTIONWITHOUTREPLACEMENT*-Methode entspricht der bereits beschriebenen *forward stepwise selection*, bei der jeweils das Modell dem Ensemble hinzugefügt wird, welches dessen Leistung maximiert. Dabei kann ein Modell nur einmal hinzugefügt werden.

SELECTIONWITHREPLACEMENT bezeichnet die gleiche Vorgehensweise wie bei der vorangegangenen Methode. Hierbei können nun jedoch auch Modelle mehrfach dem Ensemble hinzugefügt werden, falls der Algorithmus dies für sinnvoll erachtet. Die *BESTN*-Methode initialisiert ein Ensemble bereits mit N Modellen. Dabei muss vorher jedoch festgelegt werden, nach welchen Kriterien diese Modelle ausgesucht werden.

Indeterministische Auswahl

Bei den Methoden der indeterministischen Auswahl beschränkt man sich darauf, die deterministischen Auswahlverfahren auf zufälligen Teilmengen der Bibliothek anzuwenden. Es werden somit die gleichen Algorithmen wie bei der deterministischen Auswahl verwendet, jedoch auf einer randomisierten Menge von Modellen.

3.3.3 Ensembleberechnung

Ein Standardverfahren bei der Ensembleberechnung ist das *majority voting*, bei dem die einfache Mehrheit der beteiligten Modelle die Klassifikation bestimmt.

Dem gegenüber stehen weitere, kompliziertere Verfahren, wie z.B. das *Bayesian voting*, bei dem die Ergebnisse des Ensembles auf Basis von Bayesschen bedingten Wahrscheinlichkeiten berechnet werden. Dieses Verfahren erzielt jedoch nur optimale Ergebnisse, falls das richtige Modell, welches die Daten beschreibt aus der Bibliothek gewählt wurde.

3.4 Anwendung eines LBEC als SPAM-Filter

In diesem Abschnitt wird der von Caruna u. a., 2004 entwickelte SPAM-Filter auf Basis eines LBECs erläutert und deren Ergebnisse reflektiert.

3.4.1 Durchführung

Der entwickelte LBEC wurde mit Hilfe des WEKA-Toolkits (Witten u. Frank, 2005) implementiert. Dabei handelt es sich um ein in JAVA geschriebenes Toolkit, welches es ermöglicht mit relativ einfachen Mitteln Machine Learning Programme zu entwickeln. Der entwickelte LBEC wurde als Plugin für WEKA implementiert.

Korpusverarbeitung

Der verwendete eMail-Korpus wurde in die Klassen HAM und SPAM aufgeteilt - erwünschte bzw. unerwünschte eMails. Als Klassifikations-Attribute wurde die Wortliste der eMails benutzt. Dazu war eine Vorverarbeitung der eMails notwendig. Zunächst wurde der Nachrichtentext aus den eMails extrahiert und die einzelnen Wörter auf ihren Wortstamm reduziert. Des Weiteren wurden mit Hilfe einer Stop-Wort-Liste diejenigen Wörter herausgefiltert, von denen man wusste, dass sie die Ergebnisse der Klassifikation nicht positiv beeinflussen würden. Dies war deshalb sinnvoll, damit die Anzahl der Attribute begrenzt werden konnte und somit eine höhere Aussagekraft in den Wahrscheinlichkeiten erzielt werden konnte. Zudem wurde mit Hilfe der *Information Gain*-Methode die Anzahl der Attribute begrenzt. Der Information Gain berechnet die Bedeutung der einzelnen Attribute und erstellt ein Ranking auf Grund dieser Daten. Ein Threshold begrenzt anschließend die Attributmenge. Für den LBEC wurden maximal ca. 500 Attribute zur Klassifikation herangezogen.

3.4.2 Ergebnisse

Die Klassifikationsergebnisse des LBEC, sowie Vergleichs-Klassifizierern, kann in Tabelle 3.1 eingesehen werden. Mit Worst IC ist der schlechteste individuelle Klassifizierer gemeint, der nur eine Trefferquote von knapp 67 % erreichte. Dies ist gefolgt von dem Ergebnis des besten individuellen Klassifizierers, der bei der Klassifikation mit einer Trefferwahrscheinlichkeit von 90,50 % beteiligt war. Noch besser ist der Naive Bayes Klassifizierer, der heutzutage als Standard angesehen wird, hier mit einer Trefferquote von 94,80 %. Als bester Klassifizierer stellte sich jedoch der LBEC heraus, der mit einer Wahrscheinlichkeit von annähernd 97 % einen deutlichen Abstand nicht nur zu dem Naive Bayes Klassifizierer, sondern auch zu dem

besten individuellen Klassifizierer besitzt. Dies lässt sich mit den in Abschnitt 3.1 aufgeführten Vorteilen eines Ensemble-Klassifizierers gegenüber seinen individuellen Teilklassifizierern erklären.

