

Mit KI gegen Spam

Florian Laib

Proseminar Künstliche Intelligenz im Sommersemester 2006

Abstract. Diese Ausarbeitung beschreibt zwei Ansätze des Fallbasierten Schließens im Zusammenhang mit der Klassifizierung von E-Mail Nachrichten in unerwünschte kommerzielle E-Mails, auch Spam genannt, und erwünschte bzw. Nicht-Spam Nachrichten und deren Vergleich mit dem bisher bevorzugten Klassifizierungsverfahren, dem naiven Bayes-Verfahren.

Aufgrund der sich kontinuierlich ändernden Spam-Nachricht muss sich ein Spam-Filter ständig an diese anpassen können und es ist daher davon auszugehen, dass ein Fallbasiertes Verfahren gut dafür geeignet sein könnte.

Beim Fallbasierten Filtern von E-Mail Nachrichten verzögert sich die Klassifizierung bis zur Laufzeit. Das bedeutet, dass die Trainingsdaten kontinuierlich aktualisiert werden können und die neuen Daten sofort für die Klassifizierung nutzbar sind.

Die Untersuchung beinhaltet verschiedene Verfahren zur Gewichtung der Attribute und Abstände, sowie der Größe der Nachbarschaft (*neighborhood size*) und der Dimension (*dimensionality*).

Im zweiten Ansatz wird außerdem das Verhalten der Filter über einen Zeitraum von einem Jahr mit Aktualisierung der Trainingsdaten betrachtet.

Die Ergebnisse zeigen, dass ein Fallbasierter Filter praktisch durchführ- und anwendbar und nicht schlechter als ein naiver Bayes-Filter ist. Außerdem zeigt sich, dass ein Fallbasierter Filter die sich über die Zeit ändernden Spam-Nachrichten besser erkennt als ein naiver Bayes-Filter.

1 Einleitung

In dieser Ausarbeitung werde ich zwei neue Ansätze des Fallbasierten Schließens beziehungsweise Lernens im Bereich der Filterung von unerwünschten kommerziellen E-Mails, auch Spam genannt, vorstellen und mit dem bisher bevorzugten naiven Bayes-Verfahren vergleichen.

Spam Klassifizierung wird heutzutage aufgrund mehrerer Gründen immer wichtiger. Die steigende Beliebtheit und Nutzung von E-Mails, sowie die niedrigen Kosten, haben aber auch dazu geführt, dass immer mehr unerwünschte Werbung mit E-Mails verschickt wird.

Diese Spam-Nachrichten werben für eine Vielzahl von Produkten, von Reisen über Medikamente bis hin zu Viagra oder pornographischen Inhalten. Spam-Nachrichten sind für die meisten Benutzer sehr ärgerlich, da sie ihre Zeit verschwenden, sowie das Herunterladen der E-Mails verlängern und mittlerweile eine sehr große Bandbreite des Internets verschwenden.

Außerdem wird das Internet und somit auch E-Mails in sehr großem Maße auch von Kindern und Jugendlichen verwendet, die dann unter Umständen auch Spam-Nachrichten mit pornographischem Inhalt zugeschickt bekommen.

Heute scheint es so, dass sich die Situation noch verschlechtert und ohne geeignete Gegenmaßnahmen könnte schlussendlich die Benutzbarkeit von E-Mails stark beeinträchtigt werden. Deshalb gibt es mittlerweile schon fast so etwas wie ein „Wettrüsten“ zwischen den so genannten Spammern, also den Versendern dieser unerwünschten Werbung, und den Firmen die kommerzielle Spam-Filter herstellen.

Dieses „Wettrüsten“ möchte ich hier anhand von ein Paar Beispielen einmal etwas anschaulich verdeutlichen: Im Grunde gibt es hier 2 Möglichkeiten für Spammer um die Filter zu täuschen. Zum einen verändern Spammer Wörter, die dazu führen würden, dass die Nachricht als Spam eingestuft würde. Zum anderen fügen sie Worte in die Spam-Nachricht ein, die zwar für den Filter „lesbar“ sind, aber nicht für den Anwender und gleichzeitig dazu führen, dass die Nachricht als Nicht-Spam Nachricht eingestuft wird.

Der einfachste Trick ein Wort so zu verändern, dass es nicht mehr als Spam erkannt werden kann, ist das Einfügen von Leerzeichen oder anderen Zeichen. So wird aus Viagra z.B. V i a g r a, oder V*i*a*g*r*a. Dies kann zugegebenermaßen sehr einfach von Filtern erkannt werden, wenn man sie entsprechend dafür programmiert.

Deshalb haben die Spammer HTML für sich entdeckt. Wie in jeder anderen Programmiersprache kann man auch in HTML Kommentare in den Quelltext einbinden, die natürlich nicht in der eigentlichen Nachricht angezeigt werden.

So wird aus Viagra dann `V<!--Kommentar-->i<!--Kommentar-->a<!--Kommentar-->g<!--Kommentar-->r<!--Kommentar-->a`. Viele Filter erkennen das Wort Viagra dann nicht mehr, obwohl es weiterhin in der Nachricht als Viagra vorkommt. Noch schlimmer ist aber, dass einige Filter auch die Kommentare lesen können, und bei geeigneten Kommentaren, so die Nachricht als Nicht-Spam einstufen könnten.

Da dieser Trick sehr weit verbreitet ist, können neuere Filter mittlerweile Kommentare ignorieren oder das bloße Vorhandensein von Kommentaren als Kriterium zur Einstufung als Spam benutzen. Welcher Anwender schreibt schon Kommentare in eine E-Mail?

Eine weitere weit verbreitete Technik ist es Leerzeichen oder Buchstaben mit Schriftgröße 0 oder 1 in ein Wort einzufügen. So wird aus Viagra dann `Vziagra` und viele Filter erkennen dann Viagra nicht mehr, obwohl in der eigentlichen Nachricht weiterhin Viagra steht, da das „z“ fast unsichtbar ist.

Kommen wir nun zum zweiten großen Bereich, dem Hinzufügen von Wörtern die dazu führen sollen, dass die Nachricht als Nicht-Spam eingestuft wird.

Spammer schreiben dazu weißen Text auf weißen Hintergrund um den Nicht-Spam Text für den Leser „unsichtbar“ zu machen, während der Filter ihn gleichzeitig immer noch analysiert. Seitdem dies von Filtern erkannt wird, nehmen Spammer einfach zwei ähnlichen Farben anstatt immer nur weiß. Gute Filter können auch dies erkennen. Eine weitere Möglichkeit besteht darin den Nicht-Spam Text in einem sehr kleinen Fenster, von z.B. 8x8 Pixel, zu verstecken.

Es gibt noch eine Vielzahl von weiteren Techniken und Tricks, die die Spammer für sich entdeckt haben.

Die Filter stehen diesen aber nicht machtlos gegenüber. Je mehr der Spammer versucht seine Nachricht so zu verändern, dass sie nicht mehr von Filtern als Spam erkannt wird, desto eher ist sie als solche identifizierbar. Denn welcher normale Anwender benutzt schon Kommentare oder Buchstaben mit Textgröße 0?