Klassifizierer	Ergebnis
LBEC	96,90 %
Bayes	94,80 %
Best IC	90,50 %
Worst IC	66,90 %

Tabelle 3.1: Ergebnisvergleich der Klassifikation

Kapitel 4

Implementierung eines lernfähigen SPAM-Filters

In diesem Kapitel wird die Implementierung eines SPAM-Filters beschrieben, der mit Machine Learning Klassifikatoren die eMails filtert. Dies ist ein Projekt, welches im Rahmen des dieser Ausarbeitung zu Grunde liegenden Seminars durchgeführt wurde.

4.1 Durchführung

Der diesem Kapitel zu Grunde liegende SPAM-Filter wurde mit Hilfe des WEKA-Toolkits zur Erstellung von Machine Learning Klassifikatoren erstellt. Der SPAM-Filter wurde komplett in JAVA implementiert und ist in vier Packages aufgeteilt: *start*, *gui*, *data*, *classifiers*. Das *start*-Package enthält die Klasse "Start", mit deren Hilfe die Applikation gestartet wird. Es wird die graphische Benutzeroberfläche aus dem *gui*-Package geladen (vgl. Screenshot 4.1). Da das Ziel war, den Klassifizierer auf Text-Dateien arbeiten zu lassen, wurde auf das Herunterladen der eMails vom Server verzichtet. Ein neues Verzeichnis kann unter dem Menüpunkt "File" → "Open Directory" geöffnet werden. Die dort enthaltenen eMails werden anschließend geöffnet und - falls ein Klassifizierer bereits geladen wurde - vorklassifiziert.

Das Verarbeiten und Einlesen der eMails wird von dem *data*-Package verwaltet und wurde nach dem entsprechenden RFC-Format 822 entwickelt.

Das *classifiers*-Package enthält alle Prozeduren zum Training und Einsetzen von Klassifizierern. Dabei wurde neben den bekannten Klassifizierern (Decision Tree, Neuronal Network, k-Nearest-Neighbour, Naive Bayes und Support Vector Machine) auch ein *Meta*-Klassifizierer entwickelt und eingebunden. Dieser enthält drei individuelle Klassifizierer, welche das Gesamtergebnis durch ein einfaches majority voting berechnen. Der Decision Tree wurde auf Basis des WEKA-Klassifizierers 'J48' implementiert, der Neuronale Netzwerk-Klassifizierer mit Hilfe der 'MultilayerPerceptron'-Klasse. Der k-Nearest-Neighbour-Klassifizierer wurde durch die Klasse 'IBk' aus WEKA implementiert, sowie der Naive Bayes-Klassifizierer mit Hilfe der Klasse 'Naive-BayesUpdateable' realisiert wurde. Die Support Vector Machine wurde mit Hilfe des

'SMO'-Algorithmus implementiert.

Bei der Entwicklung des SPAM-Filters wurde besonderen Wert auf das inkrementelle Trainieren der Klassifizierer gelegt. Es ist möglich den Filter mit neuen Informationen über eMails an die eigenen Bedürfnisse anzupassen.

4.2 Attribute

Als (Klassifikations-)Attribute werden die Charakteristika bezeichnet, mit denen Machine Learning Algorithmen arbeiten, bzw. auf welchen Informationen sie trainiert werden und klassifizieren. Neben den auf Wortvorkommen basierenden Attribute des Body- & Subject-Textes wurden hierbei auch einige weitere Attribute eingefügt:

- hasHTML - Ist die eMail im HTML-Format?
- hasNOToField - Fehlt das To: Feld?
- hasJScriptTags - Sind JavaScript-Tags im HTML-Code vorhanden?
- hasFrameTags - Sind Frame-Tags im HTML-Code vorhanden?
- hasFormTags - Sind Form-Tags im HTML-Code vorhanden?
- hasNOMessageID - Fehlt die MessageID?
- numberOfRecipients - Die Anzahl der Empfänger.
- percOfUpperCaseLetters - Das Verhältnis zwischen Groß- & Kleinbuchstaben.