Bisher typischerweise benutzte Spam-Filter haben eine „Schwarze Liste“ (*blacklist*) mit bekannten Spammern sowie handgemachte Regeln, die Nachrichten blockieren, die bestimmte Wörter oder Satzteile enthalten. Sowohl die „Schwarze Liste“ als auch die Regeln sind problematisch. Die „Schwarze Liste“ ist nicht sehr effektiv, da Spammer meistens gefälschte Adressen benutzen und die Regeln müssen speziell an die eingehenden Nachrichten eines Benutzers oder Gruppe von Benutzern angepasst werden um am besten zu arbeiten. Dieses aber benötigt Zeit und Fachkräfte und muss periodisch wiederholt werden, da sich die Spam-Nachrichten sehr schnell verändern um ihrerseits wiederum an den Filtern vorbeizukommen.

Hier kommt das Gebiet der Künstlichen Intelligenz ins Spiel. Wäre es nicht viel einfacher, schneller und günstiger, wenn der Spam-Filter automatisch auf diese Veränderungen reagieren könnte?

Dafür müsste ein Filter die Fähigkeit haben zu Lernen, d.h. also der Filter müsste anhand von Beispielen gewisse Gesetzmäßigkeiten erkennen, um so eine neue Nachricht, auch wenn diese sich im Laufe der Zeit ändert, als Spam oder Nicht-Spam zu klassifizieren. Das heißt also, der Filter lernt gewissermaßen zu Verallgemeinern. Dabei besteht der einzige manuelle Aufwand darin, dem Filter eine Menge von sowohl Spam, als auch Nicht-Spam Nachrichten als Beispiele zu geben. Danach lernt der Filter automatisch und kann so die ständigen Veränderungen des Aufbaus von Spam-Nachrichten erfassen. Außerdem kann er so vollautomatisch an jeden Benutzer und dessen individuelle Nachrichten angepasst werden.

Das bisher am häufigsten benutzte lernende Verfahren, ist das so genannte naive Bayes-Verfahren (Naive Bayes). Dieses Verfahren gehört zu der Gruppe der „*eager learning systems*“. Diese bilden anhand von Beispielen ein Modell um danach auch zuvor nicht gesehene Nachrichten zu klassifizieren.

Da sich Spam-Nachrichten kontinuierlich verändern nimmt man an, dass die „*lazy learning systems*“ besser für Spam-Filter geeignet sein könnten. Bei diesen wird die Klassifizierung bis zur Laufzeit verschoben. Deshalb werde ich nun die zwei „*lazy learning systems*“ *TiMBL* und *ECUE* und ihre Ergebnisse vorstellen.

Der weitere Aufbau dieser Ausarbeitung ist wie folgt: Im zweiten Kapitel werde ich die Grundlagen für die beide Ansätze sowie das naive Bayes-Verfahren vorstellen. Im dritten Kapitel wird das Verfahren des Fallbasierten Schließen mit den beiden Ansätzen vorgestellt und im vierten und letzten Kapitel gibt es eine Zusammenfassung.

2 Grundlagen

2.1 Message Representation

In beiden Ansätzen werden die Nachrichten als Vektor $\vec{x} = (x_1, x_2, \dots, x_n)$ dargestellt, wobei x_1, \dots, x_n die Werte der Attribute X_1, \dots, X_n sind. Im ersten Ansatz, der den so genannten *TiMBL* Filter vorstellt, sind alle Attribute

binär dargestellt und $X_i = 1$ bedeutet, dass das Attribut in der Nachricht vorkommt und $X_i = 0$, dass das entsprechende Attribut nicht in der Nachricht vorkommt.

Im zweiten Ansatz, der den so genannten *ECUE* Filter vorstellt, werden die Attribute sowohl binär, wie eben beschrieben, als auch numerisch dargestellt. D.h. es wird nicht nur analysiert ob ein Attribut in der Nachricht vorkommt, sondern auch noch festgehalten, wie oft das Attribut in der Nachricht auftritt.

2.2 Information Gain

Information Gain (IG) oder Informations-Zuwachs ist ein Verfahren um aus der Vielzahl an Attributen bzw. Merkmalen diejenige mit dem größten Informationsgehalt auszuwählen. Dadurch lassen sich die Attribute auf die für die Klassifizierung wichtigsten bzw. aussagekräftigsten Attribute reduzieren.

Laut Yang and Pedersen (1997) ist es möglich bis zu 98% der Attribute mittels Information Gain wegzulassen ohne die Klassifizierung zu verschlechtern, oder sie sogar zu verbessern.

2.3 K-fold stratified cross-validation

Das ist eine Technik, die die Resultate bei geringer Anzahl von Daten aussagekräftiger macht. Dazu werden die Daten zunächst in k gleich große Teile geteilt. Dann wird das Experiment k -Mal durchgeführt, wobei jedes Mal ein anderer Teil als Testdaten dient und die anderen $k-1$ Teile als Trainingsdaten.

2.4 False Positive und False Negative

Beim Filtern von Nachrichten können zwei grundsätzliche Fehler auftreten. Zum einen kann eine Nachricht irrtümlicherweise als Spam klassifiziert werden, was man auch mit *False Positive (FP)* bezeichnet. Zum anderen kann eine Nachricht irrtümlicherweise als Nicht-Spam klassifiziert werden, was man analog auch mit *False Negative (FN)* bezeichnet.

2.5 Naive Bayes-Verfahren

Dieses Verfahren ist das bisher bevorzugte Verfahren bei der Spam Klassifizierung.

Es beruht auf dem so genannten Bayes-Theorem des englischen Pastors Thomas Bayes.

Dieses bietet eine Lösung für Probleme, bei denen zwischen sich gegenseitig ausschließenden Behauptungen entschieden werden muss. Jeder der Behauptungen ist eine relative Häufigkeit zugeordnet, die auch als A-Priori-Wahrscheinlichkeiten bezeichnet werden.

Anhand von Daten, die auftreten, wenn eine bestimmte Behauptung zutrifft, und anhand ihrer Wahrscheinlichkeiten kann zusammen mit den A-Priori-Wahrscheinlichkeiten die Wahrscheinlichkeit berechnet werden, dass eine bestimmte Behauptung für ein neues beobachtetes Beispiel zutrifft. Diese Wahrscheinlichkeit wird auch A-Posteriori-Wahrscheinlichkeit genannt.

Als Formel ausgedrückt bedeutet dies $P(h | D) = \frac{P(D | h)P(h)}{P(D)}$. Hierbei ist D das neu beobachtete Beispiel

und $P(h | D)$ die Wahrscheinlichkeit, dass die Behauptung h stimmt, wenn das Beispiel D auftritt. $P(D | h)$ ist die Wahrscheinlichkeit, dass bestimmte Daten auftreten, wenn Behauptung h zutrifft und $P(D)$ ist die Wahrscheinlichkeit, dass das beobachtete Beispiel auftritt. Da sie bei jedem Problem konstant ist, kann sie in solchen Fällen auch weggelassen werden.

Beim naiven Bayes-Verfahren bzw. naiven Bayes-Klassifizierer wird die zu klassifizierende Nachricht nun auch als Menge von Attributen a_1, a_2, \dots, a_n dargestellt. C ist die Menge alle möglichen Zielklassen und c die gesuchte Klasse der neuen Nachricht. Die gesuchte Klasse c ist diejenige Klasse, zu der der Algorithmus die höchste Wahrscheinlichkeit ausgibt (argmax).