4.3 eMail-Korpus

Die Vorgabe war, eMails mit englischem Text zu verarbeiten. Dies erschwerte die Aufgabe, da es nicht möglich war, eMails aus den privaten Postfächern zum Trainieren der Klassifikatoren zu verwenden, da diese größtenteils in deutscher Sprache vorlagen und somit zum Training unbrauchbar waren. Genauer gesagt bestand das Problem darin englische HAM-eMails zu finden, da die meisten SPAM-eMails generell in englisch verfasst sind. Als frei zugänglicher englischer HAM-Korpus bot sich der ENRON-Korpus an. ENRON war ein amerikanischer Energiekonzern, der Konkurs beantragte und die eMails mancher Manager von ENRON im Netz veröffentlicht wurden (link: <http://www.cs.cmu.edu/~enron/>). Dieser Korpus enthält ca 500.000 eMails in einem 400 MByte TAR-File. In diesem Korpus sind zudem alle Dateianhänge vorhanden, welche jedoch für die weitere Verarbeitung im Rahmen dieser Arbeit - im Vorverarbeitungsschritt - entfernt wurden.

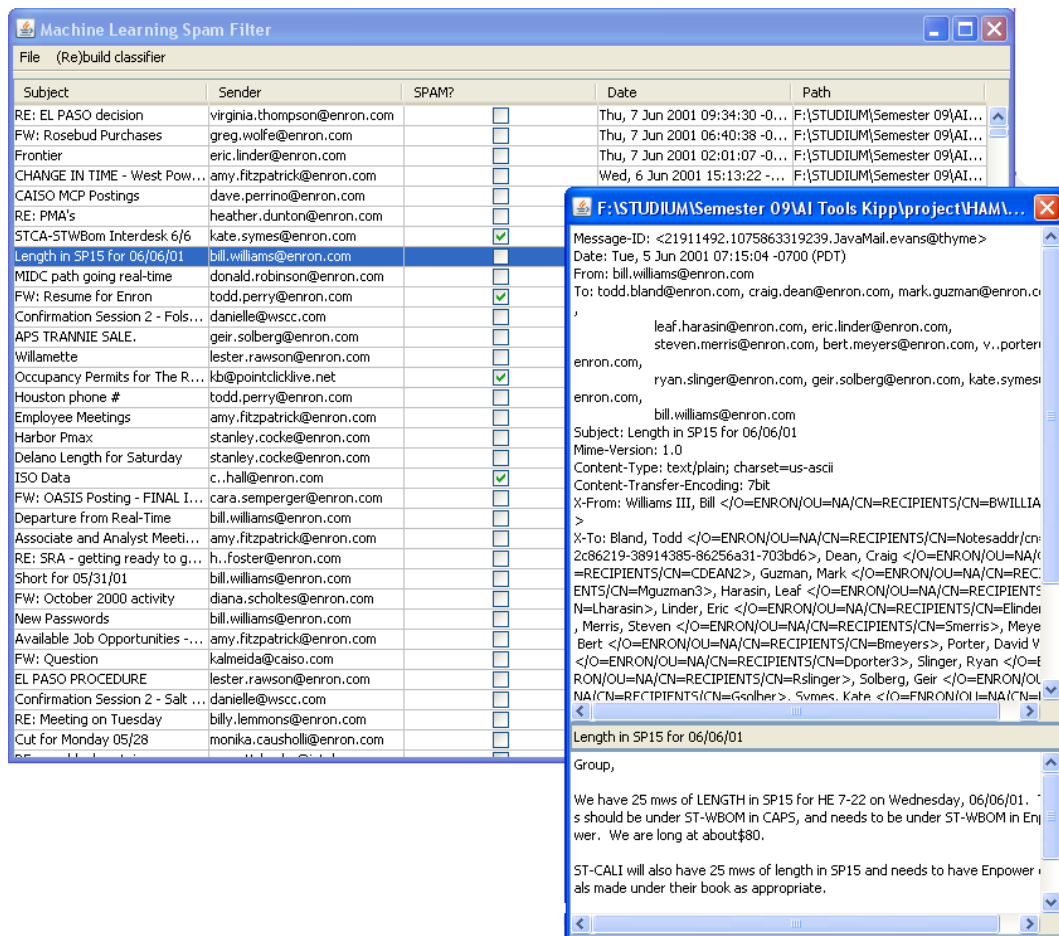


Abbildung 4.1: Screenshot: SPAM-Filter

4.3.1 Vorverarbeitung

Der Korpus befand sich in einem sehr guten Zustand und bedurfte nur geringer Bearbeitung. Die Zeilenumbrüche mussten mit Hilfe des Tools *unix2dos* in ein Windows-fähiges Format transformiert werden. Zudem waren alle eMails in verschiedene Verzeichnisse unterteilt und dort jeweils durchnummeriert. Dies machte es notwendig, die Dateien umzubenennen, damit beim Kopieren in ein einziges Trainingsverzeichnis keine Probleme mit doppelten Dateinamen auftreten.

4.3.2 Aufteilung

Natürlich konnte nicht der komplette Korpus zum Training der Klassifizierer herangezogen werden, da dies einen sehr hohen Zeitaufwand im Training, sowie beim späteren Klassifizieren mit sich bringen würde. Als gutes Mittelmaß stellte sich eine Trainingsmenge von 2.000 HAM-, sowie SPAM-eMails heraus. Es musste dabei sehr auf die Homogenität der Menge geachtet werden, da sonst die Ergebnisse leicht verfälscht werden können. So wurden die 2.000 HAM-eMails randomisiert dem Korpus entnommen und die SPAM-eMails ebenfalls willkürlich aus dem eigenen SPAM-Postfach extrahiert.

4.4 Ergebnisse

In Tabelle 4.1 sind die Ergebnisse der Klassifikator-Evaluation aufgeführt. Trainiert wurde auf einem AMD64 3800+ mit 2 GByte Arbeitsspeicher. Die Abkürzungen der Klassifizierer sind wie nachfolgend erläutert.

- SVM - Support Vector Machine
- DT - Decision Tree
- Meta - Meta Classifier
- NN - Neuronal Networks
- kNN - k-Nearest Neighbour
- NB - Naive Bayes

Hierbei ist anzumerken, dass der Meta-Klassifizierer im Rahmen des Seminars neu implementiert wurde und lediglich auf Basis eines DT-, eines kNN- und eines SVM-Klassifizierers mit einem Majority Voting entwickelt wurde. Letzteres erklärt auch, warum das Training des Meta-Klassifizierers dreimal länger dauerte als alle anderen. Der NN-Klassifizierer fällt direkt ins Auge, da dieser deutlich länger zum Trainieren benötigte als alle anderen Klassifizierer. Dies ist ein oft beobachtetes Problem bei Neuronalen Netzen und liegt in der aufwändigeren Berechnung der NN begründet. Die Dauer der Klassifikation betrug bei 400 Test-eMails wenige Sekunden.

In der Tabelle ist ersichtlich, dass der Meta-Klassifizierer schlechtere Ergebnisse erzielte, als die individuellen SVM- und DT-Klassifizierer. Dies lässt sich dadurch

erklären, dass ein einfaches Majority Voting zur Ensemble-Berechnung benutzt wurde und der kNN-Klassifizierer höchstwahrscheinlich die Ergebnisse dadurch negativ beeinflusst hat. Man könnte die Ergebnisse durch ein gewichtetes Majority Voting wahrscheinlich verbessern.

Klassifizierer	Dauer des Trainings [min]	Evaluationsergebnisse [%]
SVM	3	95,600
DT	3	94,725
Meta	9	94,050
NN	20	92,825
kNN	3	91,825
NB	3	86,650

Tabelle 4.1: Evaluation der Klassifizierer

Kapitel 5

Konklusion

Dieses Kapitel liefert eine kurze Zusammenfassung über die in der Arbeit angesprochenen Themen und gibt einen Ausblick auf die heutige Anwendung der vorgestellten Methoden innerhalb der SPAM-Klassifikation.

5.1 Zusammenfassung

In der heutigen Zeit muss die Wirtschaft Milliardenverluste durch SPAM-Nachrichten in Kauf nehmen. Dies berechnet sich auf Grund verminderter Produktivität der Mitarbeiter sowie erhöhten Netzwerk- und Wartungskosten. Um dies zu verhindern ist man an besonders leistungsfähigen eMail-Klassifizierern interessiert, welche mit hoher Genauigkeit - ohne Eingreifen des Benutzers - unerwünschte von erwünschten eMails trennen und entsprechend verwalten.