Mit dem Bayes-Theorem folgt also:

$$c = \arg \max_{c_i \in C} P(c_i | a_1, a_2, \dots, a_n)$$

$$c = \arg \max_{c_i \in C} \frac{P(a_1, a_2, \dots, a_n | c_i)P(c_i)}{P(a_1, a_2, \dots, a_n)}$$

$$c = \arg \max_{c_i \in C} P(a_1, a_2, \dots, a_n | c_i)P(c_i)$$

Diese Formel führt allerdings nur unter enormen Aufwand zum Ergebnis und wird deshalb durch folgende Annahme vereinfacht:

$$c = \arg \max_{c_i \in C} P(a_1, a_2, \dots, a_n | c_i) P(c_i)$$

$$c = \arg \max_{c_i \in C} P(a_1) P(a_2) \dots P(a_n) P(c_i)$$

$$c = \arg \max_{c_i \in C} P(c_i) \prod_j P(a_j | c_i)$$

Diese Annahme bedeutet, dass alle Attribute unabhängig voneinander auftreten müssen. Diese Annahme ist naiv, daher auch der Name des Verfahrens, da z.B. das Wort „freundliche“ sehr oft in Verbindung mit „Grüßen“ auftritt und diese also nicht unabhängig voneinander sind.

Trotzdem funktioniert das Verfahren im Allgemeinen sehr gut. Das Lernen erfolgt bei diesem Verfahren, indem die Wahrscheinlichkeiten $P(a_j | c_i)$ gespeichert und durch jede neu klassifizierte Nachricht modifiziert werden.

3 E-Mail Klassifikation durch Fallbasiertes Schließen

Das Verfahren des Fallbasierten Schließens werde ich hier nur kurz in den groben Zügen beschreiben, da es wie schon erwähnt in den beiden Ansätzen verwendet wird und also noch ausführlicher weiter unten behandelt wird. Bei diesem Verfahren hat man so genannte Trainings Daten (Nachrichten), welche Spam-Nachrichten und Nicht-Spam Nachrichten enthalten. Eine neue Nachricht wird nun dadurch einer der beiden Klassen, also entweder Spam oder Nicht-Spam zugeordnet, indem in den Trainings Daten diejenigen Nachrichten gesucht und analysiert werden, welche die größte Ähnlichkeit mit der neuen Nachricht besitzen. Dafür wird das k-Nächste-Nachbarn (k-Nearest-Neighbor) Verfahren verwendet. Die Nachrichten x und y heißen hierbei Nachbarn, falls x unter den k nächsten Nachbarn von y ist. Um zu entscheiden welche Nachrichten gerade die k nächsten Nachbarn sind, gibt es verschiedene Verfahren, wobei ich weiter unten eines davon genauer vorstellen werde. Die neue Nachricht wird dann der Klasse zugeordnet, welcher die Mehrheit der Nachbarn zugeordnet ist. Der Hauptunterschied zwischen diesem Verfahren und dem naiven Bayes-Verfahren liegt darin, dass das Bayes-Verfahren bevor es eine neu zu klassifizierende Nachricht sieht, bereits ein Modell aus Beispieldaten ermittelt hat. Dagegen hat das Verfahren des Fallbasierten Schließens kein solches Modell, sondern klassifiziert eine neue Nachricht, indem es die ähnlichsten bereits gespeicherten Nachrichten sucht. Dadurch wird dieses Verfahren bei der eigentlichen Klassifizierungsphase zeitaufwändiger, braucht aber weniger Zeit um neue Beispiele den gespeicherten Daten hinzuzufügen bzw. um ein globales Modell zu finden. Beim Bayes-Verfahren ist dies also genau umgekehrt, da bei jeder neu hinzugefügten Nachricht alle Wahrscheinlichkeiten neu berechnet werden müssen.

3.1 Grundlagen für den Fallbasierten Filter TiMBL

Wie schon erwähnt gehören die beiden neuen Ansätze des Fallbasierten bzw. Beispiel-Basierten Schließens oder Lernen, die ich hier vorstellen möchte auch zu den „*lazy learning systems*“. Während also beide Ansätze vom Prinzip die gleiche Strategie verfolgen, gibt es doch auch ein Paar unterschiede im Aufbau und Ablauf, die ich nun vorstellen werde.

3.1.1 Benchmark Corpus

Bisher gibt es keine standardisierten Nachrichtenmengen, die unter allen Forschern ausgetauscht werden könnte und somit zu einheitlichen Experimenten mit vergleichbaren Ergebnissen führen könnte. Das ist von daher schwierig, da man zwar Spam-Nachrichten ohne weiteres veröffentlichen kann, da sie ohnehin wahllos an eine große Zahl von Menschen geschickt werden und damit eigentlich schon öffentlich verfügbar sind. Bei Nicht-Spam Nachrichten ist dies nicht ohne weiteres möglich, da sehr leicht die Privatsphäre von Sendern und Empfängern verletzt werden könnte.

Daher wird in diesem 1. Ansatz ein Spam-Filter für *Mailing Lists* untersucht, da man dort die Nachrichten ohne Verletzung der Privatsphäre untereinander austauschen kann. In diesem Fall untersucht man anstatt den Nachrichten die ein einzelner Nutzer bekommt, die Nachrichten die beim Verteilungsserver der Liste eintreffen und sendet sie erst danach an die Abonnenten der Liste.

Mailinglisten werden häufig von Spammern „angegriffen“ die entweder nicht zwischen Adressen von Mailinglisten und Privatnutzern unterscheiden können, oder aber gerade versuchen über die Mailinglisten alle Abonnenten mit ihrer Spam-Nachricht zu erreichen.

In beiden Fällen kann es aufgrund der Vielzahl von Abonnenten zu einer großen Verschwendung von Bandbreite und Speicherplatz kommen.

Um das zu verhindern sind viele Listen durch einen Moderator überwacht, der jede eingehende Nachricht lesen und überprüfen muss bevor er sie an die Abonnenten sendet.

Aufgrund der enormen Größe von vielen Listen finden sich oft nicht genügend Moderatoren oder die Moderatoren sind einfach überfordert. Daher wäre gerade auch hier ein Filter nötig der automatisch die Spam-Nachrichten filtert, oder sie zumindest für die Moderatoren als solche markiert.

Die meisten Listen sind allerdings auf ein Thema spezialisiert und so können Filter das Fehlen bestimmter Fachsprache als Anzeichen für Spam heranziehen. Deshalb kann das Ergebnis nicht ohne weiteres auf Filter für einzelne Privatnutzer übernommen werden, es bietet aber trotzdem ein Anzeichen auf dessen praktische Anwendung und Filter für Mailinglisten sind wie oben beschrieben sowieso schon für sich wertvoll.

Die Linguist List, die hier bei diesem Ansatz verwendet wird, ist eine solche moderierte Liste über Sprachwissenschaften. Sie besteht aus 2893 Nachrichten, wovon 2412 Nicht-Spam Nachrichten sind und 481 Spam-Nachrichten. Dies entspricht ungefähr 16% an Spam-Nachrichten, einem Wert der ungefähr dem entspricht, was Cranor and LaMacchia (1998) und Sahami (1998) herausgefunden haben. Da das Experiment von 2001 ist entspricht diese Spam-Quote nicht mehr dem heutigen Stand, bei dem man von ungefähr 50-70% (vzbv Berlin 2005) an Spam-Nachrichten ausgeht.

Hierbei sind die Nachrichten der Linguist List nicht so sehr Thema spezifisch wie man eigentlich annehmen müsste. Sie enthalten z.B. Jobangebote, Softwareankündigungen und sogar aufbrausende Antworten. Aufgrund der doch geringen Anzahl an Nachrichten wurde in diesem Ansatz die 10-fold stratified cross-validation Technik (siehe 2.3) angewandt.

3.1.2 Preprocessing der Nachrichten

Wie bereits in Punkt 2.1 beschrieben wird jeder der Nachrichten als Vektor dargestellt. Die Attribute sind beim *TiMBL* Filter gleichbedeutend mit Worten, das heißt eine 1, falls das Wort in der Nachricht auftaucht und eine 0, falls nicht.

Damit verschieden Formen eines Wortes nicht als unterschiedliche Wörter behandelt werden, wird ein so genannter *Lemmatizer* eingesetzt, der jedes Wort auf seine Stammform bringt. So wird z.B. aus dem Englischen „was“ ein „be“.

Es ist zwar auch möglich Attribute in Zusammenhang mit Satzteilen wie z.B. „be over 21“, oder „*non-textual characteristics*“ (wie z.B. das Vorhandensein eines Attachments, oder ob eine Nachricht an einem Sonntag verschickt wurde, und man feststellte, dass Nicht-Spam Nachrichten bei einer bestimmten Mailingliste sehr selten am Wochenende verschickt werden) zu benutzen. Beides wurde hier aber nicht berücksichtigt, da die Berücksichtigung von Satzteilen zu keiner konstanten Verbesserung der Klassifizierung führte und „*non-textual characteristics*“ zwar eine Verbesserung liefern können, aber zusätzlichen manuellen Aufwand nach sich ziehen, man aber gerade einen voll automatischen Filter haben möchte.

Des Weiteren hat man, um die Anzahl der verwendeten Attribute zu reduzieren, alle Wörter die in weniger als 4 Nachrichten vorkamen, weggelassen.

Danach hat man den Information Gain (siehe 2.2) aller Attribute berechnet und diejenigen m Attribute mit dem höchsten IG-Wert ausgewählt, wobei m im Experiment zwischen 50 und 700 und 50-iger Schritten variiert wurde.

Unter den Wörtern mit den höchsten IG-Werten sind vor allem Wörter die häufig in Spam-Nachrichten und selten in Nicht-Spam Nachrichten vorkommen, wie z.B. „remove“, „free“, „your“ und „money“, oder aber Wörter die häufig in Nachrichten über Sprachwissenschaften und selten in Spam-Nachrichten vorkommen, wie „language“, „university“ oder „linguistic“.

Satzzeichen sowie andere Symbole wie z.B. „!“ , „@“ , „%“ und „\$“ wurden nicht weggelassen, sondern auch als „Wörter“ behandelt. Diese Zeichen gehören auch zu den „Wörtern“ mit den höchsten IG-Werten, was auch nicht verwundert, da sie mit viel größerer Häufigkeit in Spam-Nachrichten vorkommen, als in Nicht-Spam Nachrichten.

3.1.3 K-Nächste-Nachbarn

Wie schon erwähnt gibt es verschiedene Verfahren um den Abstand zwischen zwei Nachrichten zu berechnen. *TiMBL* verwendet hier das so genannte *overlap* Verfahren. Hierbei wird die Anzahl der Attribute gezählt bei denen die beiden Nachrichten unterschiedliche Werte haben.

Seien $\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$ und $\vec{x}_j = (x_{j1}, x_{j2}, \dots, x_{jn})$ zwei Nachrichten, so ist ihre *overlap* Entfernung:

$$d(\vec{x}_i, \vec{x}_j) \equiv \sum_{r=1}^n \delta(x_{ir}, x_{jr})$$

wobei

$$\delta(x, y) \equiv \begin{cases} 0, & \text{if } x=y \\ 1, & \text{otherwise} \end{cases}$$

Im Gegensatz zum sonst üblichen k-Nächste-Nachbarn Verfahren, wo genau die k nächsten Nachrichten ausgewählt werden, wählt *TiMBL* alle Nachrichten mit den k nächsten Entfernungen aus. Das bedeutet falls es bei einigen der k nächsten Entfernungen mehrere Nachrichten mit der gleichen Entfernung gibt, enthält die „Nachbarschaft“ (*neighborhood*) mehr als k Nachrichten.

3.1.4 Cost-sensitive classification

Wie bereits unter 2.4 erwähnt gibt es zwei verschiedene Arten von Fehlern beim Filtern von E-Mails. Eine erwünschte Nachricht als Spam zu klassifizieren (im Weiteren auch mit $L \rightarrow S$ bezeichnet) sowie eine Spam-Nachricht als erwünschte Nachricht zu klassifizieren und damit nicht zu blockieren (im Weiteren auch $S \rightarrow L$). Im Allgemeinen wird angenommen, dass das Blockieren einer erwünschten Nachricht (d.h. einer Nicht-Spam Nachricht) für einen Benutzer weitaus schlimmer ist als ab und zu einmal eine Spam-Nachricht zu bekommen, die man dann von Hand löschen müsste.

Deshalb muss man beim Klassifizieren von Nachrichten auch noch eine Kostenkomponente berücksichtigen. Dabei nimmt man an, dass $L \rightarrow S$ (d.h. *False Positive*) um das λ -Fache kostenintensiver ist als $S \rightarrow L$ (d.h. *False Negative*). Kosten bedeutet hier den zusätzlichen Aufwand, den ein Moderator oder ein Abonnent einer Mailingliste hat, um den Fehler des Filters wieder rückgängig zu machen. Im Einzelnen wird die untenstehende Kostentabelle verwendet, wobei $L \rightarrow L$ und $S \rightarrow S$ bedeutet eine Nachricht richtigerweise als Spam oder Nicht-Spam zu klassifizieren, was den Kosten 0 entspricht, da der Filter keine Fehler gemacht hat.

Tabelle 1: Kostentabelle

	Classified as legitimate	Classified as spam
Legitimate message	$c(L \rightarrow L) = 0$	$c(L \rightarrow S) = \lambda$
Spam message	$c(S \rightarrow L) = 1$	$c(S \rightarrow S) = 0$

Seien nun $W_L(\vec{x})$ und $W_S(\vec{x})$ die „degrees of confidence“ des Filters, das die Nachricht \vec{x} eine erwünschte Nachricht bzw. eine Spam-Nachricht ist. Für das k-Nächste-Nachbarn Verfahren entsprechen diese Werte gerade dem prozentualen Anteil der k Nachbarn, die der entsprechenden Klassen zugehören. Das bedeutet, dass wenn z.B. von 10 Nachbarn 4 zu der Kategorie Spam gehören, der Wert von $W_S(\vec{x})$ gerade 0.4 und der Wert von $W_L(\vec{x})$ gerade 0.6 ist.

Damit wird eine Nachricht genau dann als Spam klassifiziert, wenn die Kosten sie als Nicht-Spam zu klassifizieren größer sind als die Kosten um sie als Spam zu klassifizieren.