Die bereits etablierte Methode der Machine Learning SPAM-Filter wurde in dieser Arbeit genauer betrachtet und vertiefend wurde sich mit dem Klassifikations-Typ des Library Based Meta-Klassifiziers bzw. Library Based Ensemble-Klassifizierer beschäftigt. Bei diesem werden aus einer Vielzahl unterschiedlicher Klassifizierer einzelne ausgewählt und zu einem Klassifikationsverbund zusammengeschlossen. Eine vorher festgelegte Metrik berechnet dann das Gesamtergebnis der Klassifikation auf Basis der Teilergebnisse der beteiligten Klassifizierer (vgl. Kapitel 3). Auf diese Art können genauere Ergebnisse der Klassifikation - gegenüber dem Naive Bayes-Klassifizierer mit 94,8 % Erkennungsleistung - auf annähernd 97 % (96,9 %) beobachtet werden.

Des Weiteren wurde während dem dieser Arbeit zu Grunde liegenden Seminars ein Machine Learning eMail-Klassifikaitons-Tool auf Basis des WEKA-Toolkits erstellt. Die bei der Durchführung aufgetretenen Probleme und Lösungen wurden erläutert und die Klassifikationsergebnisse sowohl präsentiert als auch vorgestellt.

5.2 Ausblick

Machine Learning Algorithmen gewinnen immer mehr an Bedeutung auf dem Gebiet der SPAM-Filter. Angedachte Ideen eines per-eMail-fees oder Gesetze gegen die Ver-

sendung von SPAM sind zwar theoretisch durchdachte Lösungen, deren Praktikabilität jedoch fraglich erscheint. Zudem sind Machine Learning Filter, die sich adaptiv an den jeweiligen Benutzer anpassen von sehr hohem Nutzen, da der Benutzer selbst so wenig wie möglich von der aufwändigen Klassifikation mitbekommen soll. Heutige Klassifizierer arbeiten bereits in Ergebnisbereichen über der 95 % Marke und liefern somit sehr gute Ergebnisse ab. Ein großes Problem sind die so genannten “false-positives” - jene HAM-eMails, welche als SPAM klassifiziert wurden. Dies mag im privaten Bereich nicht sonderlich zu Last fallen, in wirtschaftlichen Betrieben kann dies jedoch sehr schnell zu hohen Verlusten führen. Dies ist noch ein Faktor an dem dringend gearbeitet werden sollte und auch aktuell gearbeitet wird.

Literaturverzeichnis

Carpinter 2005

CARPINTER, James M.: Evaluating ensemble classifiers for spam filtering / University of Canterbury, Christchurch, New Zealand. 2005. – Forschungsbericht

Caruna u. a. 2004

CARUNA, Rich; NICULESCU-MIZIL, Alexandru; CREW, Geoff ; KSIKES, Alex: Ensemble Selection from Libraries of Models. In: *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004, S. 137–144

Clayton 2004

CLAYTON, Richard: Stopping SPAM by Extrusion Detection / University of Camebridge. 2004. – Forschungsbericht

zeitform Internet Dienste 2003

INTERNET DIENSTE zeitform: *Strategien gegen Spam - Ein Überblick*. 2003

Koj u. a. 2006

KOJ, Magdalena; MALEIKA, Tatsiana ; DANILAVA, Sviatlana: Inkrementelle Thesauri am Beispiel von Spam- und Phishing-Mails / JW Goethe Universität Frankfurt am Main. 2006. – Forschungsbericht

Petković u. a. 2005

PETKOVIĆ, Tomislav; KOSTANJČAR, Zvonko ; PALE, Predrag: E-Mail System for Automatic Hoax-Recognition. In: BUDIN, Leo (Hrsg.); RIBARIĆ, Slobodan (Hrsg.): *MIPRO 2005 XXVII. International Convention* Bd. CTS & CIS. Opatija, Croatia, Juni 2005. – ISBN 953–233–012–7, S. 117–121

Witten u. Frank 2005

WITTEN, Ian H.; FRANK, Eibe: *Data Mining: Practical Machine Learning Tools and Techniques*. 2. San Francisco: Morgan Kaufmann, 2005