Das bedeutet:

$$W_S(\vec{x}) \times c(S \rightarrow L) + W_L(\vec{x}) \times c(L \rightarrow L) > W_L(\vec{x}) \times c(L \rightarrow S) + W_S(\vec{x}) \times c(S \rightarrow S)$$

Aus der Kostentabelle folgt nach ein paar Umformungen:

$$\vec{x} \mapsto S \text{ genau dann, wenn } W_S(\vec{x}) > t, \quad \text{mit } t = \frac{\lambda}{1 + \lambda}, \quad \lambda = \frac{t}{1 - t}$$

Hierbei ist t die so genannte Klassifizierungsschwelle. Eine Nachricht die diesen Wert übersteigt wird als Spam-Nachricht, sonst als Nicht-Spam Nachricht klassifiziert.

In diesem Ansatz wurden 3 verschiedene Werte für t bzw. λ getestet. Den Wert $t = 0.999$ was $\lambda = 999$ entspricht und bedeutet eine erwünschte Nachricht irrtümlicherweise als Spam zu klassifizieren ist genauso schlimm wie 999 Spam-Nachrichten irrtümlicherweise als erwünschte Nachrichten zu klassifizieren. Dies mag richtig sein, wenn man Spam-Nachrichten automatisch löscht, falls aber der Filter z.B. nur als Unterstützung für den Moderator einer Liste die Nachrichten vorsortiert ist ein anderer Wert zu benutzen. So könnte man sich vorstellen, dass ein Filter eine Spam-Nachricht an den Sender zurücksendet und den Grund dafür erklärt, sowie eine E-Mail Adresse eines Moderators angibt, damit der Sender die Nachricht einem Moderator noch einmal zur manuellen Klassifizierung vorlegen kann.

In diesem Fall könnte man sich die Werte $t = 0.9$ bzw. $\lambda = 9$ als passend vorstellen. Als dritten Fall könnte man sich vorstellen, dass der Filter die Spam-Nachrichten nur als solche markiert. Hierbei ist keiner der beiden Fehlerarten schlimmer als der Andere und damit sind die Werte $t = 0.5$ bzw. $\lambda = 1$ vernünftig.

3.1.5 Bewertung

Um die Ergebnisse der Experimente vergleichen und bewerten zu können, gibt es verschiedene Messgrößen. Accuracy (Acc) und error rate (Err = 1 - Acc) sind wie folgt definiert:

$$\text{Acc} = \frac{N_{L \rightarrow L} + N_{S \rightarrow S}}{N_L + N_S} \quad \text{und} \quad \text{Err} = \frac{N_{L \rightarrow S} + N_{S \rightarrow L}}{N_L + N_S}$$

Wobei $N_{Y \rightarrow Z}$ die Anzahl der Nachrichten in Kategorie Y ist die der Filter als Z klassifiziert und

$N_L = N_{L \rightarrow L} + N_{L \rightarrow S}$ die Gesamtanzahl der zu klassifizierenden Nicht-Spam Nachrichten und

$N_S = N_{S \rightarrow S} + N_{S \rightarrow L}$ die Gesamtanzahl der Spam-Nachrichten ist.

Mit der Berücksichtigung der Gewichtung der beiden Fehlerarten folgt:

$$\text{WAcc (weighted accuracy)} = \frac{\lambda \times N_{L \rightarrow L} + N_{S \rightarrow S}}{\lambda \times N_L + N_S}$$

und

$$\text{WErr (weighted error rate)} = \frac{\lambda \times N_{L \rightarrow S} + N_{S \rightarrow L}}{\lambda \times N_L + N_S}$$

Hierbei wird jede erwünschte Nachricht wie λ viele Nachrichten behandelt und das irrtümlicherweise Blockieren zählt wie λ Fehler und entsprechend die richtige Klassifizierung wie λ Erfolge.

Für aussagekräftigere Werte werden WAcc und WErr mit einer so genannten *baseline* verglichen. Die *baseline* in diesem Experiment entspricht genau dem Fall, dass überhaupt kein Filter verwendet wird.

$$\text{WAcc}^b = \frac{\lambda \times N_L}{\lambda \times N_L + N_S} \quad \text{und} \quad \text{WErr}^b = \frac{N_S}{\lambda \times N_L + N_S}$$

Daraus folgt also die so genannte *total cost ratio* (TCR) mit:

$$\text{TCR} = \frac{\text{WErr}^b}{\text{WErr}} = \frac{N_S}{\lambda \times N_{L \rightarrow S} + N_{S \rightarrow L}}$$

Je größer der TCR Wert ist, desto besser ist die Performance des Filters. Für $\text{TCR} < 1$ wäre es besser den Filter nicht zu benutzen.

Zwei andere Größen zur Bewertung der Ergebnisse sind *recall* (R) und *precision* (P), die wie folgt definiert sind:

$$\text{R} = \frac{N_{S \rightarrow S}}{N_{S \rightarrow S} + N_{S \rightarrow L}} \quad \text{und} \quad \text{P} = \frac{N_{S \rightarrow S}}{N_{S \rightarrow S} + N_{L \rightarrow S}}$$

Recall misst also den prozentualen Anteil der Spam-Nachrichten, die der Filter richtigerweise als Spam klassifiziert, also sozusagen die Wirksamkeit des Filters. *Precision* misst wie viele der blockierten Nachrichten auch wirklich Spam-Nachrichten sind, also sozusagen die Sicherheit oder Genauigkeit des Filters.

3.2 Ergebnisse des Fallbasierten Filters TiMBL

3.2.1 Attribute weighting

Im ersten Telexperiment wurde untersucht wie sich das so genannte *attribute weighting*, also das Gewichten der einzelnen Attribute auf die Klassifizierung des Filters auswirkt. Hierbei werden die einzelnen Attribute wiederum mit Hilfe von Information Gain, in Abhängigkeit ihres Nutzens für die Klassifizierung, gewichtet. Dadurch ändert sich die Formel zur Berechnung der Abstände zweier Nachrichten folgendermaßen:

$$d(\vec{x}_i, \vec{x}_j) \equiv \sum_{r=1}^n \omega_r \times \delta(x_{ir}, x_{jr}), \text{ wobei } \omega_r \text{ die Gewichtung des } r\text{-ten Attributes ist.}$$

Um die Ergebnisse vergleichen zu können, wurde das Experiment sowohl ohne *attribute weighting* als auch mit durchgeführt.

Für $\lambda = 1$ und $\lambda = 9$ und für $k = 1$, $k = 2$ und $k = 10$ (d.h. also für 1-Nächste-Nachbarn Verfahren, usw.) zeigen die Ergebnisse, dass *attribute weighting* durchgehend bessere Ergebnisse, also höhere TCR Werte (siehe 3.1.5) liefert als Filter, die kein *attribute weighting* benutzen.

Für $\lambda = 999$ liefert *attribute weighting* nicht durchgehend besser Ergebnisse, sondern schwankt zum Teil sogar unter die *baseline* (siehe 3.1.5), also dem Fall, dass gar kein Filter verwendet wurde.

Dies lässt sich mit der starken „Bestrafung“ der *False Positive* erklären, sodass schon das Klassifizieren einer erwünschten Nachricht als Spam ausreicht um den TCR Wert unterhalb die *baseline* zu bringen.

Insgesamt zeigen die Ergebnisse das *attribute weighting* eine positive Auswirkung auf die Klassifizierung hat und deshalb auch in allen weiteren Telexperimenten eingesetzt wurde.

3.2.2 Distance weighting

In diesem Telexperiment wurde das Verfahren des so genannten *distance weighting* und dessen Auswirkung auf die Klassifizierung untersucht. Hierbei werden Nachrichten deren Entfernung zu der untersuchten Nachricht kleiner ist, stärker gewichtet als Nachrichten, die weiter entfernt liegen. Dadurch ändert sich natürlich der Wert von $W_G(\vec{x})$ (siehe 3.1.4).

Der größte Vorteil dieses Verfahrens liegt darin, dass es die Wahl des k-Wertes unwichtiger macht. So ist z.B. ein k-Wert sehr gut für eine „Region“ mit wenigen Nachrichten geeignet, aber eher schlecht für eine „Region“ mit sehr vielen Nachrichten dicht an einer Stelle zusammengedrängt. In letzterem Fall könnte dies dazu führen, dass sehr viele irrelevante Nachrichten mit in die Klassifizierung einbezogen werden.

Für die Gewichtung wurden folgende zwei Formeln benutzt:

$$f_0(d) = d_{\max} - d \quad \text{und} \quad f_n(d) = \frac{1}{d^n}, n = 1, 2, 3, \dots$$

Hierbei ist $f_0(d)$ die Gewichtung einer Nachricht mit dem Abstand d zur untersuchten Nachricht und d_{\max} die größte mögliche Entfernung, die auftritt falls sich zwei Nachrichten in allen Attributen unterscheiden.

Falls eine oder mehrere Nachrichten mit der zu untersuchenden Nachricht identisch sind (d.h. $d = 0$), wird die zu untersuchende Nachricht der Klasse zugeordnet, der die Mehrheit der identischen Nachrichten zugeordnet ist und alle anderen Nachrichten werden ignoriert.

Die Ergebnisse zeigen, dass *distance weighting* die Klassifizierung verbessert. Die besten TCR Werte wurden für $f_2(d)$ und $f_3(d)$ erzielt. Die Ergebnisse zeigten weiterhin, dass für $n > 3$ keine signifikante Verbesserung mehr erzielt werden könnte.

3.2.3 Neighborhood size und dimensionality

In diesem Telexperiment wurden die Auswirkungen der so genannte *neighborhood size* (Anzahl der berücksichtigten Nachbarn), also den verschiedenen k-Werten und der *dimensionality*, also der Anzahl der berücksichtigten Attribute, untersucht.

Aufgrund der Ergebnisse der vorherigen Telexperimente wurden in diesem Experiment wieder Information Gain zur Gewichtung der Attribute und $f_3(d) = \frac{1}{d^3}$ zur Gewichtung der Entfernung benutzt.

Die Ergebnisse dieses Telexperimentes sind weniger deutlich bzw. aussagekräftig. Insgesamt lässt sich aber feststellen, dass die Performance des Filters mit steigender Anzahl berücksichtigter Attribute (wie schon erwähnt zwischen 50 und 700) besser wird. Für die k-Werte lässt sich keine einheitliche Aussage treffen, was zum Teil

durch die Verwendung von *distance weighting* erklärt werden kann, das wie schon erwähnt die Wahl der k -Werte weniger wichtig macht.

3.2.4 Vergleich der Ergebnisse mit einem naiven Bayes-Filter

In diesem Abschnitt werden nun die Ergebnisse dieses Fallbasierten Filters (wieder Information Gain zur Gewichtung der Attribute und $f_3(d) = \frac{1}{d^3}$ zur Gewichtung der Entfernung) mit den Ergebnissen eines naiven Bayes-Filters verglichen. Die untenstehenden Tabellen zeigen die besten Konfiguration für die jeweiligen λ -Werte, d.h. also für die 3 verschiedenen Szenarien im Umgang mit Spam-Nachrichten (siehe 3.1.4).

Tabelle 2: Beste Resultate für den Fallbasierten Filter

λ	k	Dimensionality	Recall (%)	Precision (%)	TCR
1	8	600	88.60	97.39	7.18
9	2	700	81.93	98.79	3.64
999	7	600	59.91	100	2.49

Tabelle 3: Beste Resultate für einen naiven Bayes-Filter

λ	Dimensionality	Recall (%)	Precision (%)	TCR
1	100	82.35	99.02	5.41
9	100	77.57	99.45	3.82
999	300	63.67	100	2.86

Hierbei ist zu beachten das im 3. Fall, also für $\lambda = 999$, die Sicherheit (*Precision*), also keine erwünschte Nachricht als Spam-Nachricht zu klassifizieren, zum entscheidenden Punkt wird. Deshalb muss *Precision* unter allen Umständen bei 100% gehalten werden.

Vergleicht man nun die beiden Tabellen, sieht man, dass der Fallbasierte Filter besser als der naive Bayes-Filter abschneidet. Die TCR-Werte sind deutlich besser für $\lambda = 1$, etwas schlechter für $\lambda = 9$ und etwas besser für $\lambda = 999$. Hierbei (für $\lambda = 999$) ist zu erwähnen das die Performance des naive Bayes-Filters nicht einheitlich diesen angegebenen Wert erreicht, d.h. dieser Wert wird nur an einer bestimmten, sehr kleinen Stelle erreicht. Außer an dieser einen Stelle übersteigt der Filter nie $TCR = 1$ für $\lambda = 999$ während der Fallbasierte Filter in bestimmten Intervallen über diesem Wert bleibt.

Für *Recall* und *Precision* sieht man, dass der Fallbasierte Filter die *Recall*-Werte für $\lambda = 1$ und $\lambda = 9$ mit kleiner Verschlechterung der Genauigkeit (*Precision*), verbessert.

3.3 Grundlagen für den Fallbasierten Filter ECUE

In diesem Ansatz wird der *ECUE* Filter vorgestellt, das für *E-Mail Classification Using Examples* steht. Diese Software kann automatisch neue Nachrichten zu ihrem Grundbestand hinzufügen und deshalb wurde hier vor allem auch eine Technik zur Verwaltung dieser Trainingsdaten untersucht.

Aus diesem Grund wurde dieser Ansatz auch in zwei Teile unterteilt. Im ersten Teil (Static Evaluation) wurden die Trainingsdaten konstant (static) gehalten und im zweiten Teil (Dynamic Evaluation) wurden die Trainingsdaten um zuvor falsch klassifizierten Nachrichten erweitert.

3.3.1 Benchmark Corpus

In diesem Ansatz wurden anders als im ersten nicht die E-Mails einer Mailingliste untersucht, sondern 10,000 Nachrichten von zwei Privatnutzern.

Als Grund wird genannt, dass bisherige Untersuchungen jeweils nur spezifische Nachrichten von Mailinglisten und „altmodische“ bzw. „veraltete“ Spam-Nachrichten untersuchten, die nicht die sich ständig ändernden Techniken der Spammer mit berücksichtigen.

Aus diesen 10,000 Nachrichten, die über einen Zeitraum von 18 Monaten gesammelt wurde und die Business- und Mailinglist-Nachrichten, sowie persönliche Nachrichten enthalten, wurden 4000 Nachrichten ausgewählt. Diese wurden wiederum in 4 Teile aufgeweilt, wobei jeder Teil 500 Spam-Nachrichten und 500 Nicht-Spam Nachrichten enthält und die die Trainingsdaten bilden werden.

Hierbei wurde im Experiment wieder die 50-fold stratified cross-validation Technik (siehe 2.3) angewandt um die Ergebnisse aussagekräftiger zu machen.

3.3.2 Preprocessing der Nachrichten

Der *ECUE* Filter verwendet keinen *Lemmatizer*, alle Attachments wurden entfernt und der ganze HTML Text wurde berücksichtigt. Außerdem wurden der Betreff, der Sender und der Empfänger mit einbezogen.

Wie schon beschrieben werden die Nachrichten als Vektor dargestellt. Bei diesem Ansatz wurden sowohl Wörter als auch Buchstaben und Symbole als mögliche Attribute herangezogen. Es wurde, wie in 2.1 beschrieben, sowohl die binäre als auch numerische Darstellung der Attribute untersucht und bewertet.

Wie beim *TiMBL* Filter wurde die Anzahl der Attribute wieder mit Information Gain eingeschränkt. Vorherige Untersuchungen mit Attributen zwischen 100 und 1,000 zeigten, dass der beste Wert bei 700 verwendete Attributen liegt.

3.3.3 Case retrieval und case base management

Anstelle des klassischen k-Nächste-Nachbarn Verfahren benutzt *ECUE* ein anderes Verfahren um die k ähnlichsten Nachrichten innerhalb der Trainingsdaten zu finden, nämlich ein Algorithmus der auf den so genannten *Case Retrieval Nets* (CRNs) basiert.

CRNs sind aus folgenden Komponenten aufgebaut:

- Case Nodes, die die gespeicherten Nachrichten repräsentieren
- Information Entity Nodes (IEs), die die Attribute repräsentieren
- Relevance Arcs, die die Case Nodes mit den sie repräsentierenden IEs verbindet

Die Idee hinter den CRNs ist nun die, dass die zu klassifizierende Nachricht mit Relevance Arcs mit dem Netz verbunden und dadurch aktiviert wird. Diese Aktivierung breitet sich dann gewissermaßen über das Netz aus. Jeder der anderen Case Nodes berechnet dann einen „Aktivierungs-Wert“ in Abhängigkeit ihrer Ähnlichkeit zu der zu klassifizierenden Nachricht. Die Case Nodes mit der größten Aktivierung sind dann die, die die größte Ähnlichkeit mit der zu klassifizierenden Nachricht haben.

Die unten stehende Abbildung zeigt ein Beispiel eines solchen *Case Retrieval Nets* für Spam-Filterung.

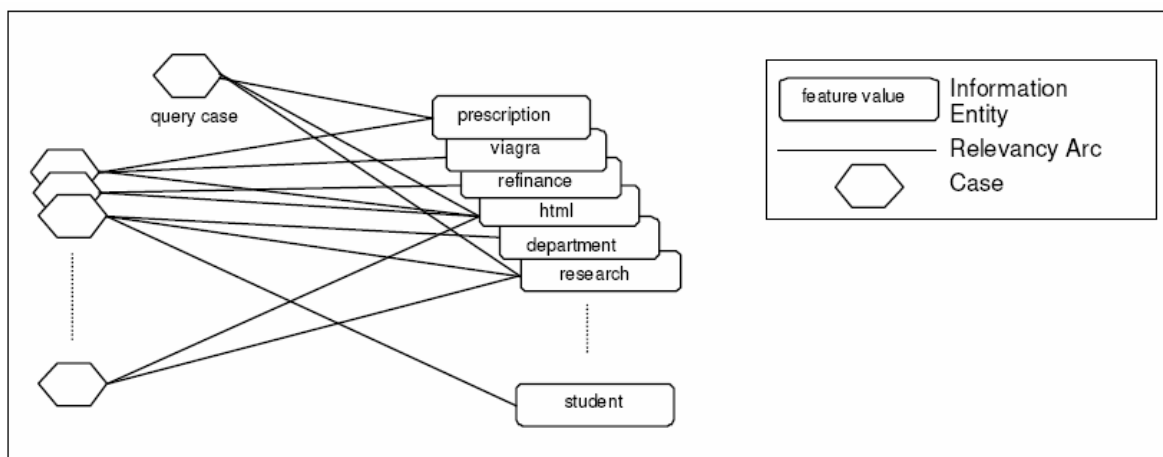


Fig. 1. Case Retrieval Net

Um die enorme Größe der Trainingsdaten zu verwalten und um automatisch neue Daten hinzufügen zu können, ist es nötig diese auf einer geeigneten Größe zu halten. Dies bezeichnet man auch als *case base editing*. Dazu muss ein Filter genau die Nachrichten löschen, die für die Klassifizierung weniger wichtig sind, oder mehrfach in ähnlicher Form auftreten, ohne die Klassifizierung zu verschlechtern.

ECUE verwendet hier die so genannte *Competence Based Editing* Technik, welche für jede Nachricht einen so genannten „Kompetenz-Modell“ erstellt. Hierbei wird der Nutzen (bei wie vielen Nachrichten trägt diese Nachricht zu richtigen Klassifizierung bei) und der Schaden (repräsentiert durch die Nachrichten die durch diese Nachricht falsch klassifiziert werden) den die jeweilige Nachricht anrichtet berücksichtigt.

Mit der Hilfe dieses Modells können nun redundante Nachrichten (d.h. Nachrichten die im Zentrum einer Gruppe von Nachrichten mit der gleichen Klassifizierung liegen) gelöscht werden, ohne die Klassifizierung zu verschlechtern.

3.3.4 Bewertung

Anders als beim *TiMBL* Filter wird beim *ECUE* Filter keine Gewichtung der Fehler vorgenommen. Zur Bewertung und zum Vergleichen der Ergebnisse dienen somit nur die Quote der *False Positive* (FPRate) und *False Negative* (FNRate) sowie ein Durchschnittsfehler $AvgError = (FPRate + FNRate)/2$.

3.4 Ergebnisse des Fallbasierten Filters ECUE

3.4.1 Static Evaluation

Das Ziel in diesem Teil der Untersuchung war es die beste Konfiguration des Filters festzustellen. D.h. sollen die Attribute binär oder numerisch dargestellt werden, und soll eine so genannte *case base editing* Technik (siehe 3.3.3) verwendet werden oder nicht. Als dritter Punkt wurde außerdem noch untersucht wie sich *attribute weighting* auswirkt.

Die besten Ergebnisse wurden erzielt als die Attribute numerisch dargestellt wurden und sowohl *attribute weighting*, als auch *case base editing* verwendet wurde. Knapper Zweiter wurde die Konfiguration mit binärer Darstellung der Attribute und *case base editing*, aber ohne *attribute weighting*.

Allerdings verlangsamt sich die Klassifizierung sehr stark, wenn man die Attribute numerisch darstellt. Die Verbesserung der Klassifizierungszeit durch das *Case Retrieval Net* stellt sich nur bei binärer Darstellung ein, nicht aber bei numerischer Darstellung. Die größere Verschlechterung entsteht allerdings erst, wenn man die Trainingsdaten mit *case base editing* Technik bearbeiten will, welche allerdings die Genauigkeit des Filters enorm steigert. Eine Konfiguration ohne *case base editing* bzw. einer schlechten Antwortzeit (response time) ist deshalb nicht für einen kommerziellen Spam-Filter geeignet.

Aus diesen Gründen wurde die Konfiguration mit binärer Darstellung der Attribute und *case base editing* der Konfiguration mit numerischer Darstellung vorgezogen.

3.4.2 Dynamic Evaluation

In diesem Teil wurde insbesondere das Lernen des Filters, d.h. also das Updaten der Trainingsdaten mit zuvor falsch klassifizierten Nachrichten, untersucht. Hierfür wurde der Filter über den Zeitraum von einem Jahr entweder ohne updaten oder mit monatlichen, wöchentlichen oder täglichen updaten der Trainingsdaten verwendet.

Da *False Positive* wie schon erwähnt schlimmere Fehler sind als *False Negative* wurde der Filter so konfiguriert, dass alle k nächsten Nachbarn einer untersuchten Nachricht mit Spam klassifiziert sein müssen um die neue Nachricht auch als Spam zu klassifizieren (*unanimous voting*). Des Weiteren wurde wiederum die binäre Darstellung der Attribute gewählt und *case base editing* verwendet.

Das beste Ergebnis ergab sich dann, wenn die Trainingsdaten täglichen mit den an diesem Tag falsch klassifizierten Nachrichten aktualisiert wurden.

3.4.3 Vergleich der Ergebnisse mit einem naiven Bayes-Filter

Im Vergleich der Ergebnisse des Fallbasierten Filters mit konstanten Trainingsdaten (siehe 3.4.1) mit den Ergebnissen eines naiven Bayes-Filters lässt sich kein eindeutiger Sieger feststellen. In zwei von vier Testdatensätzen (3.3.1) liefert der Bayes-Filter die besseren Ergebnisse und in den anderen zwei der Fallbasierte Filter.

Bei der Untersuchung mit dynamischen Trainingsdaten (siehe 3.4.2) lässt sich jedoch ein eindeutigeres Ergebnis feststellen. Der naive Bayes-Filter schneidet besser ab, falls die Trainingsdaten nicht aktualisiert werden, aber sobald diese dynamisch aktualisiert werden schneidet der Fallbasierte Filter besser ab.

In den unten stehenden Graphiken erkennt man, dass das tägliche Aktualisieren mit zuvor falsch klassifizierten Nachrichten die Performance des Fallbasierten Filters verbessert, während für den naiven Bayes-Filter sogar eine Verschlechterung eintritt. Tägliches Aktualisieren verbessert für den Bayes-Filter zwar die FPRate stärker als für den Fallbasierten Filter, die Verschlechterung der FNRate führt aber insgesamt zu der schlechteren Performance.

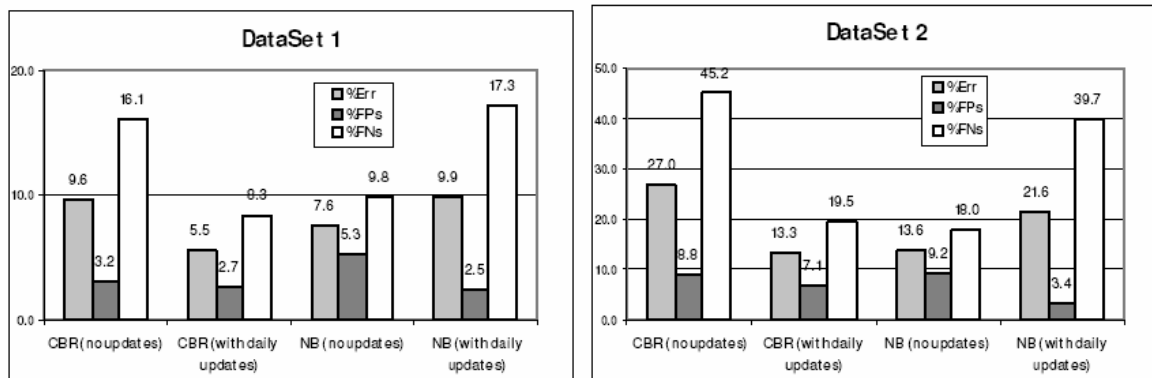


Fig. 5. Results of evaluations over a period of time

Es ist außerdem festzuhalten, dass das Aktualisieren eines naiven Bayes-Filters einen eigenen Lernprozess erfordert, bei dem die Wahrscheinlichkeiten für alle Nachrichten neu berechnet werden muss. Das Aktualisieren eines Fallbasierten Filters (wie den hier benutzten *ECUE*) mit neuen Trainingsdaten, erfordert hingegen einfach nur das Hinzufügen neuer Nachrichten.

4 Zusammenfassung und Ausblick

Insgesamt lässt sich nun sagen, dass ein Fallbasierter Spam-Filter praktisch durchführ- und anwendbar ist und mit konstanten Trainingsdaten (*case base*) zumindest kein schlechteres, und mit dynamischem Aktualisieren der Trainingsdaten sogar ein besseres Ergebnis liefert als ein vergleichbarer Bayes-Filter.

Dies ist insofern auch wichtig, da Filter die wie der naive Bayes-Filter, auf Wahrscheinlichkeiten basieren schon patentiert sind und man deshalb nach Alternativen suchen muss.

Insbesondere die Fähigkeit einen Fallbasierten Filter ohne jeglichen Aufwand mit neuen Nachrichten zu Aktualisieren und ihn damit an die sich ständig ändernden Spam-Nachrichten anzupassen macht ihn zu einer Alternative die sogar bessere Ergebnisse liefert als der Bayes-Filter.

Die Ansätze die hier präsentiert wurden lassen aber noch weitere Aspekte offen, die noch untersucht werden können.

So könnte man z.B. noch untersuchen ob sich die Klassifizierung verbessert, falls „*non-textual characteristics*“ (siehe 3.1.2) berücksichtigt werden. Außerdem gibt es noch andere Verfahren um Attribute und Abstände zu gewichten, die möglicherweise bessere Ergebnisse liefern könnten als die hier vorgestellten. Des Weiteren könnte man nach Verfahren suchen um bei numerischer Darstellung der Attribute die Klassifizierungszeit und insbesondere die Zeit für das *case base editing* zu verbessern.

Eine weitere Möglichkeit bestehende Filter zu verbessern könnte auch sein Fallbasierte und Wahrscheinlichkeitsbasierte Verfahren zu kombinieren.

5 Literaturverzeichnis

1. Sakkis G., Androutsopoulos I., Paliouras G., Karkaletsis V., Spyropoulos C.D., Stamatopoulos P.: A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists. *Information Retrieval*, 6 (1), Kluwer Academic Publishers (2000) 49-73
2. Sarah Jane Delany, Pádraig Cunningham, Lorcan Coyle An Assessment of Case-Based Reasoning for Spam Filtering. Dublin Institute of Technology, Kevin St., Dublin 8, Ireland
3. Yang Y and Pedersen JO (1997) A comparative study on feature selection in text categorization. In: *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US, pp. 412–420.
4. Cranor LF and LaMacchia BA (1998) Spam!, *Communications of ACM*, 41(8):74–83.
5. Sahami M, Dumais S, Heckerman D and Horvitz E (1998) A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization—Papers from the AAAI Workshop*, Madison Wisconsin, pp. 55–62. AAAI Technical Report WS-98-05.
6. Verbraucherzentrale Bundesverband - vzbv Berlin 2005
Verbraucherschutz im Internet - Wie viel Vertrauen ist gerechtfertigt? Dossier zum Weltverbrauchertag 2005